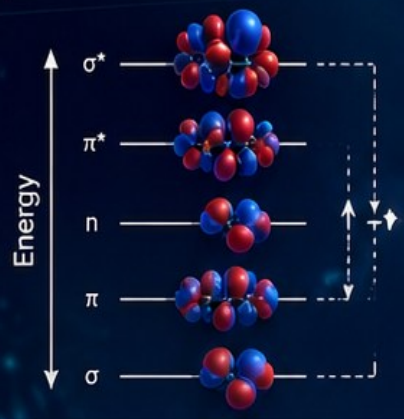
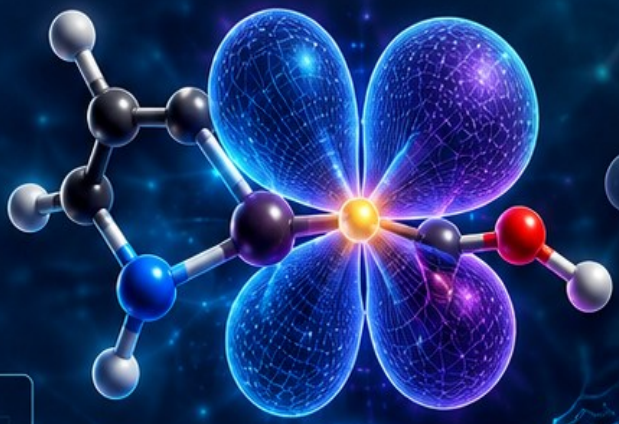
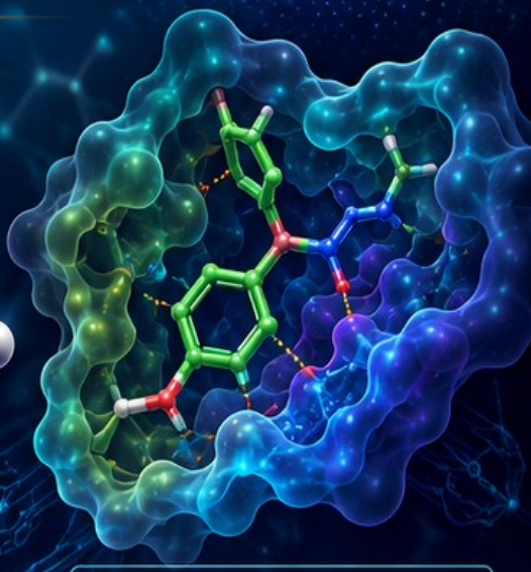
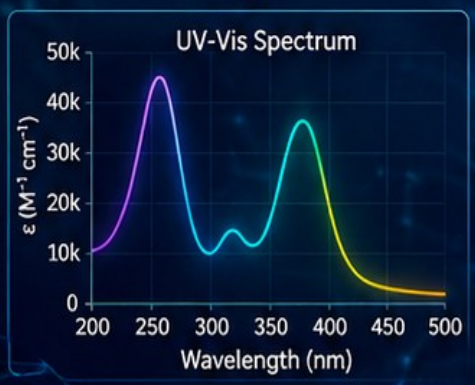
PRIORITAS SOFTWARE KIMIA KOMPUTASI

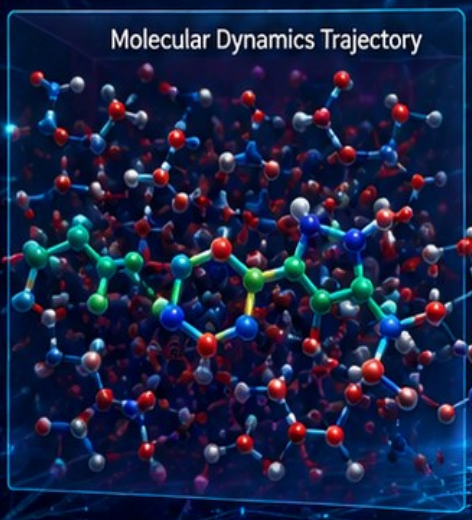




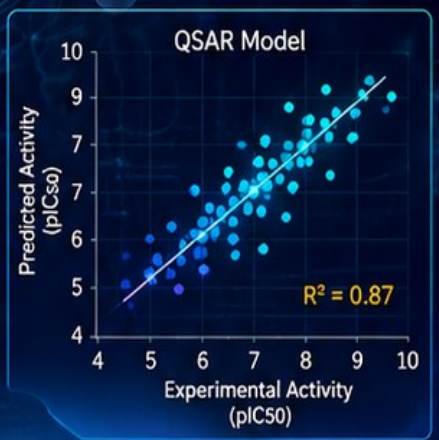














KASMUI

PRIORITAS SOFTWARE KIMIA KOMPUTASI

Buku Rancangan, Arsitektur, Validasi, dan Peta Jalan Pengembangan

Platform Python/Linux untuk Riset Kimia Komputasi

KASMUI

AGUSTUS 2026

KATA PENGANTAR

Kimia komputasi telah berkembang dari kegiatan yang berpusat pada satu kalkulasi menjadi ekosistem riset yang menuntut pengelolaan struktur, pemilihan metode, eksekusi dalam jumlah besar, analisis multimodal, validasi, dan pelaporan yang dapat diulang. Perubahan ini membuat kebutuhan terhadap software bukan lagi sebatas menyediakan tombol untuk menjalankan optimasi atau menghitung energi. Software yang bernilai harus mampu mengurangi bottleneck nyata: preparasi data yang rawan kesalahan, screening banyak molekul, parsing keluaran yang tidak seragam, pengelolaan job panjang, integrasi docking dan molecular dynamics, analisis spektrum, pembentukan dataset QSAR, serta pencatatan provenance yang membuat hasil dapat ditelusuri kembali.

Buku ini disusun dengan titik berangkat tersebut. Dokumen sumber menempatkan QM Screening & Workflow Manager, Reaction Path & Mechanism Explorer, SpectraQM Analyzer, DockingFlow, Molecular Dynamics Workbench, Conformer & Molecular Preparation Studio, QSAR/ADMET Studio, dan ReproChem Research Manager sebagai prioritas. Arah itu dipertahankan dan dikembangkan menjadi arsitektur buku yang membahas alasan ilmiah, kebutuhan fungsional, struktur data, alur komputasi, validasi, risiko, rancangan pengujian, dan hubungan antar-modul. Dengan demikian, pembahasan tidak hanya menjawab software apa yang perlu dibuat, tetapi juga mengapa modul tersebut penting dan bagaimana prinsip implementasinya dapat dijaga agar ilmiah serta maintainable.

Python dan Linux dipilih sebagai fondasi karena keduanya memungkinkan integrasi yang luas antara electronic structure, cheminformatics, machine learning, simulasi molekuler, basis data, plotting, dan antarmuka pengguna. PySCF menyediakan kerangka electronic-structure yang fleksibel; RDKit menangani banyak kebutuhan representasi dan cheminformatika; xTB menyediakan jalur perhitungan berbiaya rendah yang sesuai untuk prescreening; OpenMM memberi Python API yang kuat untuk molecular dynamics; sedangkan scikit-learn menyediakan ekosistem yang matang untuk model QSAR dan machine learning. Pemanfaatan library tersebut tetap membutuhkan lapisan desain yang mengatur data, provenance, error handling, resource management, dan scientific validation.

Salah satu gagasan utama buku ini adalah menghindari pembangunan aplikasi sebagai pulau-pulau yang tidak saling berkomunikasi. Struktur molekul yang telah dibersihkan pada MolPrep seharusnya dapat langsung digunakan oleh QMScreen atau DockFlow. Hasil optimasi DFT seharusnya dapat mengalir ke SpectraQM atau ReactionQM. Pose docking yang telah divalidasi dapat menjadi input MD. Descriptor yang dihasilkan dari RDKit, DFT, dan docking dapat membentuk dataset QSAR. Seluruh aliran tersebut sebaiknya dicatat oleh ReproChem agar data tidak hilang konteks ketika dipakai kembali oleh modul lain.

Pembahasan juga menekankan bahwa otomatisasi bukan alasan untuk menyembunyikan asumsi. Antarmuka yang sederhana harus tetap menampilkan parameter ilmiah yang menentukan hasil, seperti charge, multiplicity, metode, basis set, konvergensi, model pelarut, grid docking, force field, time step, pembagian dataset, dan metrik validasi. Default dapat membantu pengguna, tetapi default harus terdokumentasi, dapat diubah, dan tersimpan dalam record proyek. Prinsip ini penting terutama ketika software digunakan dalam pendidikan, karena kemudahan antarmuka tidak boleh memutus hubungan antara klik pengguna dan makna fisik atau statistik di balik perhitungan.

Buku ini memosisikan validasi sebagai proses berlapis. Unit test memeriksa fungsi kecil; integration test memeriksa komunikasi antar-modul; reference test membandingkan hasil numerik dengan data acuan; regression test menjaga agar pembaruan versi tidak mengubah perilaku secara tidak sengaja; dan scientific validation menilai apakah metode serta interpretasi memang cocok untuk pertanyaan penelitian. Pendekatan ini penting karena software dapat bebas dari error pemrograman tetapi tetap menghasilkan kesimpulan yang lemah apabila metode tidak sesuai atau validasi ilmiahnya tidak memadai.

Bagian latihan dirancang untuk menilai pemahaman konseptual sekaligus kemampuan desain. Pertanyaan tidak hanya meminta definisi istilah, tetapi mengajak pembaca menentukan workflow, mengidentifikasi risiko, merancang schema data, memilih tahap prescreening, menafsirkan kriteria transition state, menilai redocking, memeriksa stabilitas MD, mencegah data leakage pada QSAR, dan merancang provenance. Dengan cara itu, buku dapat digunakan sebagai bahan pengembangan software, referensi mata kuliah, maupun dasar diskusi tim riset.

Semoga buku ini membantu membangun perangkat komputasi kimia yang tidak sekadar dapat dijalankan, tetapi juga dapat dipercaya, diuji, dipelihara, dikembangkan, dan digunakan kembali. Orientasi akhirnya adalah ekosistem riset yang menyatukan perhitungan, data, analisis, dan pelaporan dalam satu alur yang transparan. Setiap modul yang dibahas dapat dikembangkan secara bertahap; nilai terbesar akan muncul ketika seluruh modul menggunakan prinsip data dan provenance yang sama sehingga membentuk satu platform komputasi kimia terpadu.

Semarang, Agustus 2026

Penyusun,

KASMUI

DAFTAR ISI

KATA PENGANTAR 3

DAFTAR ISI 5

BAB 1 LANDASAN, PRIORITAS, DAN ARSITEKTUR SOFTWARE KIMIA KOMPUTASI 23

1.1 Perubahan Paradigma dari Kalkulator Energi ke Platform Riset 23

1.1.1 Spesifikasi Fungsional dan Scientific Guardrails 25

1.1.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 27

1.1.3 Sketsa Implementasi Python..... 28

1.2 Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility29

1.2.1 Spesifikasi Fungsional dan Scientific Guardrails 31

1.2.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 32

1.2.3 Sketsa Implementasi Python..... 33

1.3 Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI 34

1.3.1 Spesifikasi Fungsional dan Scientific Guardrails 36

1.3.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 38

1.3.3 Sketsa Implementasi Python..... 39

1.4 Ekosistem Python dan Linux untuk Komputasi Kimia 40

1.4.1 Spesifikasi Fungsional dan Scientific Guardrails 41

1.4.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 43

1.4.3 Sketsa Implementasi Python..... 44

1.5 Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF 46

1.5.1 Spesifikasi Fungsional dan Scientific Guardrails 47

1.5.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 49

1.5.3 Sketsa Implementasi Python..... 50

1.6 Strategi Multi-Fidelity untuk Efisiensi Komputasi 51

1.6.1 Spesifikasi Fungsional dan Scientific Guardrails 53

1.6.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 54

1.6.3 Sketsa Implementasi Python 55
1.7 Model Data Molekul, Job, Result, dan Project 56
1.7.1 Spesifikasi Fungsional dan Scientific Guardrails 58
1.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 60
1.7.3 Sketsa Implementasi Python 60
1.8 Charge, Multiplicity, Unit, dan Validasi Input 62
1.8.1 Spesifikasi Fungsional dan Scientific Guardrails 63
1.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 65
1.8.3 Sketsa Implementasi Python 66
1.9 Manajemen CPU, Memori, GPU, dan Parallelism 67
1.9.1 Spesifikasi Fungsional dan Scientific Guardrails 69
1.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 71
1.9.3 Sketsa Implementasi Python 71
1.10 Error Handling, Resume, Checkpoint, dan Failure Recovery 73
1.10.1 Spesifikasi Fungsional dan Scientific Guardrails 74
1.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 76
1.10.3 Sketsa Implementasi Python 77
1.11 Validasi Ilmiah, Benchmark, dan Regression Test 78
1.11.1 Spesifikasi Fungsional dan Scientific Guardrails 80
1.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 81
1.11.3 Sketsa Implementasi Python 82
1.12 Peta Prioritas Sembilan Modul Komputasi Jisamsu 83
1.12.1 Spesifikasi Fungsional dan Scientific Guardrails 85
1.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 87
1.12.3 Sketsa Implementasi Python 88
1.13 Rangkuman Bab 89
BAB 2 QMSCREEN: HIGH-THROUGHPUT QUANTUM CHEMISTRY DAN WORKFLOW MANAGER 91
2.1 Posisi QMScreen sebagai Perluasan QM Komputasi 92
2.1.1 Spesifikasi Fungsional dan Scientific Guardrails 94

2.1.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 96
2.1.3 Sketsa Implementasi Python..... 96
2.2 Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV 98
2.2.1 Spesifikasi Fungsional dan Scientific Guardrails..... 99
2.2.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 101
2.2.3 Sketsa Implementasi Python..... 102
2.3 Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan 103
2.3.1 Spesifikasi Fungsional dan Scientific Guardrails..... 105
2.3.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 106
2.3.3 Sketsa Implementasi Python..... 107
2.4 Generate 3D dan Conformer Generation untuk Batch Besar 108
2.4.1 Spesifikasi Fungsional dan Scientific Guardrails..... 110
2.4.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 112
2.4.3 Sketsa Implementasi Python..... 112
2.5 Prescreening GFN2-xTB dan Seleksi Kandidat.....114
2.5.1 Spesifikasi Fungsional dan Scientific Guardrails..... 116
2.5.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 118
2.5.3 Sketsa Implementasi Python..... 119
2.6 Optimasi Geometri DFT dan Strategi Basis Set Bertingkat.....120
2.6.1 Spesifikasi Fungsional dan Scientific Guardrails..... 121
2.6.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 123
2.6.3 Sketsa Implementasi Python..... 124
2.7 Frequency, Thermochemistry, dan Pemeriksaan Minimum 125
2.7.1 Spesifikasi Fungsional dan Scientific Guardrails..... 127
2.7.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 128
2.7.3 Sketsa Implementasi Python..... 129
2.8 HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum.....130
2.8.1 Spesifikasi Fungsional dan Scientific Guardrails..... 132
2.8.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 134

2.8.3 Sketsa Implementasi Python 135
2.9 Scheduling Paralel, Resource Limit, dan Fair Queue 136
2.9.1 Spesifikasi Fungsional dan Scientific Guardrails 138
2.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 140
2.9.3 Sketsa Implementasi Python 140
2.10 Resume Calculation, Deduplication, dan Cache Hasil 142
2.10.1 Spesifikasi Fungsional dan Scientific Guardrails 143
2.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 145
2.10.3 Sketsa Implementasi Python 146
2.11 Ranking Multikriteria dan Ekspor Data Riset 147
2.11.1 Spesifikasi Fungsional dan Scientific Guardrails 149
2.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 150
2.11.3 Sketsa Implementasi Python 151
2.12 Benchmark Kinerja dan Validasi Hasil High-Throughput 152
2.12.1 Spesifikasi Fungsional dan Scientific Guardrails 154
2.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 156
2.12.3 Sketsa Implementasi Python 157
2.13 Rangkuman Bab 158
BAB 3 REACTIONQM: JALUR REAKSI, TRANSITION STATE, DAN MEKANISME 160
3.1 Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme 161
3.1.1 Spesifikasi Fungsional dan Scientific Guardrails 163
3.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 165
3.1.3 Sketsa Implementasi Python 165
3.2 Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping 167
3.2.1 Spesifikasi Fungsional dan Scientific Guardrails 168
3.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 170
3.2.3 Sketsa Implementasi Python 171
3.3 Optimasi Minimum Reaktan dan Produk 172
3.3.1 Spesifikasi Fungsional dan Scientific Guardrails 174

3.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 175
3.3.3 Sketsa Implementasi Python 176
3.4 Penyusunan Guess Transition State yang Rasional 177
3.4.1 Spesifikasi Fungsional dan Scientific Guardrails 179
3.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 181
3.4.3 Sketsa Implementasi Python 181
3.5 Algoritma Transition-State Search dan Hessian 183
3.5.1 Spesifikasi Fungsional dan Scientific Guardrails 185
3.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 187
3.5.3 Sketsa Implementasi Python 188
3.6 Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal 189
3.6.1 Spesifikasi Fungsional dan Scientific Guardrails 190
3.6.2 Risiko, Failure Mode, dan Strategi Mitigasi 192
3.6.3 Sketsa Implementasi Python 193
3.7 Intrinsic Reaction Coordinate dan Verifikasi Konektivitas 194
3.7.1 Spesifikasi Fungsional dan Scientific Guardrails 196
3.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 197
3.7.3 Sketsa Implementasi Python 198
3.8 Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs 199
3.8.1 Spesifikasi Fungsional dan Scientific Guardrails 201
3.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 203
3.8.3 Sketsa Implementasi Python 204
3.9 Energi Aktivasi dan Interpretasi Kinetika 205
3.9.1 Spesifikasi Fungsional dan Scientific Guardrails 207
3.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 209
3.9.3 Sketsa Implementasi Python 209
3.10 Reaksi Multitahap, Intermediate, dan Reaction Network 211
3.10.1 Spesifikasi Fungsional dan Scientific Guardrails 212
3.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 214

3.10.3 Sketsa Implementasi Python 215
3.11 Visualisasi Diagram Energi dan Bond Change 216
3.11.1 Spesifikasi Fungsional dan Scientific Guardrails 218
3.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 219
3.11.3 Sketsa Implementasi Python 220
3.12 Benchmark dan Validasi Mekanisme Reaksi 221
3.12.1 Spesifikasi Fungsional dan Scientific Guardrails 223
3.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 225
3.12.3 Sketsa Implementasi Python 226
3.13 Rangkuman Bab 227
BAB 4 SPECTRAQM: SPEKTROSKOPI KOMPUTASI DAN ANALISIS ORBITAL 229
4.1 Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika 230
4.1.1 Spesifikasi Fungsional dan Scientific Guardrails 232
4.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 234
4.1.3 Sketsa Implementasi Python 234
4.2 Frekuensi Harmonik dan Interpretasi Mode Vibrasi 236
4.2.1 Spesifikasi Fungsional dan Scientific Guardrails 237
4.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 239
4.2.3 Sketsa Implementasi Python 240
4.3 Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening 241
4.3.1 Spesifikasi Fungsional dan Scientific Guardrails 243
4.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 244
4.3.3 Sketsa Implementasi Python 245
4.4 Raman Activity dan Simulasi Spektrum Raman 246
4.4.1 Spesifikasi Fungsional dan Scientific Guardrails 248
4.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 250
4.4.3 Sketsa Implementasi Python 250
4.5 TDDFT, Excitation Energy, dan UV-Vis 252
4.5.1 Spesifikasi Fungsional dan Scientific Guardrails 254

4.5.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 256
4.5.3 Sketsa Implementasi Python..... 257
4.6 Oscillator Strength dan Assignment Transisi Elektronik 258
4.6.1 Spesifikasi Fungsional dan Scientific Guardrails..... 259
4.6.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 261
4.6.3 Sketsa Implementasi Python..... 262
4.7 NMR Shielding, Chemical Shift, dan Referensi 263
4.7.1 Spesifikasi Fungsional dan Scientific Guardrails..... 265
4.7.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 266
4.7.3 Sketsa Implementasi Python..... 267
4.8 HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals.....268
4.8.1 Spesifikasi Fungsional dan Scientific Guardrails..... 270
4.8.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 272
4.8.3 Sketsa Implementasi Python..... 273
4.9 DOS dan Projected Density of States.....274
4.9.1 Spesifikasi Fungsional dan Scientific Guardrails..... 276
4.9.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 278
4.9.3 Sketsa Implementasi Python..... 278
4.10 Perbandingan Spektrum Teoretis dan Eksperimen 280
4.10.1 Spesifikasi Fungsional dan Scientific Guardrails..... 281
4.10.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 283
4.10.3 Sketsa Implementasi Python..... 284
4.11 Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab285
4.11.1 Spesifikasi Fungsional dan Scientific Guardrails..... 287
4.11.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 288
4.11.3 Sketsa Implementasi Python..... 289
4.12 Desain Dashboard SpectraQM dan Ekspor Gambar/Data 290
4.12.1 Spesifikasi Fungsional dan Scientific Guardrails..... 292
4.12.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 294

4.12.3 Sketsa Implementasi Python 294
4.13 Rangkuman Bab 296
BAB 5 DOCKFLOW: MOLECULAR DOCKING DAN ANALISIS INTERAKSI LIGAN-PROTEIN 297
5.1 Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut 298
5.1.1 Spesifikasi Fungsional dan Scientific Guardrails 300
5.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 302
5.1.3 Sketsa Implementasi Python 302
5.2 Preparasi Ligan untuk Docking yang Konsisten 304
5.2.1 Spesifikasi Fungsional dan Scientific Guardrails 305
5.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 307
5.2.3 Sketsa Implementasi Python 308
5.3 Preparasi Protein, Protonasi, dan Penanganan Komponen Kristal 309
5.3.1 Spesifikasi Fungsional dan Scientific Guardrails 311
5.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 312
5.3.3 Sketsa Implementasi Python 313
5.4 Penetapan Binding Site, Grid Center, dan Search Space 314
5.4.1 Spesifikasi Fungsional dan Scientific Guardrails 316
5.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 317
5.4.3 Sketsa Implementasi Python 318
5.5 Parameter AutoDock Vina, Exhaustiveness, dan Replikasi 320
5.5.1 Spesifikasi Fungsional dan Scientific Guardrails 322
5.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 324
5.5.3 Sketsa Implementasi Python 325
5.6 Pose Clustering dan Seleksi Konformasi Representatif 326
5.6.1 Spesifikasi Fungsional dan Scientific Guardrails 327
5.6.2 Risiko, Failure Mode, dan Strategi Mitigasi 329
5.6.3 Sketsa Implementasi Python 330
5.7 Ikatan Hidrogen dan Kontak Polar 331
5.7.1 Spesifikasi Fungsional dan Scientific Guardrails 333

5.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 334
5.7.3 Sketsa Implementasi Python 335
5.8 Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge 336
5.8.1 Spesifikasi Fungsional dan Scientific Guardrails 338
5.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 339
5.8.3 Sketsa Implementasi Python 340
5.9 Interaction Fingerprint dan Residue Frequency 342
5.9.1 Spesifikasi Fungsional dan Scientific Guardrails 344
5.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 345
5.9.3 Sketsa Implementasi Python 346
5.10 Redocking, RMSD, dan Validasi Protokol 347
5.10.1 Spesifikasi Fungsional dan Scientific Guardrails 349
5.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 350
5.10.3 Sketsa Implementasi Python 351
5.11 Ranking Multikriteria setelah Docking 352
5.11.1 Spesifikasi Fungsional dan Scientific Guardrails 354
5.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 356
5.11.3 Sketsa Implementasi Python 357
5.12 Integrasi DockFlow dengan QM, MD, dan QSAR 358
5.12.1 Spesifikasi Fungsional dan Scientific Guardrails 359
5.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 361
5.12.3 Sketsa Implementasi Python 362
5.13 Rangkuman Bab 363
BAB 6 MD KOMPUTASI: MOLECULAR DYNAMICS BERBASIS OPENMM 365
6.1 Posisi Molecular Dynamics setelah Docking 366
6.1.1 Spesifikasi Fungsional dan Scientific Guardrails 368
6.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 370
6.1.3 Sketsa Implementasi Python 370
6.2 Pemilihan Force Field dan Konsistensi Parameter 372

6.2.1 Spesifikasi Fungsional dan Scientific Guardrails 373
6.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 375
6.2.3 Sketsa Implementasi Python 376
6.3 Parameterisasi Ligan dan Pemeriksaan Muatan 377
6.3.1 Spesifikasi Fungsional dan Scientific Guardrails 379
6.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 380
6.3.3 Sketsa Implementasi Python 381
6.4 Solvation Box, Model Air, Ion, dan Periodic Boundary 382
6.4.1 Spesifikasi Fungsional dan Scientific Guardrails 384
6.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 386
6.4.3 Sketsa Implementasi Python 386
6.5 Energy Minimization dan Pemeriksaan Geometri Awal 388
6.5.1 Spesifikasi Fungsional dan Scientific Guardrails 390
6.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 392
6.5.3 Sketsa Implementasi Python 393
6.6 Equilibration NVT: Suhu dan Stabilitas Sistem 394
6.6.1 Spesifikasi Fungsional dan Scientific Guardrails 395
6.6.2 Risiko, Failure Mode, dan Strategi Mitigasi 397
6.6.3 Sketsa Implementasi Python 398
6.7 Equilibration NPT: Tekanan, Densitas, dan Volume 399
6.7.1 Spesifikasi Fungsional dan Scientific Guardrails 401
6.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 402
6.7.3 Sketsa Implementasi Python 403
6.8 Production MD, Time Step, Constraints, dan Output 404
6.8.1 Spesifikasi Fungsional dan Scientific Guardrails 406
6.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 408
6.8.3 Sketsa Implementasi Python 409
6.9 Checkpoint, Resume, dan Ketahanan terhadap Interupsi 410
6.9.1 Spesifikasi Fungsional dan Scientific Guardrails 412

6.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 414
6.9.3 Sketsa Implementasi Python 414
6.10 RMSD, RMSF, Radius of Gyration, dan SASA 416
6.10.1 Spesifikasi Fungsional dan Scientific Guardrails 417
6.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 419
6.10.3 Sketsa Implementasi Python 420
6.11 H-Bond, Contact Occupancy, Distance, dan Analisis Ligan 421
6.11.1 Spesifikasi Fungsional dan Scientific Guardrails 423
6.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 424
6.11.3 Sketsa Implementasi Python 425
6.12 Replikasi, Konvergensi, dan Pelaporan Simulasi MD 426
6.12.1 Spesifikasi Fungsional dan Scientific Guardrails 428
6.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 430
6.12.3 Sketsa Implementasi Python 430
6.13 Rangkuman Bab 432
BAB 7 MOLPREP: CONFORMER DAN MOLECULAR PREPARATION STUDIO 433
7.1 Mengapa Geometri Awal Menentukan Mutu Hasil 434
7.1.1 Spesifikasi Fungsional dan Scientific Guardrails 436
7.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 438
7.1.3 Sketsa Implementasi Python 438
7.2 Parsing SMILES, InChI, SDF, MOL, dan XYZ 440
7.2.1 Spesifikasi Fungsional dan Scientific Guardrails 441
7.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 443
7.2.3 Sketsa Implementasi Python 444
7.3 Sanitization, Valence Check, dan Kekonsistenan Struktur 445
7.3.1 Spesifikasi Fungsional dan Scientific Guardrails 447
7.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 448
7.3.3 Sketsa Implementasi Python 449
7.4 Salt Removal, Fragment Selection, dan Standardization 450

7.4.1 Spesifikasi Fungsional dan Scientific Guardrails 452
7.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 454
7.4.3 Sketsa Implementasi Python 454
7.5 Tautomer Generation dan Konsekuensi terhadap Energi 456
7.5.1 Spesifikasi Fungsional dan Scientific Guardrails 458
7.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 460
7.5.3 Sketsa Implementasi Python 461
7.6 Protonation State, Charge, dan Kondisi Kimia 462
7.6.1 Spesifikasi Fungsional dan Scientific Guardrails 463
7.6.2 Risiko, Failure Mode, dan Strategi Mitigasi 465
7.6.3 Sketsa Implementasi Python 466
7.7 Stereoisomer Enumeration dan Integritas Stereokimia 467
7.7.1 Spesifikasi Fungsional dan Scientific Guardrails 469
7.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 470
7.7.3 Sketsa Implementasi Python 471
7.8 Conformer Generation dan Sampling Ruang Konformasi 472
7.8.1 Spesifikasi Fungsional dan Scientific Guardrails 474
7.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 476
7.8.3 Sketsa Implementasi Python 477
7.9 UFF/MMFF Relaxation dan Energy Filtering 478
7.9.1 Spesifikasi Fungsional dan Scientific Guardrails 480
7.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 482
7.9.3 Sketsa Implementasi Python 482
7.10 RMSD Clustering dan Penghapusan Duplikat 484
7.10.1 Spesifikasi Fungsional dan Scientific Guardrails 485
7.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 487
7.10.3 Sketsa Implementasi Python 488
7.11 xTB Refinement dan Seleksi Top Conformer 489
7.11.1 Spesifikasi Fungsional dan Scientific Guardrails 491

7.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 492
7.11.3 Sketsa Implementasi Python 493
7.12 Integrasi MolPrep dengan QMScreen dan DockFlow 494
7.12.1 Spesifikasi Fungsional dan Scientific Guardrails 496
7.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 498
7.12.3 Sketsa Implementasi Python 498
7.13 Rangkuman Bab 500
BAB 8 QSAR/ADMET STUDIO: DESKRIPTOR, MACHINE LEARNING, DAN SCREENING 501
8.1 Definisi Masalah QSAR dan Kualitas Endpoint 502
8.1.1 Spesifikasi Fungsional dan Scientific Guardrails 504
8.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 506
8.1.3 Sketsa Implementasi Python 506
8.2 Kurasi Dataset, Duplikat, Satuan, dan Missing Value 508
8.2.1 Spesifikasi Fungsional dan Scientific Guardrails 509
8.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 511
8.2.3 Sketsa Implementasi Python 512
8.3 Descriptor Fisikokimia RDKit dan Fingerprints 513
8.3.1 Spesifikasi Fungsional dan Scientific Guardrails 515
8.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 516
8.3.3 Sketsa Implementasi Python 517
8.4 Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge 518
8.4.1 Spesifikasi Fungsional dan Scientific Guardrails 520
8.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 522
8.4.3 Sketsa Implementasi Python 522
8.5 Descriptor Docking dan Interaction Fingerprint 524
8.5.1 Spesifikasi Fungsional dan Scientific Guardrails 526
8.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 528
8.5.3 Sketsa Implementasi Python 528
8.6 Feature Cleaning, Scaling, dan Multicollinearity 530

8.6.1 Spesifikasi Fungsional dan Scientific Guardrails 531
8.6.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 533
8.6.3 Sketsa Implementasi Python..... 534
8.7 Feature Selection dan Pencegahan Data Leakage 535
8.7.1 Spesifikasi Fungsional dan Scientific Guardrails..... 537
8.7.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 538
8.7.3 Sketsa Implementasi Python..... 539
8.8 Pembagian Train, Validation, Test dan Cross-Validation.....540
8.8.1 Spesifikasi Fungsional dan Scientific Guardrails..... 542
8.8.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 544
8.8.3 Sketsa Implementasi Python..... 544
8.9 Linear/PLS, Random Forest, Gradient Boosting, dan SVM 546
8.9.1 Spesifikasi Fungsional dan Scientific Guardrails..... 548
8.9.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 549
8.9.3 Sketsa Implementasi Python..... 550
8.10 Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics.....551
8.10.1 Spesifikasi Fungsional dan Scientific Guardrails..... 553
8.10.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 555
8.10.3 Sketsa Implementasi Python..... 556
8.11 Y-Randomization dan Applicability Domain.....557
8.11.1 Spesifikasi Fungsional dan Scientific Guardrails..... 558
8.11.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 560
8.11.3 Sketsa Implementasi Python..... 561
8.12 Explainability, Prediction, dan Ekspor Model 562
8.12.1 Spesifikasi Fungsional dan Scientific Guardrails..... 564
8.12.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 565
8.12.3 Sketsa Implementasi Python..... 566
8.13 Rangkuman Bab 568
BAB 9 REPROCHEM: DATA, PROVENANCE, DAN REPRODUCIBLE COMPUTATIONAL RESEARCH 569

9.1 Reproducibility sebagai Fitur Inti Software Ilmiah.....570
9.1.1 Spesifikasi Fungsional dan Scientific Guardrails..... 572
9.1.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 574
9.1.3 Sketsa Implementasi Python..... 574
9.2 Project ID, Molecule ID, Calculation ID, dan Relasi Data 576
9.2.1 Spesifikasi Fungsional dan Scientific Guardrails..... 577
9.2.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 579
9.2.3 Sketsa Implementasi Python..... 580
9.3 Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity 581
9.3.1 Spesifikasi Fungsional dan Scientific Guardrails..... 583
9.3.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 584
9.3.3 Sketsa Implementasi Python..... 585
9.4 Calculation Specification: Method, Basis, Solvent, dan Convergence 586
9.4.1 Spesifikasi Fungsional dan Scientific Guardrails..... 588
9.4.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 590
9.4.3 Sketsa Implementasi Python..... 590
9.5 Environment Snapshot: Versi Software, CPU, GPU, dan OS.....592
9.5.1 Spesifikasi Fungsional dan Scientific Guardrails..... 594
9.5.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 596
9.5.3 Sketsa Implementasi Python..... 597
9.6 Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan598
9.6.1 Spesifikasi Fungsional dan Scientific Guardrails..... 599
9.6.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 601
9.6.3 Sketsa Implementasi Python..... 602
9.7 Hash, Checksum, Deduplication, dan Integritas File 603
9.7.1 Spesifikasi Fungsional dan Scientific Guardrails..... 605
9.7.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 606
9.7.3 Sketsa Implementasi Python..... 607
9.8 Database Schema, Indexing, dan Query Hasil 608

9.8.1 Spesifikasi Fungsional dan Scientific Guardrails 610
9.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 612
9.8.3 Sketsa Implementasi Python 612
9.9 Status Job, Error Log, Restart, dan Audit Trail 614
9.9.1 Spesifikasi Fungsional dan Scientific Guardrails 616
9.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 617
9.9.3 Sketsa Implementasi Python 618
9.10 FAIR Data, Metadata, dan Ekspor Proyek 619
9.10.1 Spesifikasi Fungsional dan Scientific Guardrails 621
9.10.2 Risiko, Failure Mode, dan Strategi Mitigasi 623
9.10.3 Sketsa Implementasi Python 623
9.11 Generator Laporan DOCX/PDF/CSV yang Reproducible 625
9.11.1 Spesifikasi Fungsional dan Scientific Guardrails 626
9.11.2 Risiko, Failure Mode, dan Strategi Mitigasi 628
9.11.3 Sketsa Implementasi Python 629
9.12 Backup, Migrasi Versi, dan Retensi Data Riset 630
9.12.1 Spesifikasi Fungsional dan Scientific Guardrails 632
9.12.2 Risiko, Failure Mode, dan Strategi Mitigasi 633
9.12.3 Sketsa Implementasi Python 634
9.13 Rangkuman Bab 636
BAB 10 INTEGRASI KOMPUTASI JISAMSU, IMPLEMENTASI, VALIDASI, DAN PETA JALAN 637
10.1 Satu Core Engine dan Banyak Modul Antarmuka 638
10.1.1 Spesifikasi Fungsional dan Scientific Guardrails 640
10.1.2 Risiko, Failure Mode, dan Strategi Mitigasi 642
10.1.3 Sketsa Implementasi Python 642
10.2 Kontrak Data Antar-Modul dan QCSchema-Inspired Interface 644
10.2.1 Spesifikasi Fungsional dan Scientific Guardrails 645
10.2.2 Risiko, Failure Mode, dan Strategi Mitigasi 647
10.2.3 Sketsa Implementasi Python 648

10.3 Struktur Package Python dan Dependency Boundary 649
10.3.1 Spesifikasi Fungsional dan Scientific Guardrails 651
10.3.2 Risiko, Failure Mode, dan Strategi Mitigasi 652
10.3.3 Sketsa Implementasi Python 653
10.4 Streamlit GUI, CLI, dan Service Layer 654
10.4.1 Spesifikasi Fungsional dan Scientific Guardrails 656
10.4.2 Risiko, Failure Mode, dan Strategi Mitigasi 658
10.4.3 Sketsa Implementasi Python 658
10.5 Workspace Pengguna, Permission Linux, dan Keamanan File 660
10.5.1 Spesifikasi Fungsional dan Scientific Guardrails 662
10.5.2 Risiko, Failure Mode, dan Strategi Mitigasi 664
10.5.3 Sketsa Implementasi Python 665
10.6 Packaging Conda/Pip dan Paket Debian 666
10.6.1 Spesifikasi Fungsional dan Scientific Guardrails 667
10.6.2 Risiko, Failure Mode, dan Strategi Mitigasi 669
10.6.3 Sketsa Implementasi Python 670
10.7 Logging, Telemetri Lokal, dan Diagnostik Error 671
10.7.1 Spesifikasi Fungsional dan Scientific Guardrails 673
10.7.2 Risiko, Failure Mode, dan Strategi Mitigasi 674
10.7.3 Sketsa Implementasi Python 675
10.8 Unit Test, Integration Test, Scientific Benchmark, dan Regression Test 676
10.8.1 Spesifikasi Fungsional dan Scientific Guardrails 678
10.8.2 Risiko, Failure Mode, dan Strategi Mitigasi 680
10.8.3 Sketsa Implementasi Python 681
10.9 Dokumentasi Pengguna, Tutorial, dan Contoh Proyek 682
10.9.1 Spesifikasi Fungsional dan Scientific Guardrails 684
10.9.2 Risiko, Failure Mode, dan Strategi Mitigasi 686
10.9.3 Sketsa Implementasi Python 686
10.10 Versioning, Release Notes, dan Migrasi Database 688

10.10.1 Spesifikasi Fungsional dan Scientific Guardrails 689
10.10.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 691
10.10.3 Sketsa Implementasi Python..... 692
10.11 Prioritas Implementasi Bertahap 2026 dan Sesudahnya 693
10.11.1 Spesifikasi Fungsional dan Scientific Guardrails 695
10.11.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 696
10.11.3 Sketsa Implementasi Python..... 697
10.12 Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan 698
10.12.1 Spesifikasi Fungsional dan Scientific Guardrails 700
10.12.2 Risiko, Failure Mode, dan Strategi Mitigasi..... 702
10.12.3 Sketsa Implementasi Python..... 702
10.13 Rangkuman Bab 704
10.14 LATIHAN INTEGRATIF DAN PEMBAHASAN 705
10.15 SOAL PENGEMBANGAN DAN PROYEK 708
GLOSARIUM 710
DAFTAR PUSTAKA 717

BAB 1 LANDASAN, PRIORITAS, DAN ARSITEKTUR SOFTWARE KIMIA KOMPUTASI

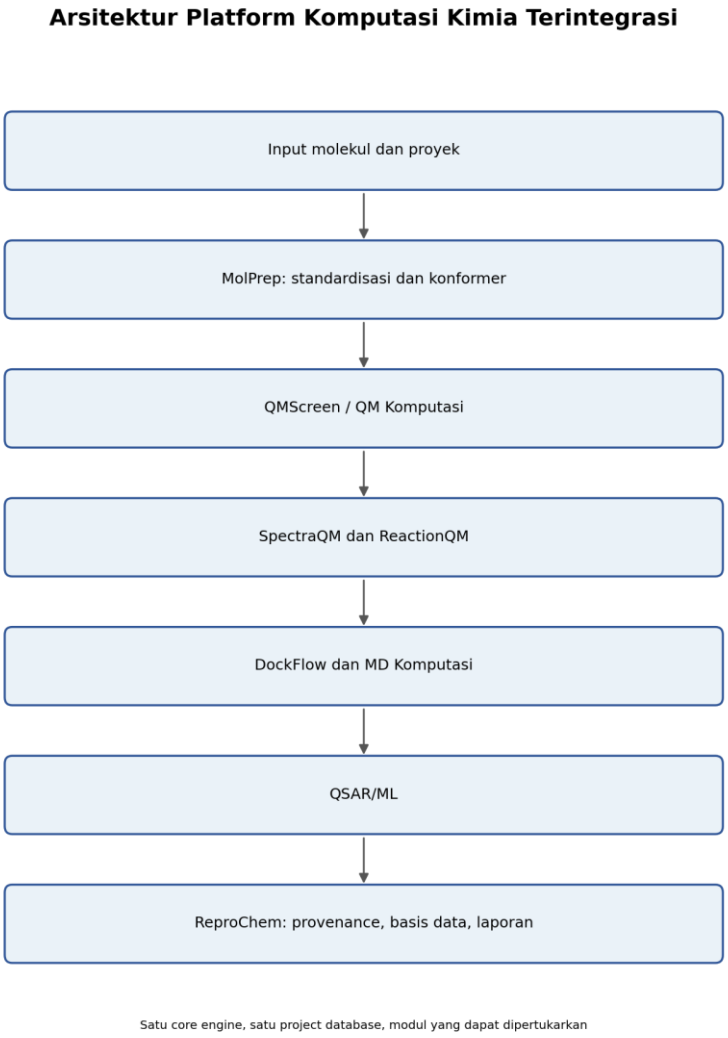
Bab ini berfokus pada menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Ekosistem teknis yang menjadi perhatian utama meliputi Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.

1.1 Perubahan Paradigma dari Kalkulator Energi ke Platform Riset

Perubahan Paradigma dari Kalkulator Energi ke Platform Riset harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.



Gambar 1.1. Landasan, Prioritas, Dan Arsitektur Software Kimia Komputasi

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Perubahan Paradigma dari Kalkulator Energi ke Platform Riset, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan

mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.1, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa

keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.1. Spesifikasi fungsional Perubahan Paradigma dari Kalkulator Energi ke Platform Riset

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.2. Matriks risiko Perubahan Paradigma dari Kalkulator Energi ke Platform Riset

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_1_perubahan_paradigma_dari_kalkula(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.2 Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility

Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.2, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.3. Spesifikasi fungsional Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.4. Matriks risiko Bottleneck Riset: Preparasi, Throughput, Analisis, dan Reproducibility

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_2_bottleneck_riset_preparasi_throu(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.3 Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI

Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini,

pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.3, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.5. Spesifikasi fungsional Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.6. Matriks risiko Prinsip Modularitas dan Pemisahan Scientific Engine dari GUI

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_3_prinsip_modularitas_dan_pemisaha(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.

- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.4 Ekosistem Python dan Linux untuk Komputasi Kimia

Ekosistem Python dan Linux untuk Komputasi Kimia harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Ekosistem Python dan Linux untuk Komputasi Kimia, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.4, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa

keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.7. Spesifikasi fungsional Ekosistem Python dan Linux untuk Komputasi Kimia

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.8. Matriks risiko Ekosistem Python dan Linux untuk Komputasi Kimia

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.4.3 Sketsa Implementasi Python

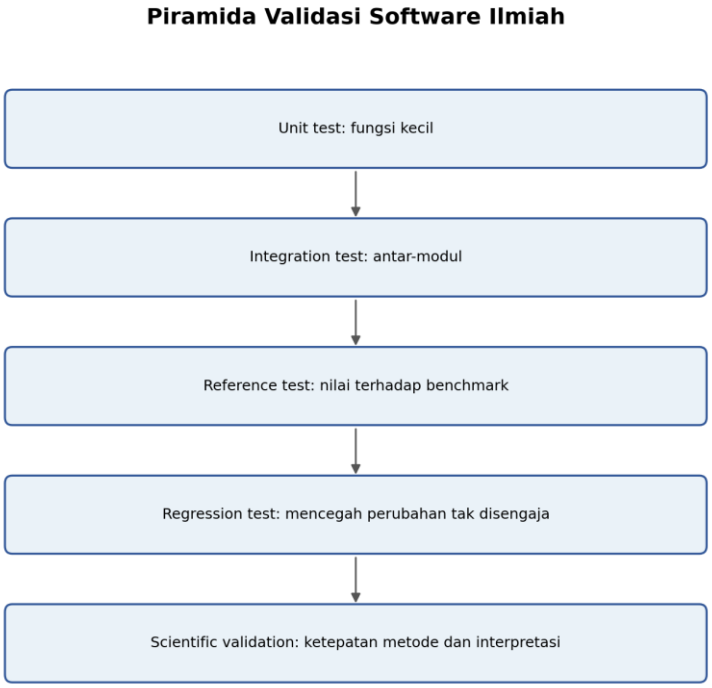
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_4_ekosistem_python_dan_linux_untuk(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 1.2. Kerangka pendukung untuk Ekosistem Python dan Linux untuk Komputasi Kimia

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.5 Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF

Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.5, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.9. Spesifikasi fungsional Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.10. Matriks risiko Tingkatan Metode: Force Field, xTB, HF, DFT, dan Post-HF

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_5_tingkatan_metode_force_field_xtb(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.6 Strategi Multi-Fidelity untuk Efisiensi Komputasi

Strategi Multi-Fidelity untuk Efisiensi Komputasi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Strategi Multi-Fidelity untuk Efisiensi Komputasi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.6, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.11. Spesifikasi fungsional Strategi Multi-Fidelity untuk Efisiensi Komputasi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.12. Matriks risiko Strategi Multi-Fidelity untuk Efisiensi Komputasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_6_strategi_multi_fidelity_untuk_ef(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.7 Model Data Molekul, Job, Result, dan Project

Model Data Molekul, Job, Result, dan Project harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan

hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Model Data Molekul, Job, Result, dan Project, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.7, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.13. Spesifikasi fungsional Model Data Molekul, Job, Result, dan Project

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.14. Matriks risiko Model Data Molekul, Job, Result, dan Project

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_7_model_data_molekul_job_result_da(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.8 Charge, Multiplicity, Unit, dan Validasi Input

Charge, Multiplicity, Unit, dan Validasi Input harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan format data, validasi sintaks, unit, charge, multiplicity, stereokimia, provenance. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Charge, Multiplicity, Unit, dan Validasi Input, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.8, aturan kontrol yang disarankan adalah: format data harus memiliki nilai/keadaan yang eksplisit; validasi sintaks harus memiliki nilai/keadaan yang eksplisit; unit harus memiliki nilai/keadaan yang eksplisit; charge harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar format data, validasi sintaks, unit, charge, multiplicity, stereokimia. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.15. Spesifikasi fungsional Charge, Multiplicity, Unit, dan Validasi Input

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi format data dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk format data; simpan status, pesan, dan provenance.	Record format data, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi validasi sintaks dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk validasi sintaks; simpan status, pesan, dan provenance.	Record validasi sintaks, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi unit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk unit; simpan status, pesan, dan provenance.	Record unit, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan charge dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk charge; simpan status, pesan, dan provenance.	Record charge, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis multiplicity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk multiplicity; simpan status, pesan, dan provenance.	Record multiplicity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi stereokimia dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereokimia; simpan status, pesan, dan provenance.	Record stereokimia, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.16. Matriks risiko Charge, Multiplicity, Unit, dan Validasi Input

Risiko	Penyebab	Mitigasi
Input tidak konsisten	format data salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	validasi sintaks tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	unit dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	charge dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau multiplicity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	stereokimia menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser provenance berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.8.3 Sketsa Implementasi Python

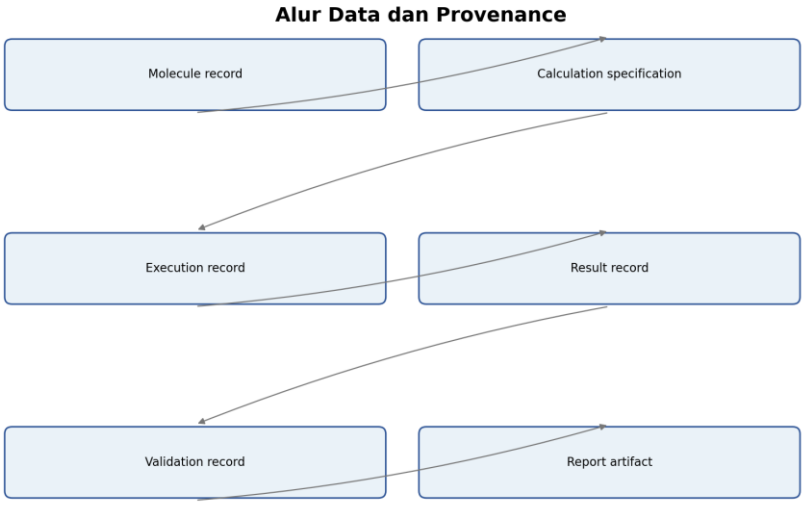
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_8_charge_multiplicity_unit_dan_val(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="format data")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="validasi sintaks")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk format data dan validasi sintaks perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 1.3. Kerangka pendukung untuk Charge, Multiplicity, Unit, dan Validasi Input

Checklist implementasi:

- Input format data memiliki validator dan pesan error yang jelas.
- Parameter validasi sintaks tersimpan dalam job specification.
- Raw output untuk unit disimpan sebelum parsing.
- Hasil charge mempunyai unit dan provenance.
- Kegagalan multiplicity dapat dilacak melalui log.
- Perubahan stereokimia dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.9 Manajemen CPU, Memori, GPU, dan Parallelism

Manajemen CPU, Memori, GPU, dan Parallelism harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan worker, job queue, CPU thread, memory limit, retry policy, checkpoint, resource monitor. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Manajemen CPU, Memori, GPU, dan Parallelism, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.9, aturan kontrol yang disarankan adalah: worker harus memiliki nilai/keadaan yang eksplisit; job queue harus memiliki

nilai/keadaan yang eksplisit; CPU thread harus memiliki nilai/keadaan yang eksplisit; memory limit harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar worker, job queue, CPU thread, memory limit, retry policy, checkpoint. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.17. Spesifikasi fungsional Manajemen CPU, Memori, GPU, dan Parallelism

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi worker dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk worker; simpan status, pesan, dan provenance.	Record worker, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi job queue dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk job queue; simpan status, pesan, dan provenance.	Record job queue, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi CPU thread dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk CPU thread; simpan status, pesan, dan provenance.	Record CPU thread, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan memory limit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk memory limit; simpan status, pesan, dan provenance.	Record memory limit, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis retry policy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk retry policy; simpan status, pesan, dan provenance.	Record retry policy, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor resource monitor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk resource monitor; simpan status, pesan, dan provenance.	Record resource monitor, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.18. Matriks risiko Manajemen CPU, Memori, GPU, dan Parallelism

Risiko	Penyebab	Mitigasi
Input tidak konsisten	worker salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	job queue tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	CPU thread dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	memory limit dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau retry policy tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	checkpoint menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser resource monitor berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_9_manajemen_cpu_memori_gpu_dan_par(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="worker")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="job queue")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk worker dan job queue perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input worker memiliki validator dan pesan error yang jelas.
- Parameter job queue tersimpan dalam job specification.
- Raw output untuk CPU thread disimpan sebelum parsing.
- Hasil memory limit mempunyai unit dan provenance.
- Kegagalan retry policy dapat dilacak melalui log.
- Perubahan checkpoint dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.10 Error Handling, Resume, Checkpoint, dan Failure Recovery

Error Handling, Resume, Checkpoint, dan Failure Recovery harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Error Handling, Resume, Checkpoint, dan Failure Recovery, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.10, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.19. Spesifikasi fungsional Error Handling, Resume, Checkpoint, dan Failure Recovery

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.20. Matriks risiko Error Handling, Resume, Checkpoint, dan Failure Recovery

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_10_error_handling_resume_checkpoin(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.11 Validasi Ilmiah, Benchmark, dan Regression Test

Validasi Ilmiah, Benchmark, dan Regression Test harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Validasi Ilmiah, Benchmark, dan Regression Test, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.11, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.21. Spesifikasi fungsional Validasi Ilmiah, Benchmark, dan Regression Test

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.22. Matriks risiko Validasi Ilmiah, Benchmark, dan Regression Test

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik

Risiko	Penyebab	Mitigasi
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
```

```
    multiplicity: int = 1
    method: str = "auto"

def run_1_11_validasi_ilmiah_benchmark_dan_r(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.12 Peta Prioritas Sembilan Modul Komputasi Jisamsu

Peta Prioritas Sembilan Modul Komputasi Jisamsu harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menetapkan kebutuhan ilmiah, prinsip desain, dan arsitektur platform Python/Linux yang menyelesaikan bottleneck nyata dalam riset kimia komputasi. Pada tingkat implementasi, modul memanfaatkan Python 3, Linux/WSL, PySCF, RDKit, xTB, OpenMM, NumPy, SciPy, Pandas, Matplotlib, SQLite dan

Streamlit. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Peta Prioritas Sembilan Modul Komputasi Jisamsu, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 1.12, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

1.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 1.23. Spesifikasi fungsional Peta Prioritas Sembilan Modul Komputasi Jisamsu

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

1.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 1.24. Matriks risiko Peta Prioritas Sembilan Modul Komputasi Jisamsu

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

1.12.3 Sketsa Implementasi Python

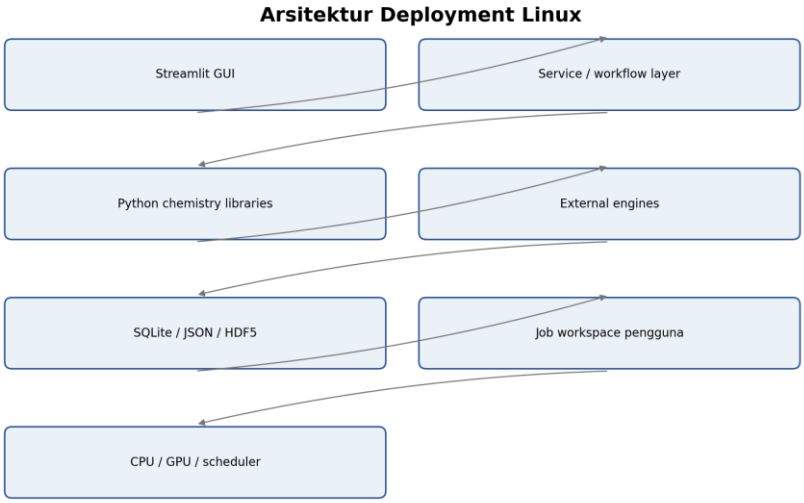
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section1Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_1_12_peta_prioritas_sembilan_modul_k(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 1.4. Kerangka pendukung untuk Peta Prioritas Sembilan Modul Komputasi Jisamsu

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

1.13 Rangkuman Bab

Bab 1 menempatkan LANDASAN, PRIORITAS, DAN ARSITEKTUR SOFTWARE KIMIA KOMPUTASI sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

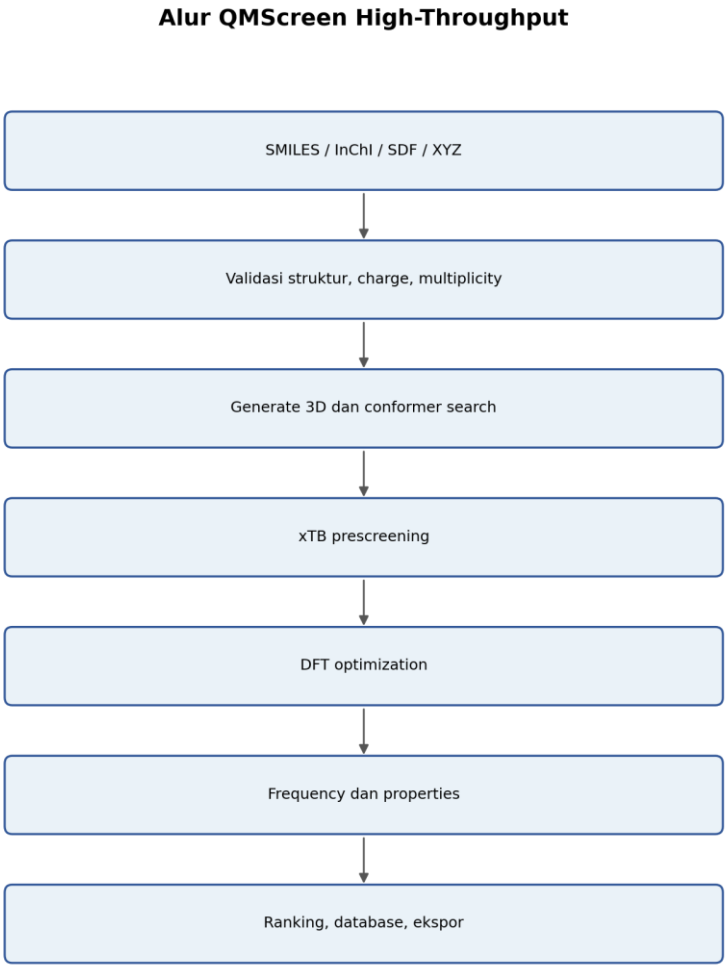
Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama,

GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 2 QMSCREEN: HIGH-THROUGHPUT QUANTUM CHEMISTRY DAN WORKFLOW MANAGER

Bab ini berfokus pada mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Ekosistem teknis yang menjadi perhatian utama meliputi RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 2.1. Qmscreen: High-Throughput Quantum Chemistry Dan Workflow Manager

2.1 Posisi QMScreen sebagai Perluasan QM Komputasi

Posisi QMScreen sebagai Perluasan QM Komputasi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Posisi QMScreen sebagai Perluasan QM Komputasi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.1, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki

nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.1. Spesifikasi fungsional Posisi QMScreen sebagai Perluasan QM Komputasi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.2. Matriks risiko Posisi QMScreen sebagai Perluasan QM Komputasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_1_posisi_qmscreen_sebagai_perluasa(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.2 Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV

Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan format data, validasi sintaks, unit, charge, multiplicity, stereokimia, provenance. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.2, aturan kontrol yang disarankan adalah: format data harus memiliki nilai/keadaan yang eksplisit; validasi sintaks harus memiliki nilai/keadaan yang eksplisit; unit harus memiliki nilai/keadaan yang eksplisit; charge harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar format data, validasi sintaks, unit, charge, multiplicity, stereokimia. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.3. Spesifikasi fungsional Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi format data dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk format data; simpan status, pesan, dan provenance.	Record format data, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi validasi sintaks dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk validasi sintaks; simpan status, pesan, dan provenance.	Record validasi sintaks, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi unit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk unit; simpan status, pesan, dan provenance.	Record unit, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan charge dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk charge; simpan status, pesan, dan provenance.	Record charge, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis multiplicity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk multiplicity; simpan status, pesan, dan provenance.	Record multiplicity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi stereokimia dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereokimia; simpan status, pesan, dan provenance.	Record stereokimia, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.4. Matriks risiko Desain Input Massal SMILES, InChI, SDF, XYZ, dan CSV

Risiko	Penyebab	Mitigasi
Input tidak konsisten	format data salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	validasi sintaks tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	unit dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	charge dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau multiplicity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	stereokimia menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser provenance berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_2_desain_input_massal_smiles_inchi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="format data")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="validasi sintaks")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk format data dan validasi sintaks perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input format data memiliki validator dan pesan error yang jelas.
- Parameter validasi sintaks tersimpan dalam job specification.
- Raw output untuk unit disimpan sebelum parsing.
- Hasil charge mempunyai unit dan provenance.
- Kegagalan multiplicity dapat dilacak melalui log.
- Perubahan stereokimia dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.3 Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan

Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan

pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.3, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine.

Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.5. Spesifikasi fungsional Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan format data dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk format data; simpan status, pesan, dan provenance.	Record format data, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark validasi sintaks dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk validasi sintaks; simpan status, pesan, dan provenance.	Record validasi sintaks, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi unit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk unit; simpan status, pesan, dan provenance.	Record unit, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.6. Matriks risiko Validasi Struktur dan Normalisasi Molekul sebelum Perhitungan

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_3_validasi_struktur_dan_normalisas(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.4 Generate 3D dan Conformer Generation untuk Batch Besar

Generate 3D dan Conformer Generation untuk Batch Besar harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Generate 3D dan Conformer Generation untuk Batch Besar, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.4, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki

nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.7. Spesifikasi fungsional Generate 3D dan Conformer Generation untuk Batch Besar

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.8. Matriks risiko Generate 3D dan Conformer Generation untuk Batch Besar

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.4.3 Sketsa Implementasi Python

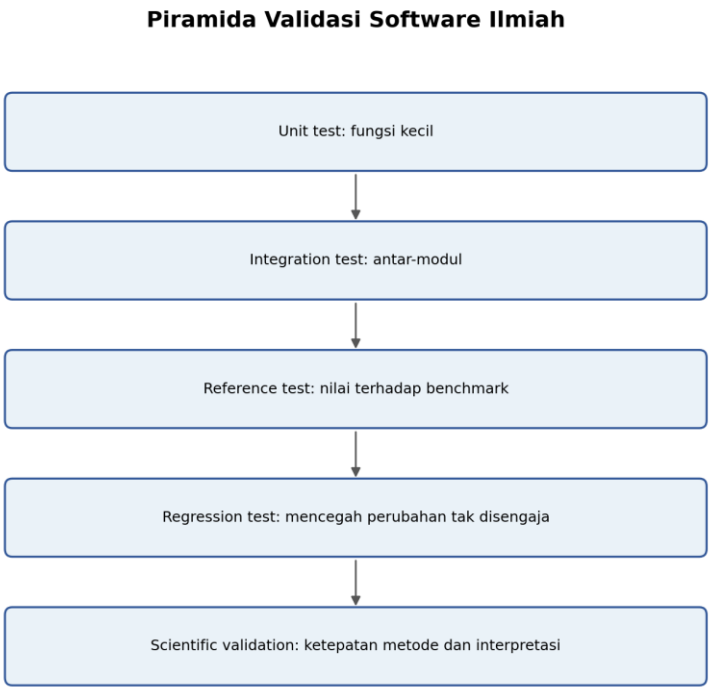
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_4_generate_3d_dan_conformer_genera(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 2.2. Kerangka pendukung untuk Generate 3D dan Conformer Generation untuk Batch Besar

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.5 Prescreening GFN2-xTB dan Seleksi Kandidat

Prescreening GFN2-xTB dan Seleksi Kandidat harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF,

geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Prescreening GFN2-xTB dan Seleksi Kandidat, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.5, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.9. Spesifikasi fungsional Prescreening GFN2-xTB dan Seleksi Kandidat

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.10. Matriks risiko Prescreening GFN2-xTB dan Seleksi Kandidat

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_5_prescreening_gfn2_xtb_dan_seleks(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.

- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.6 Optimasi Geometri DFT dan Strategi Basis Set Bertingkat

Optimasi Geometri DFT dan Strategi Basis Set Bertingkat harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Optimasi Geometri DFT dan Strategi Basis Set Bertingkat, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan

mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.6, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa

keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.11. Spesifikasi fungsional Optimasi Geometri DFT dan Strategi Basis Set Bertingkat

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.12. Matriks risiko Optimasi Geometri DFT dan Strategi Basis Set Bertingkat

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_6_optimasi_geometri_dft_dan_strate(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.7 Frequency, Thermochemistry, dan Pemeriksaan Minimum

Frequency, Thermochemistry, dan Pemeriksaan Minimum harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Frequency, Thermochemistry, dan Pemeriksaan Minimum, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.7, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.13. Spesifikasi fungsional Frequency, Thermochemistry, dan Pemeriksaan Minimum

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.14. Matriks risiko Frequency, Thermochemistry, dan Pemeriksaan Minimum

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik

Risiko	Penyebab	Mitigasi
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
```

```
    multiplicity: int = 1
    method: str = "auto"

def run_2_7_frequency_thermochemistry_dan_pe(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.8 HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum

HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF.

Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.8, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.15. Spesifikasi fungsional HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.16. Matriks risiko HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.8.3 Sketsa Implementasi Python

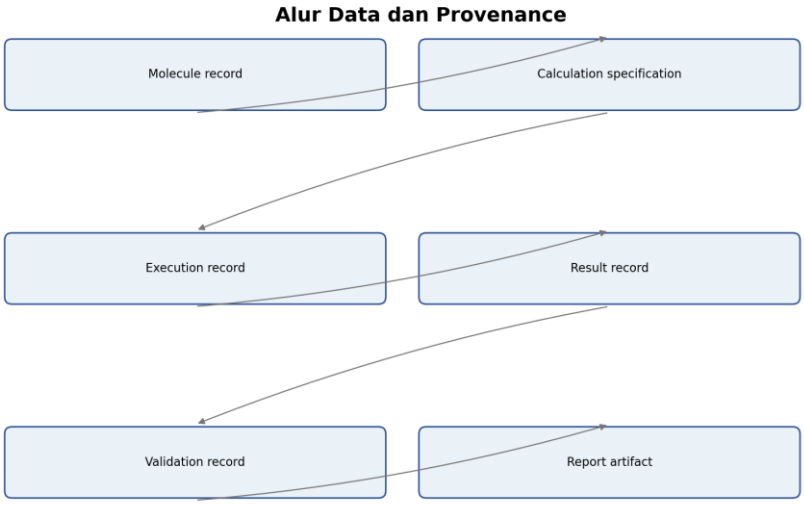
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_8_homo_lumo_dipole_charge_dan_desc(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 2.3. Kerangka pendukung untuk HOMO-LUMO, Dipole, Charge, dan Descriptor Kuantum

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.9 Scheduling Paralel, Resource Limit, dan Fair Queue

Scheduling Paralel, Resource Limit, dan Fair Queue harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Scheduling Paralel, Resource Limit, dan Fair Queue, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.9, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki

nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.17. Spesifikasi fungsional Scheduling Paralel, Resource Limit, dan Fair Queue

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan worker dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk worker; simpan status, pesan, dan provenance.	Record worker, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark job queue dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk job queue; simpan status, pesan, dan provenance.	Record job queue, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi CPU thread dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk CPU thread; simpan status, pesan, dan provenance.	Record CPU thread, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.18. Matriks risiko Scheduling Paralel, Resource Limit, dan Fair Queue

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_9_scheduling_paralel_resource_limi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.10 Resume Calculation, Deduplication, dan Cache Hasil

Resume Calculation, Deduplication, dan Cache Hasil harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Resume Calculation, Deduplication, dan Cache Hasil, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.10, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.19. Spesifikasi fungsional Resume Calculation, Deduplication, dan Cache Hasil

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.20. Matriks risiko Resume Calculation, Deduplication, dan Cache Hasil

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_10_resume_calculation_deduplicatio(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.11 Ranking Multikriteria dan Ekspor Data Riset

Ranking Multikriteria dan Ekspor Data Riset harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Ranking Multikriteria dan Ekspor Data Riset, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.11, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.21. Spesifikasi fungsional Ranking Multikriteria dan Ekspor Data Riset

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.22. Matriks risiko Ranking Multikriteria dan Ekspor Data Riset

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_11_ranking_multikriteria_dan_ekspo(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.12 Benchmark Kinerja dan Validasi Hasil High-Throughput

Benchmark Kinerja dan Validasi Hasil High-Throughput harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mengembangkan sistem high-throughput untuk memproses puluhan hingga ribuan kandidat secara bertahap dari struktur hingga ranking properti kuantum. Pada tingkat implementasi, modul memanfaatkan RDKit, xTB/GFN2-xTB, PySCF, geomeTRIC, concurrent.futures, job queue, SQLite dan format CSV/XLSX/PDF. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat

menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan method, basis set, SCF, geometry, frequency, orbital, thermochemistry. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Benchmark Kinerja dan Validasi Hasil High-Throughput, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 2.12, aturan kontrol yang disarankan adalah: method harus memiliki nilai/keadaan yang eksplisit; basis set harus memiliki nilai/keadaan yang eksplisit; SCF harus memiliki nilai/keadaan yang eksplisit; geometry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

2.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar method, basis set, SCF, geometry, frequency, orbital. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 2.23. Spesifikasi fungsional Benchmark Kinerja dan Validasi Hasil High-Throughput

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan geometry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk geometry; simpan status, pesan, dan provenance.	Record geometry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi orbital dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk orbital; simpan status, pesan, dan provenance.	Record orbital, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor thermochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk thermochemistry; simpan status, pesan, dan provenance.	Record thermochemistry, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

2.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 2.24. Matriks risiko Benchmark Kinerja dan Validasi Hasil High-Throughput

Risiko	Penyebab	Mitigasi
Input tidak konsisten	method salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	basis set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	SCF dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	geometry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau frequency tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	orbital menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser thermochemistry berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

2.12.3 Sketsa Implementasi Python

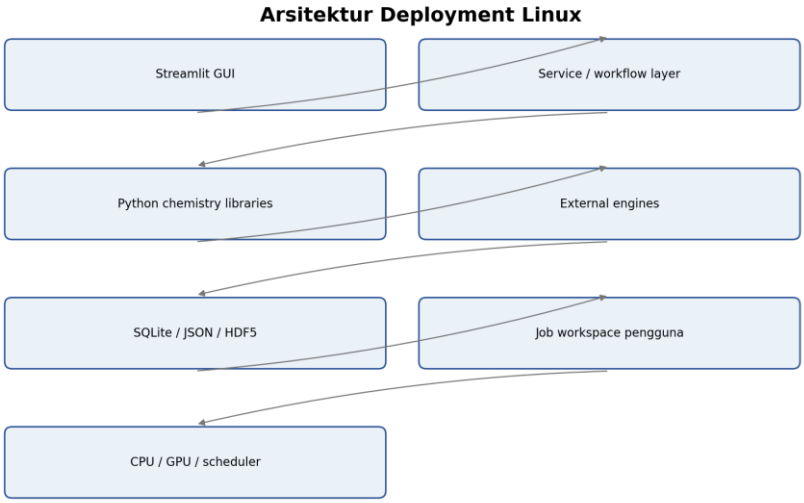
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section2Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_2_12_benchmark_kinerja_dan_validasi_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="method")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="basis set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk method dan basis set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 2.4. Kerangka pendukung untuk Benchmark Kinerja dan Validasi Hasil High-Throughput

Checklist implementasi:

- Input method memiliki validator dan pesan error yang jelas.
- Parameter basis set tersimpan dalam job specification.
- Raw output untuk SCF disimpan sebelum parsing.
- Hasil geometry mempunyai unit dan provenance.
- Kegagalan frequency dapat dilacak melalui log.
- Perubahan orbital dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

2.13 Rangkuman Bab

Bab 2 menempatkan QMSCREEN: HIGH-THROUGHPUT QUANTUM CHEMISTRY DAN WORKFLOW MANAGER sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

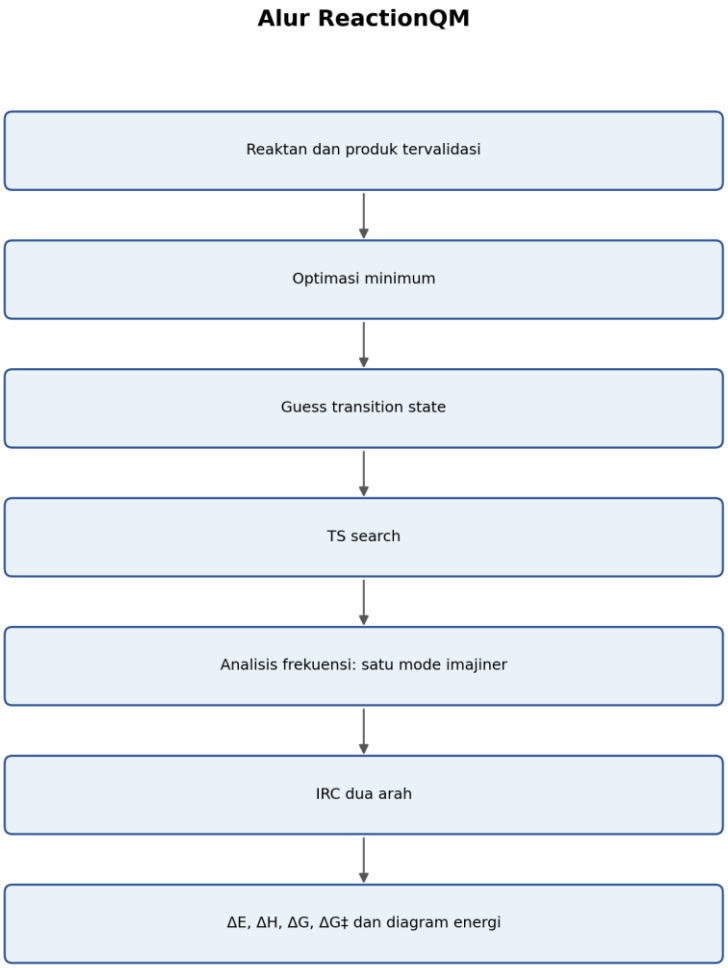
Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan

perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 3 REACTIONQM: JALUR REAKSI, TRANSITION STATE, DAN MEKANISME

Bab ini berfokus pada menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Ekosistem teknis yang menjadi perhatian utama meliputi PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 3.1. Reactionqm: Jalur Reaksi, Transition State, Dan Mekanisme

3.1 Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme

Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.1, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit;

Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.1. Spesifikasi fungsional Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.2. Matriks risiko Mengapa Energi Minimum Saja Tidak Cukup untuk Mekanisme

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_1_mengapa_energi_minimum_saja_tida(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.2 Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping

Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.2, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.3. Spesifikasi fungsional Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.4. Matriks risiko Representasi Reaktan, Produk, Stoikiometri, dan Atom Mapping

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_2_representasi_reaktan_produk_stoi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.3 Optimasi Minimum Reaktan dan Produk

Optimasi Minimum Reaktan dan Produk harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Optimasi Minimum Reaktan dan Produk, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.3, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.5. Spesifikasi fungsional Optimasi Minimum Reaktan dan Produk

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.6. Matriks risiko Optimasi Minimum Reaktan dan Produk

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_3_optimasi_minimum_reaktan_dan_pro(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.4 Penyusunan Guess Transition State yang Rasional

Penyusunan Guess Transition State yang Rasional harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi,

dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Penyusunan Guess Transition State yang Rasional, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.4, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.7. Spesifikasi fungsional Penyusunan Guess Transition State yang Rasional

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.8. Matriks risiko Penyusunan Guess Transition State yang Rasional

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.4.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

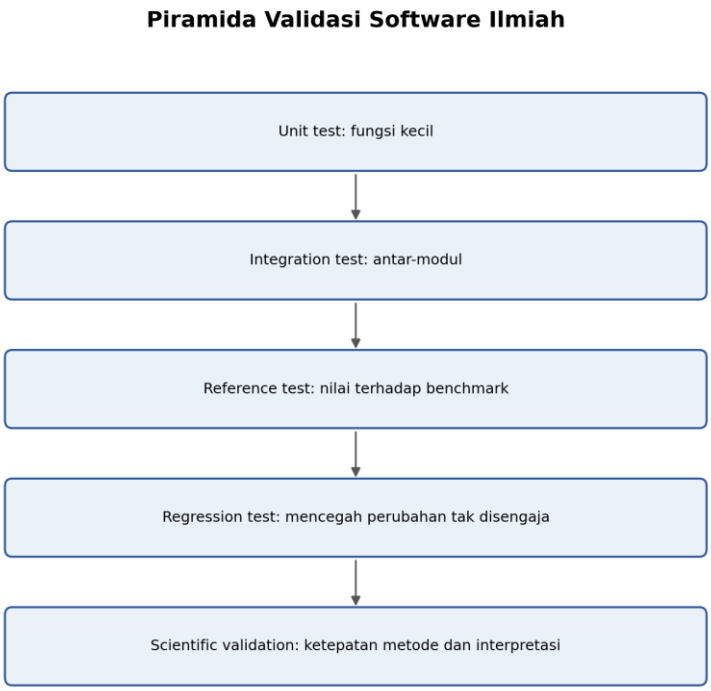
provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_4_penyusunan_guess_transition_stat(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 3.2. Kerangka pendukung untuk Penyusunan Guess Transition State yang Rasional

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.5 Algoritma Transition-State Search dan Hessian

Algoritma Transition-State Search dan Hessian harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC,

optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Algoritma Transition-State Search dan Hessian, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.5, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.9. Spesifikasi fungsional Algoritma Transition-State Search dan Hessian

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.10. Matriks risiko Algoritma Transition-State Search dan Hessian

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_5_algoritma_transition_state_searc(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.

- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.6 Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal

Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan

mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.6, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa

keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.11. Spesifikasi fungsional Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.12. Matriks risiko Kriteria Satu Frekuensi Imajiner dan Analisis Mode Normal

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_6_kriteria_satu_frekuensi_imajiner(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.7 Intrinsic Reaction Coordinate dan Verifikasi Konektivitas

Intrinsic Reaction Coordinate dan Verifikasi Konektivitas harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Intrinsic Reaction Coordinate dan Verifikasi Konektivitas, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.7, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.13. Spesifikasi fungsional Intrinsic Reaction Coordinate dan Verifikasi Konektivitas

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.14. Matriks risiko Intrinsic Reaction Coordinate dan Verifikasi Konektivitas

Risiko	Penyebab	Mitigasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
```

```
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_7_intrinsic_reaction_coordinate_da(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.8 Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs

Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari

minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.8, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.15. Spesifikasi fungsional Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.16. Matriks risiko Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.8.3 Sketsa Implementasi Python

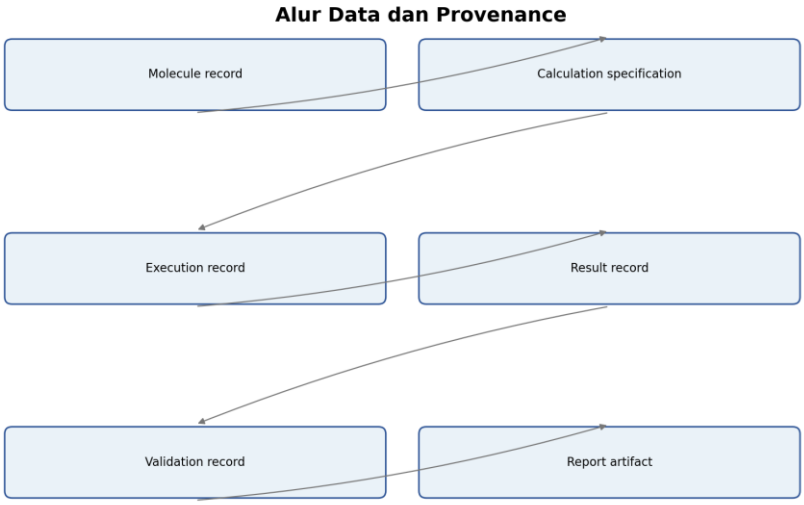
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_8_energi_reaksi_entalpi_entropi_da(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 3.3. Kerangka pendukung untuk Energi Reaksi, Entalpi, Entropi, dan Energi Bebas Gibbs

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.9 Energi Aktivasi dan Interpretasi Kinetika

Energi Aktivasi dan Interpretasi Kinetika harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Energi Aktivasi dan Interpretasi Kinetika, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.9, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit;

Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.17. Spesifikasi fungsional Energi Aktivasi dan Interpretasi Kinetika

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.18. Matriks risiko Energi Aktivasi dan Interpretasi Kinetika

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_9_energi_aktivasi_dan_interpretasi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.10 Reaksi Multitahap, Intermediate, dan Reaction Network

Reaksi Multitahap, Intermediate, dan Reaction Network harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Reaksi Multitahap, Intermediate, dan Reaction Network, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.10, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.19. Spesifikasi fungsional Reaksi Multitahap, Intermediate, dan Reaction Network

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.20. Matriks risiko Reaksi Multitahap, Intermediate, dan Reaction Network

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_10_reaksi_multitahap_intermediate_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.11 Visualisasi Diagram Energi dan Bond Change

Visualisasi Diagram Energi dan Bond Change harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Visualisasi Diagram Energi dan Bond Change, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.11, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.21. Spesifikasi fungsional Visualisasi Diagram Energi dan Bond Change

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.22. Matriks risiko Visualisasi Diagram Energi dan Bond Change

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_11_visualisasi_diagram_energi_dan_(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.12 Benchmark dan Validasi Mekanisme Reaksi

Benchmark dan Validasi Mekanisme Reaksi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan workflow terarah untuk menelusuri mekanisme reaksi dari minimum reaktan dan produk menuju transition state, IRC, serta parameter termokimia aktivasi. Pada tingkat implementasi, modul memanfaatkan PySCF/engine QM, geomeTRIC, optimasi constrained, Hessian/frequency, IRC, graph chemistry dan penyimpanan reaction network. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini,

pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reactant, product, TS, Hessian, imaginary mode, IRC, barrier. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Benchmark dan Validasi Mekanisme Reaksi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 3.12, aturan kontrol yang disarankan adalah: reactant harus memiliki nilai/keadaan yang eksplisit; product harus memiliki nilai/keadaan yang eksplisit; TS harus memiliki nilai/keadaan yang eksplisit; Hessian harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

3.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reactant, product, TS, Hessian, imaginary mode, IRC. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 3.23. Spesifikasi fungsional Benchmark dan Validasi Mekanisme Reaksi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reactant dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reactant; simpan status, pesan, dan provenance.	Record reactant, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi product dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk product; simpan status, pesan, dan provenance.	Record product, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi TS dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk TS; simpan status, pesan, dan provenance.	Record TS, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan Hessian dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk Hessian; simpan status, pesan, dan provenance.	Record Hessian, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis imaginary mode dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk imaginary mode; simpan status, pesan, dan provenance.	Record imaginary mode, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi IRC dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk IRC; simpan status, pesan, dan provenance.	Record IRC, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor barrier dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk barrier; simpan status, pesan, dan provenance.	Record barrier, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika

input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

3.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 3.24. Matriks risiko Benchmark dan Validasi Mekanisme Reaksi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reactant salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	product tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	TS dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	Hessian dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau imaginary mode tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	IRC menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser barrier berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

3.12.3 Sketsa Implementasi Python

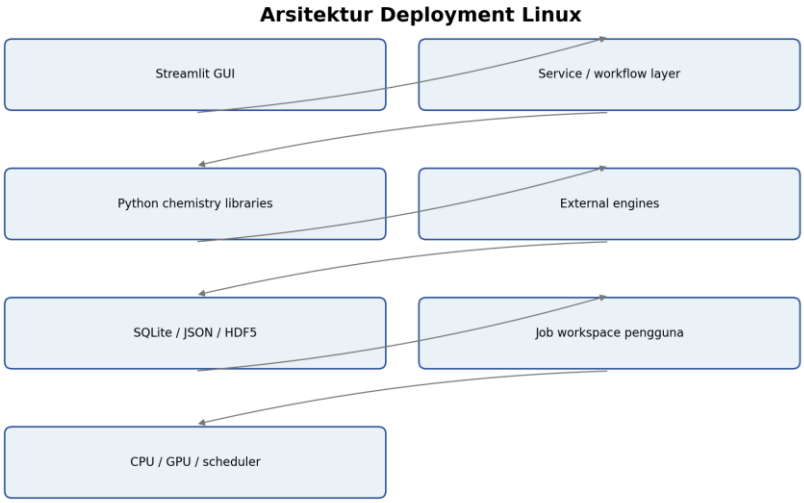
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section3Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_3_12_benchmark_dan_validasi_mekanism(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reactant")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="product")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reactant dan product perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 3.4. Kerangka pendukung untuk Benchmark dan Validasi Mekanisme Reaksi

Checklist implementasi:

- Input reactant memiliki validator dan pesan error yang jelas.
- Parameter product tersimpan dalam job specification.
- Raw output untuk TS disimpan sebelum parsing.
- Hasil Hessian mempunyai unit dan provenance.
- Kegagalan imaginary mode dapat dilacak melalui log.
- Perubahan IRC dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

3.13 Rangkuman Bab

Bab 3 menempatkan REACTIONQM: JALUR REAKSI, TRANSITION STATE, DAN MEKANISME sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

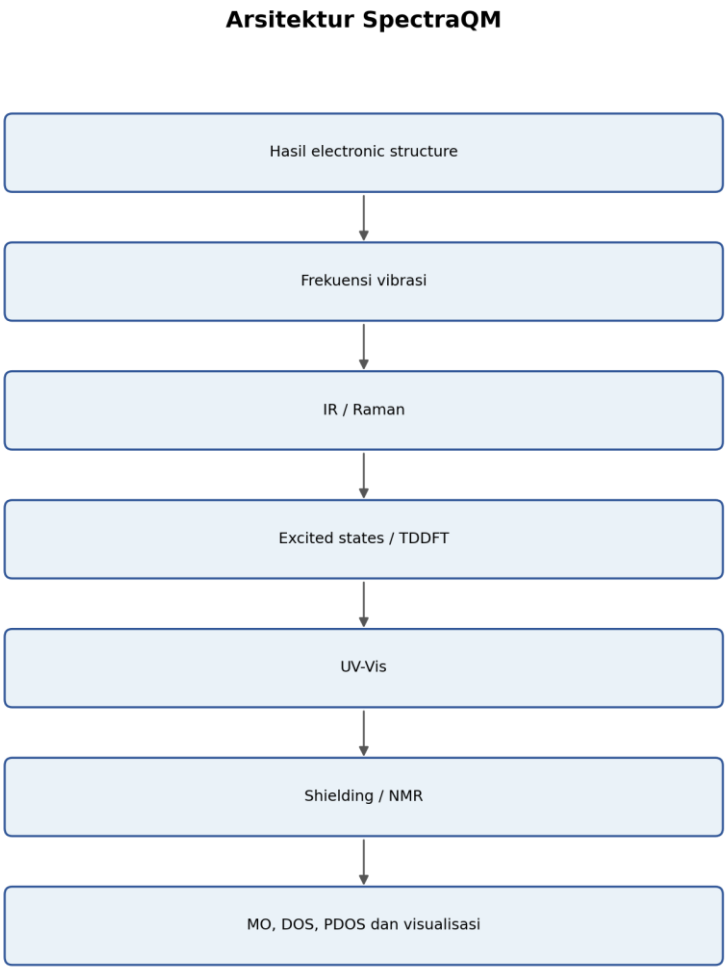
Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama,

GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 4 SPECTRAQM: SPEKTROSKOPI KOMPUTASI DAN ANALISIS ORBITAL

Bab ini berfokus pada memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Ekosistem teknis yang menjadi perhatian utama meliputi PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 4.1. Spectraqm: Spektroskopi Komputasi Dan Analisis Orbital

4.1 Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika

Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.1, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki

nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.1. Spesifikasi fungsional Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.2. Matriks risiko Arsitektur Analisis Spektrum yang Terpisah dari Termodinamika

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_1_arsitektur_analisis_spektrum_yan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.2 Frekuensi Harmonik dan Interpretasi Mode Vibrasi

Frekuensi Harmonik dan Interpretasi Mode Vibrasi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Frekuensi Harmonik dan Interpretasi Mode Vibrasi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.2, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.3. Spesifikasi fungsional Frekuensi Harmonik dan Interpretasi Mode Vibrasi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.4. Matriks risiko Frekuensi Harmonik dan Interpretasi Mode Vibrasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_2_frekuensi_harmonik_dan_interpret(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.

- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.3 Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening

Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.3, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.5. Spesifikasi fungsional Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.6. Matriks risiko Simulasi Spektrum IR: Intensitas, Scaling, dan Broadening

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_3_simulasi_spektrum_ir_intensitas_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.4 Raman Activity dan Simulasi Spektrum Raman

Raman Activity dan Simulasi Spektrum Raman harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Raman Activity dan Simulasi Spektrum Raman, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.4, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki

nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.7. Spesifikasi fungsional Raman Activity dan Simulasi Spektrum Raman

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.8. Matriks risiko Raman Activity dan Simulasi Spektrum Raman

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.4.3 Sketsa Implementasi Python

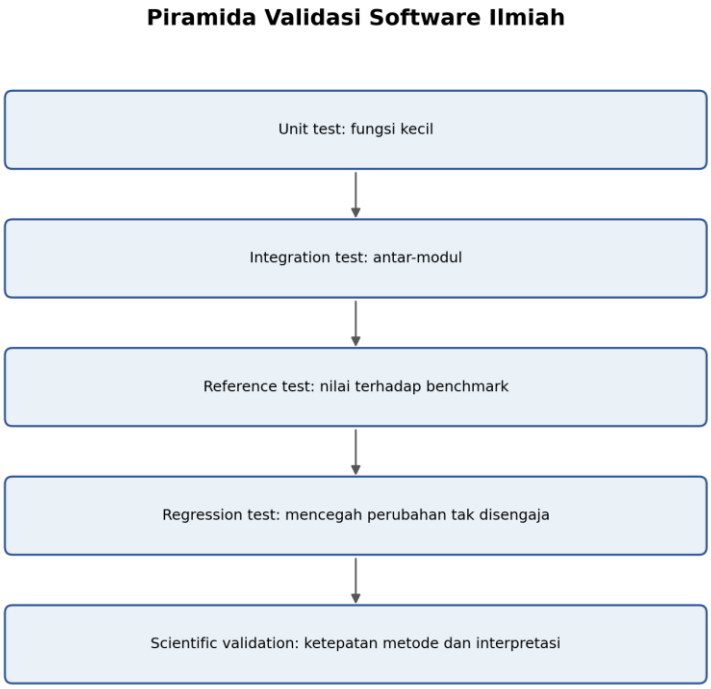
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_4_raman_activity_dan_simulasi_spek(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 4.2. Kerangka pendukung untuk Raman Activity dan Simulasi Spektrum Raman

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.5 TDDFT, Excitation Energy, dan UV-Vis

TDDFT, Excitation Energy, dan UV-Vis harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal,

vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk TDDFT, Excitation Energy, dan UV-Vis, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.5, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.9. Spesifikasi fungsional TDDFT, Excitation Energy, dan UV-Vis

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.10. Matriks risiko TDDFT, Excitation Energy, dan UV-Vis

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_5_tddft_excitation_energy_dan_uv_v(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.

- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.6 Oscillator Strength dan Assignment Transisi Elektronik

Oscillator Strength dan Assignment Transisi Elektronik harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Oscillator Strength dan Assignment Transisi Elektronik, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.6, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.11. Spesifikasi fungsional Oscillator Strength dan Assignment Transisi Elektronik

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.12. Matriks risiko Oscillator Strength dan Assignment Transisi Elektronik

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_6_oscillator_strength_dan_assignme(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.7 NMR Shielding, Chemical Shift, dan Referensi

NMR Shielding, Chemical Shift, dan Referensi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk NMR Shielding, Chemical Shift, dan Referensi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.7, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.13. Spesifikasi fungsional NMR Shielding, Chemical Shift, dan Referensi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.14. Matriks risiko NMR Shielding, Chemical Shift, dan Referensi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_7_nmr_shielding_chemical_shift_dan(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.8 HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals

HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini,

pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.8, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.15. Spesifikasi fungsional HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.16. Matriks risiko HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.8.3 Sketsa Implementasi Python

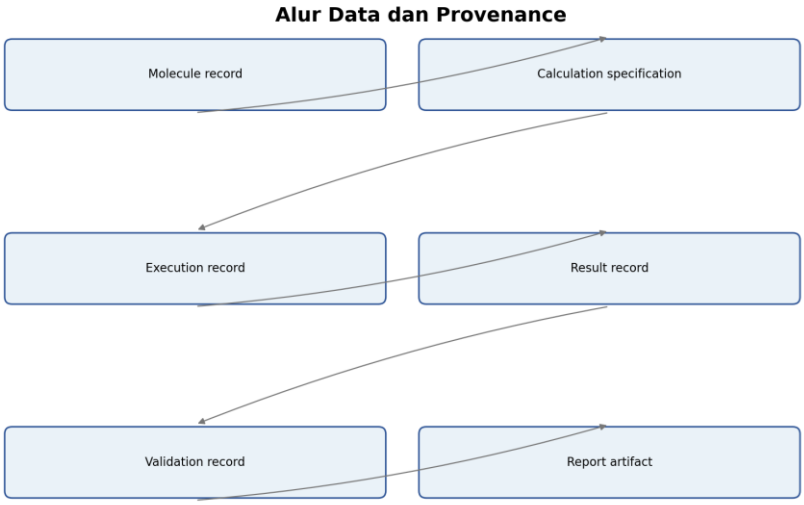
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_8_homo_lumo_gap_energi_dan_frontie(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 4.3. Kerangka pendukung untuk HOMO, LUMO, Gap Energi, dan Frontier Molecular Orbitals

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.9 DOS dan Projected Density of States

DOS dan Projected Density of States harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk DOS dan Projected Density of States, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.9, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki

nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.17. Spesifikasi fungsional DOS dan Projected Density of States

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.18. Matriks risiko DOS dan Projected Density of States

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_9_dos_dan_projected_density_of_sta(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.10 Perbandingan Spektrum Teoretis dan Eksperimen

Perbandingan Spektrum Teoretis dan Eksperimen harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Perbandingan Spektrum Teoretis dan Eksperimen, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.10, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.19. Spesifikasi fungsional Perbandingan Spektrum Teoretis dan Eksperimen

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.20. Matriks risiko Perbandingan Spektrum Teoretis dan Eksperimen

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_10_perbandingan_spektrum_teoretis_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.

- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.11 Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab

Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan frequency, intensity, broadening, assignment, scaling, comparison, export. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.11, aturan kontrol yang disarankan adalah: frequency harus memiliki nilai/keadaan yang eksplisit; intensity harus memiliki nilai/keadaan yang eksplisit; broadening harus memiliki nilai/keadaan yang eksplisit; assignment harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar frequency, intensity, broadening, assignment, scaling, comparison. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.21. Spesifikasi fungsional Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan assignment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk assignment; simpan status, pesan, dan provenance.	Record assignment, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis scaling dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk scaling; simpan status, pesan, dan provenance.	Record scaling, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi comparison dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk comparison; simpan status, pesan, dan provenance.	Record comparison, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor export dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk export; simpan status, pesan, dan provenance.	Record export, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.22. Matriks risiko Uncertainty, Scaling Factor, dan Pelaporan yang Bertanggung Jawab

Risiko	Penyebab	Mitigasi
Input tidak konsisten	frequency salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	intensity tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	broadening dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	assignment dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau scaling tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	comparison menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser export berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_11_uncertainty_scaling_factor_dan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="frequency")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="intensity")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk frequency dan intensity perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input frequency memiliki validator dan pesan error yang jelas.
- Parameter intensity tersimpan dalam job specification.
- Raw output untuk broadening disimpan sebelum parsing.
- Hasil assignment mempunyai unit dan provenance.
- Kegagalan scaling dapat dilacak melalui log.
- Perubahan comparison dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.12 Desain Dashboard SpectraQM dan Ekspor Gambar/Data

Desain Dashboard SpectraQM dan Ekspor Gambar/Data harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah memisahkan dan mengintegrasikan analisis IR, Raman, UV-Vis, NMR, orbital molekul, DOS/PDOS, serta perbandingan komputasi-eksperimen. Pada tingkat implementasi, modul memanfaatkan PySCF dan parser engine eksternal, vibrational analysis, TDDFT, shielding data, NumPy/SciPy, Matplotlib dan data eksperimen. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Desain Dashboard SpectraQM dan Ekspor Gambar/Data, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 4.12, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki

nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

4.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 4.23. Spesifikasi fungsional Desain Dashboard SpectraQM dan Ekspor Gambar/Data

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan frequency dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk frequency; simpan status, pesan, dan provenance.	Record frequency, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark intensity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk intensity; simpan status, pesan, dan provenance.	Record intensity, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi broadening dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk broadening; simpan status, pesan, dan provenance.	Record broadening, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

4.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 4.24. Matriks risiko Desain Dashboard SpectraQM dan Ekspor Gambar/Data

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

4.12.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

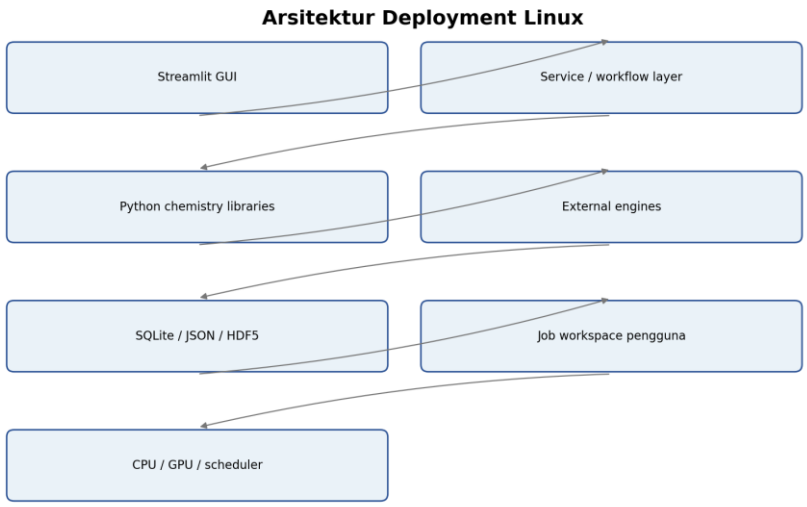
provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section4Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_4_12_desain_dashboard_spectraqm_dan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 4.4. Kerangka pendukung untuk Desain Dashboard SpectraQM dan Ekspor Gambar/Data

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

4.13 Rangkuman Bab

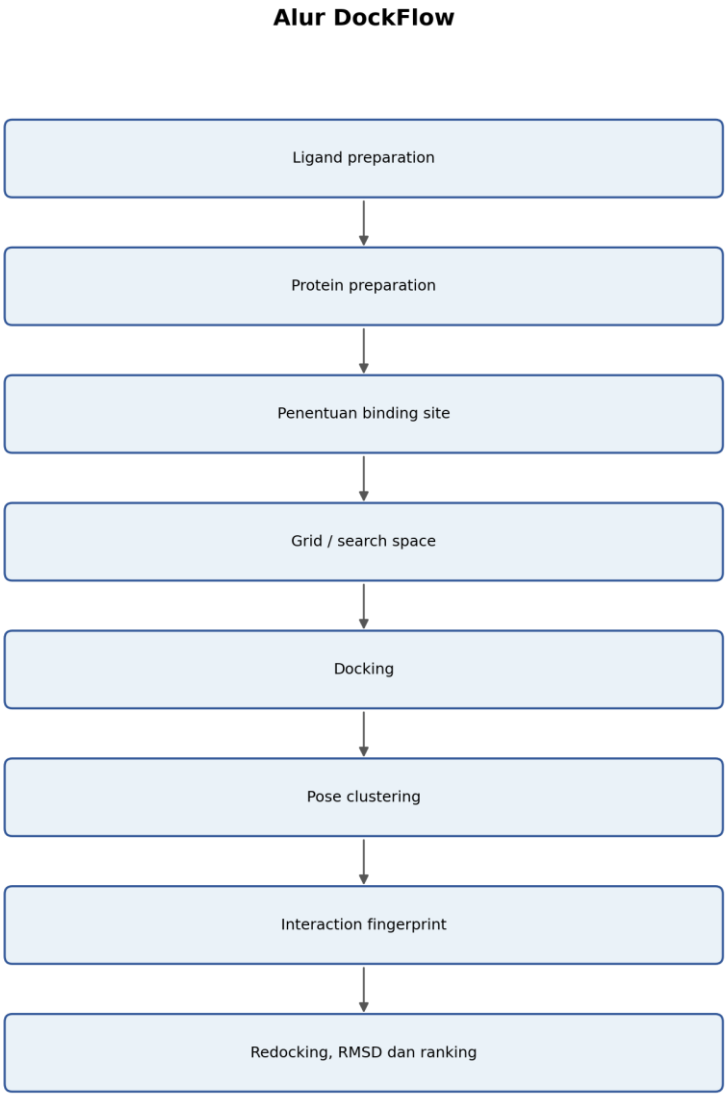
Bab 4 menempatkan SPECTRAQM: SPEKTROSKOPI KOMPUTASI DAN ANALISIS ORBITAL sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 5 DOCKFLOW: MOLECULAR DOCKING DAN ANALISIS INTERAKSI LIGAN-PROTEIN

Bab ini berfokus pada membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Ekosistem teknis yang menjadi perhatian utama meliputi RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 5.1. Dockflow: Molecular Docking Dan Analisis Interaksi Ligan-Protein

5.1 Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut

Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job

selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.1, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.1. Spesifikasi fungsional Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.2. Matriks risiko Docking sebagai Hipotesis Pose, Bukan Bukti Afinitas Absolut

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine,

penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_1_docking_sebagai_hipotesis_pose_b(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.2 Preparasi Ligan untuk Docking yang Konsisten

Preparasi Ligan untuk Docking yang Konsisten harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Preparasi Ligan untuk Docking yang Konsisten, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan

transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.2, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.3. Spesifikasi fungsional Preparasi Ligan untuk Docking yang Konsisten

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.4. Matriks risiko Preparasi Ligan untuk Docking yang Konsisten

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_2_preparasi_ligan_untuk_docking_ya(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.

- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.3 Preparasi Protein, Protonasi, dan Penanganan Komponen Kristal

Preparasi Protein, Protonasi, dan Penanganan Komponen Kristal harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Preparasi Protein, Protonasi, dan Penanganan Komponen

Kristal, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.3, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.5. Spesifikasi fungsional Preparasi Protein, Protonasi, dan Penanganan Komponen Kristal

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.6. Matriks risiko Preparasi Protein, Protonasi, dan Penanganan Komponen Kristal

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output

Risiko	Penyebab	Mitigasi
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_3_preparasi_protein_protonasi_dan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
```

```
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.4 Penetapan Binding Site, Grid Center, dan Search Space

Penetapan Binding Site, Grid Center, dan Search Space harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine

dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Penetapan Binding Site, Grid Center, dan Search Space, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.4, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error

sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.7. Spesifikasi fungsional Penetapan Binding Site, Grid Center, dan Search Space

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.8. Matriks risiko Penetapan Binding Site, Grid Center, dan Search Space

Risiko	Penyebab	Mitigasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.4.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

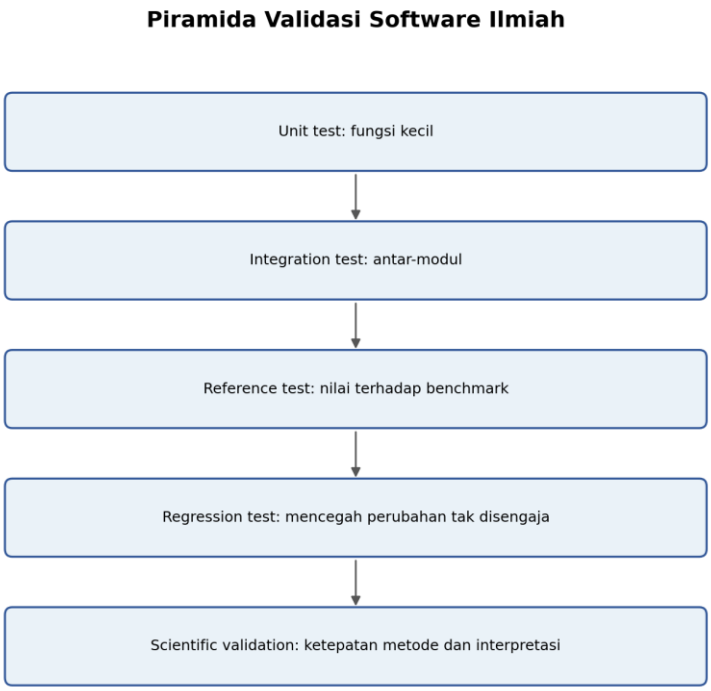
```
from dataclasses import dataclass
from pathlib import Path

@dataclass
```

```
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_4_penetapan_binding_site_grid_cent(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 5.2. Kerangka pendukung untuk Penetapan Binding Site, Grid Center, dan Search Space

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.5 Parameter AutoDock Vina, Exhaustiveness, dan Replikasi

Parameter AutoDock Vina, Exhaustiveness, dan Replikasi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser

PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Parameter AutoDock Vina, Exhaustiveness, dan Replikasi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.5, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.9. Spesifikasi fungsional Parameter AutoDock Vina, Exhaustiveness, dan Replikasi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.10. Matriks risiko Parameter AutoDock Vina, Exhaustiveness, dan Replikasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_5_parameter_autodock_vina_exhausti(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.

- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.6 Pose Clustering dan Seleksi Konformasi Representatif

Pose Clustering dan Seleksi Konformasi Representatif harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Pose Clustering dan Seleksi Konformasi Representatif, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.6, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.11. Spesifikasi fungsional Pose Clustering dan Seleksi Konformasi Representatif

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.12. Matriks risiko Pose Clustering dan Seleksi Konformasi Representatif

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_6_pose_clustering_dan_seleksi_konf(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.

- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.7 Ikatan Hidrogen dan Kontak Polar

Ikatan Hidrogen dan Kontak Polar harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Ikatan Hidrogen dan Kontak Polar, catatan minimal mencakup

identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.7, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.13. Spesifikasi fungsional Ikatan Hidrogen dan Kontak Polar

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.14. Matriks risiko Ikatan Hidrogen dan Kontak Polar

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output

Risiko	Penyebab	Mitigasi
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_7_ikatan_hidrogen_dan_kontak_polar(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
```

```
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.8 Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge

Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine

dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.8, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error

sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.15. Spesifikasi fungsional Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.16. Matriks risiko Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge

Risiko	Penyebab	Mitigasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.8.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

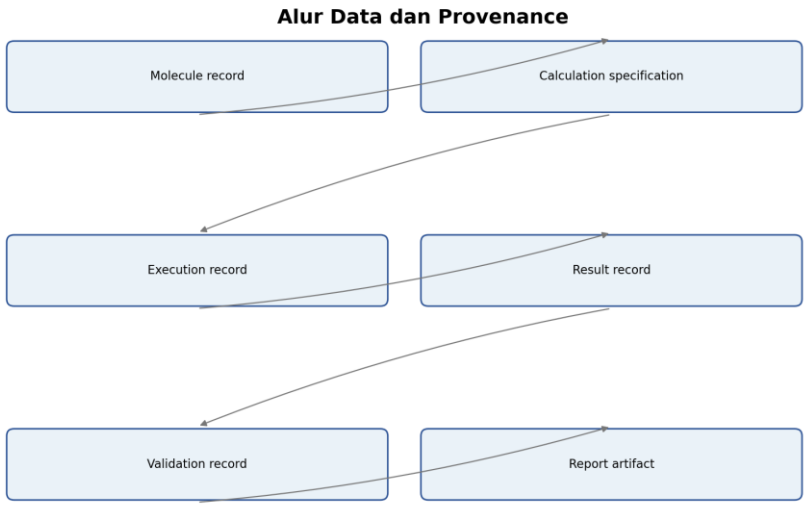
```
from dataclasses import dataclass
from pathlib import Path

@dataclass
```

```
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_8_interaksi_hidrofobik_kation_dan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 5.3. Kerangka pendukung untuk Interaksi Hidrofobik, π-π, Kation-π, dan Salt Bridge

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.

- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.9 Interaction Fingerprint dan Residue Frequency

Interaction Fingerprint dan Residue Frequency harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Interaction Fingerprint dan Residue Frequency, catatan minimal

mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.9, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.17. Spesifikasi fungsional Interaction Fingerprint dan Residue Frequency

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.18. Matriks risiko Interaction Fingerprint dan Residue Frequency

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output

Risiko	Penyebab	Mitigasi
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_9_interaction_fingerprint_dan_resi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
```

```
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.10 Redocking, RMSD, dan Validasi Protokol

Redocking, RMSD, dan Validasi Protokol harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine

dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Redocking, RMSD, dan Validasi Protokol, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.10, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error

sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.19. Spesifikasi fungsional Redocking, RMSD, dan Validasi Protokol

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.20. Matriks risiko Redocking, RMSD, dan Validasi Protokol

Risiko	Penyebab	Mitigasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
```

```
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_10_redocking_rmsd_dan_validasi_pro(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.11 Ranking Multikriteria setelah Docking

Ranking Multikriteria setelah Docking harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering,

interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Ranking Multikriteria setelah Docking, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk

analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.11, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.21. Spesifikasi fungsional Ranking Multikriteria setelah Docking

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.22. Matriks risiko Ranking Multikriteria setelah Docking

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_11_ranking_multikriteria_setelah_d(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.

- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.12 Integrasi DockFlow dengan QM, MD, dan QSAR

Integrasi DockFlow dengan QM, MD, dan QSAR harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah membuat docking sebagai workflow tervalidasi dari preparasi struktur sampai redocking, pose clustering, interaction fingerprint, dan ranking kandidat. Pada tingkat implementasi, modul memanfaatkan RDKit, AutoDock Vina, parser PDB/PDBQT, protein preparation tools, spatial analysis, interaction fingerprints, NumPy/Pandas dan visualisasi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Integrasi DockFlow dengan QM, MD, dan QSAR, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 5.12, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

5.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 5.23. Spesifikasi fungsional Integrasi DockFlow dengan QM, MD, dan QSAR

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

5.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 5.24. Matriks risiko Integrasi DockFlow dengan QM, MD, dan QSAR

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

5.12.3 Sketsa Implementasi Python

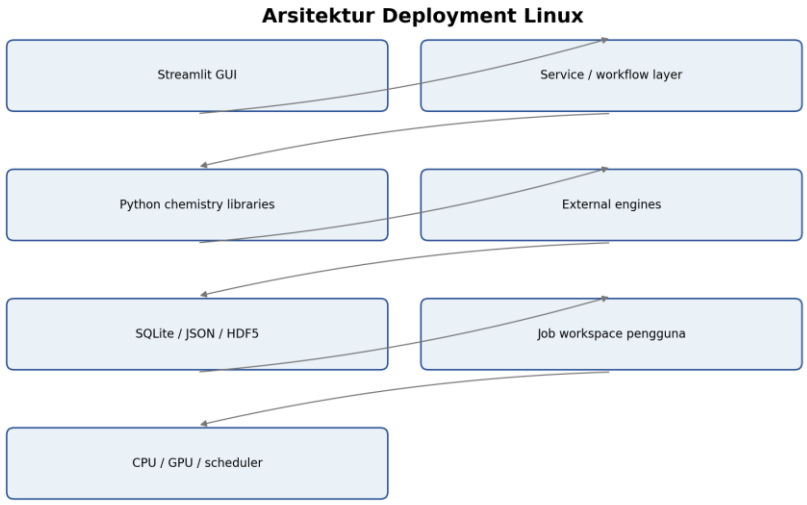
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section5Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_5_12_integrasi_dockflow_dengan_qm_md(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 5.4. Kerangka pendukung untuk Integrasi DockFlow dengan QM, MD, dan QSAR

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

5.13 Rangkuman Bab

Bab 5 menempatkan DOCKFLOW: MOLECULAR DOCKING DAN ANALISIS INTERAKSI LIGAN-PROTEIN sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang

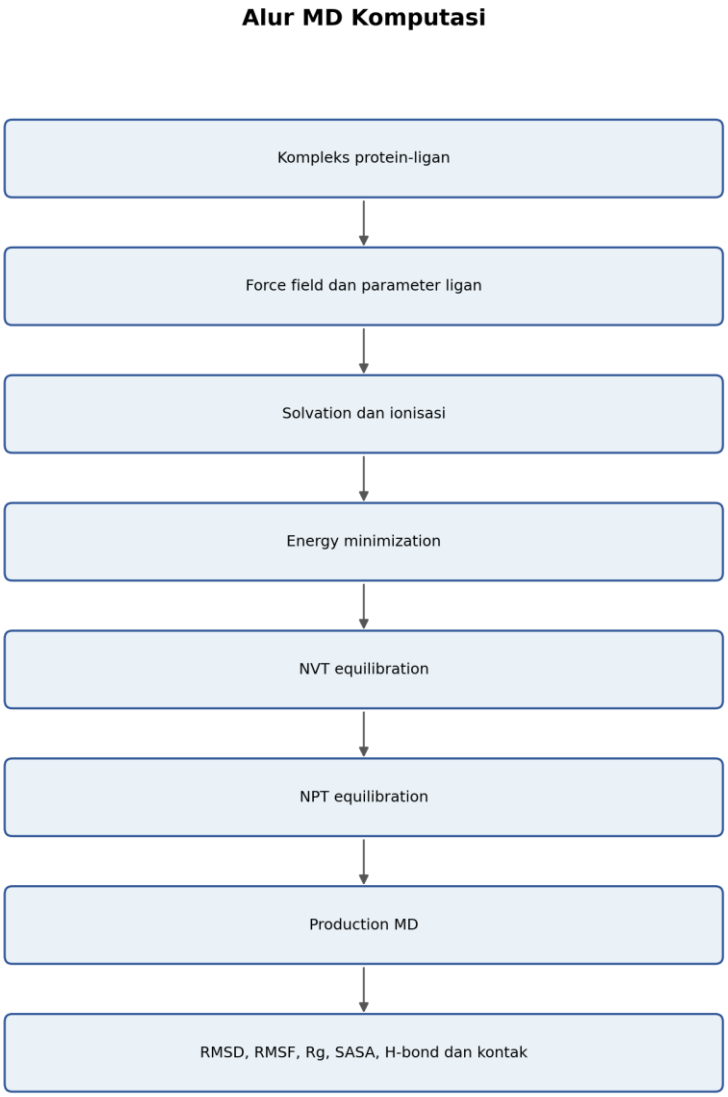
konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 6 MD KOMPUTASI: MOLECULAR DYNAMICS BERBASIS OPENMM

Bab ini berfokus pada menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Ekosistem teknis yang menjadi perhatian utama meliputi OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 6.1. Md Komputasi: Molecular Dynamics Berbasis Openmm

6.1 Posisi Molecular Dynamics setelah Docking

Posisi Molecular Dynamics setelah Docking harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job

selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Posisi Molecular Dynamics setelah Docking, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.1, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.1. Spesifikasi fungsional Posisi Molecular Dynamics setelah Docking

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.2. Matriks risiko Posisi Molecular Dynamics setelah Docking

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_1_posisi_molecular_dynamics_setela(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.2 Pemilihan Force Field dan Konsistensi Parameter

Pemilihan Force Field dan Konsistensi Parameter harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Pemilihan Force Field dan Konsistensi Parameter, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.2, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.3. Spesifikasi fungsional Pemilihan Force Field dan Konsistensi Parameter

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.4. Matriks risiko Pemilihan Force Field dan Konsistensi Parameter

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_2_pemilihan_force_field_dan_konsis(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.

- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.3 Parameterisasi Ligan dan Pemeriksaan Muatan

Parameterisasi Ligan dan Pemeriksaan Muatan harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Parameterisasi Ligan dan Pemeriksaan Muatan, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.3, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.5. Spesifikasi fungsional Parameterisasi Ligan dan Pemeriksaan Muatan

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.6. Matriks risiko Parameterisasi Ligan dan Pemeriksaan Muatan

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output

Risiko	Penyebab	Mitigasi
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_3_parameterisasi_ligan_dan_pemerik(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
```

```
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.4 Solvation Box, Model Air, Ion, dan Periodic Boundary

Solvation Box, Model Air, Ion, dan Periodic Boundary harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data

masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Solvation Box, Model Air, Ion, dan Periodic Boundary, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.4, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat

keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.7. Spesifikasi fungsional Solvation Box, Model Air, Ion, dan Periodic Boundary

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.8. Matriks risiko Solvation Box, Model Air, Ion, dan Periodic Boundary

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.4.3 Sketsa Implementasi Python

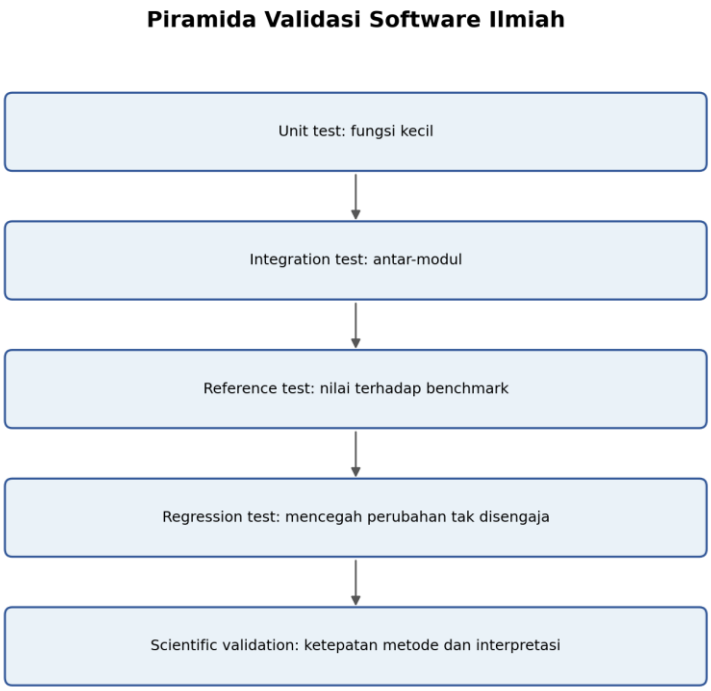
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_4_solvation_box_model_air_ion_dan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 6.2. Kerangka pendukung untuk Solvation Box, Model Air, Ion, dan Periodic Boundary

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.5 Energy Minimization dan Pemeriksaan Geometri Awal

Energy Minimization dan Pemeriksaan Geometri Awal harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field

biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Energy Minimization dan Pemeriksaan Geometri Awal, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.5, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.9. Spesifikasi fungsional Energy Minimization dan Pemeriksaan Geometri Awal

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.10. Matriks risiko Energy Minimization dan Pemeriksaan Geometri Awal

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_5_energy_minimization_dan_pemeriks(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.

- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.6 Equilibration NVT: Suhu dan Stabilitas Sistem

Equilibration NVT: Suhu dan Stabilitas Sistem harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Equilibration NVT: Suhu dan Stabilitas Sistem, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.6, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.11. Spesifikasi fungsional Equilibration NVT: Suhu dan Stabilitas Sistem

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.12. Matriks risiko Equilibration NVT: Suhu dan Stabilitas Sistem

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_6_equilibration_nvt_suhu_dan_stabi(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.7 Equilibration NPT: Tekanan, Densitas, dan Volume

Equilibration NPT: Tekanan, Densitas, dan Volume harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Equilibration NPT: Tekanan, Densitas, dan Volume, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.7, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.13. Spesifikasi fungsional Equilibration NPT: Tekanan, Densitas, dan Volume

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.14. Matriks risiko Equilibration NPT: Tekanan, Densitas, dan Volume

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_7_equilibration_npt_tekanan_densit(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.8 Production MD, Time Step, Constraints, dan Output

Production MD, Time Step, Constraints, dan Output harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil.

Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Production MD, Time Step, Constraints, dan Output, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.8, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.15. Spesifikasi fungsional Production MD, Time Step, Constraints, dan Output

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.16. Matriks risiko Production MD, Time Step, Constraints, dan Output

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.8.3 Sketsa Implementasi Python

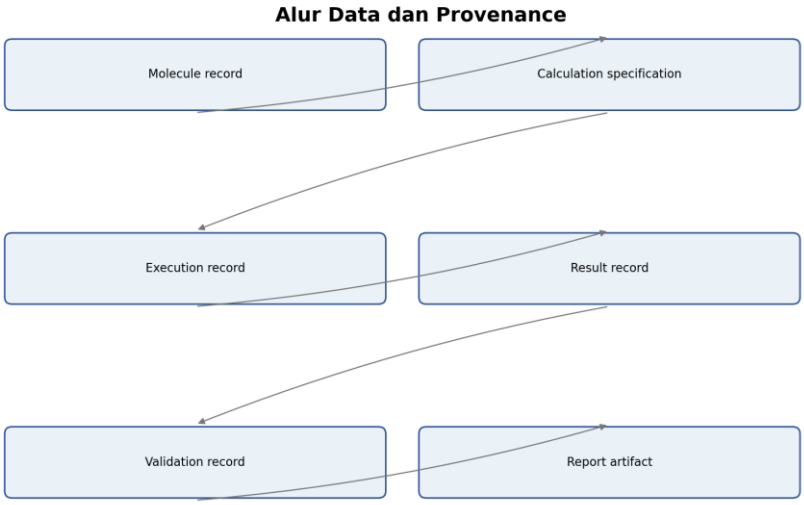
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_8_production_md_time_step_constrai(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 6.3. Kerangka pendukung untuk Production MD, Time Step, Constraints, dan Output

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.9 Checkpoint, Resume, dan Ketahanan terhadap Interupsi

Checkpoint, Resume, dan Ketahanan terhadap Interupsi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Checkpoint, Resume, dan Ketahanan terhadap Interupsi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.9, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki

nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.17. Spesifikasi fungsional Checkpoint, Resume, dan Ketahanan terhadap Interupsi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.18. Matriks risiko Checkpoint, Resume, dan Ketahanan terhadap Interupsi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_9_checkpoint_resume_dan_ketahanan_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.10 RMSD, RMSF, Radius of Gyration, dan SASA

RMSD, RMSF, Radius of Gyration, dan SASA harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk RMSD, RMSF, Radius of Gyration, dan SASA, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.10, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.19. Spesifikasi fungsional RMSD, RMSF, Radius of Gyration, dan SASA

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.20. Matriks risiko RMSD, RMSF, Radius of Gyration, dan SASA

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_10_rmsd_rmsf_radius_of_gyration_da(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.

- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.11 H-Bond, Contact Occupancy, Distance, dan Analisis Ligan

H-Bond, Contact Occupancy, Distance, dan Analisis Ligan harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk H-Bond, Contact Occupancy, Distance, dan Analisis Ligan, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.11, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.21. Spesifikasi fungsional H-Bond, Contact Occupancy, Distance, dan Analisis Ligan

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.22. Matriks risiko H-Bond, Contact Occupancy, Distance, dan Analisis Ligan

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_11_h_bond_contact_occupancy_distan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.12 Replikasi, Konvergensi, dan Pelaporan Simulasi MD

Replikasi, Konvergensi, dan Pelaporan Simulasi MD harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyediakan antarmuka dan workflow simulasi dinamika molekul yang reproducible, dapat di-resume, dan otomatis menganalisis stabilitas serta interaksi. Pada tingkat implementasi, modul memanfaatkan OpenMM, force field biomolekuler, parameter ligan, PME, integrator, checkpoint, DCD/trajectory, MDAnalysis/MDTraj dan plotting. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan force field, integrator, ensemble, trajectory, checkpoint, RMSD, contact. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Replikasi, Konvergensi, dan Pelaporan Simulasi MD, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 6.12, aturan kontrol yang disarankan adalah: force field harus memiliki nilai/keadaan yang eksplisit; integrator harus memiliki

nilai/keadaan yang eksplisit; ensemble harus memiliki nilai/keadaan yang eksplisit; trajectory harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

6.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar force field, integrator, ensemble, trajectory, checkpoint, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 6.23. Spesifikasi fungsional Replikasi, Konvergensi, dan Pelaporan Simulasi MD

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi force field dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk force field; simpan status, pesan, dan provenance.	Record force field, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi integrator dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk integrator; simpan status, pesan, dan provenance.	Record integrator, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi ensemble dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ensemble; simpan status, pesan, dan provenance.	Record ensemble, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan trajectory dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk trajectory; simpan status, pesan, dan provenance.	Record trajectory, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checkpoint dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checkpoint; simpan status, pesan, dan provenance.	Record checkpoint, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor contact dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk contact; simpan status, pesan, dan provenance.	Record contact, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

6.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 6.24. Matriks risiko Replikasi, Konvergensi, dan Pelaporan Simulasi MD

Risiko	Penyebab	Mitigasi
Input tidak konsisten	force field salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	integrator tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	ensemble dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	trajectory dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checkpoint tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser contact berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

6.12.3 Sketsa Implementasi Python

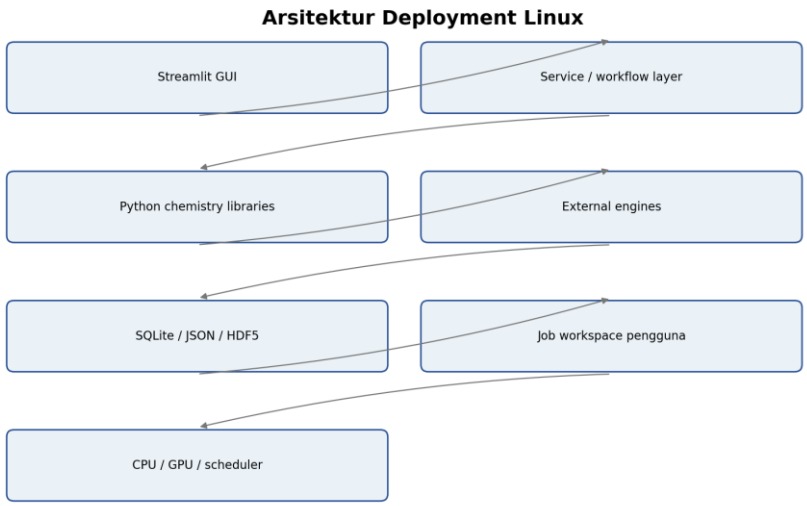
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section6Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_6_12_replikasi_konvergensi_dan_pelap(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="force field")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="integrator")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk force field dan integrator perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 6.4. Kerangka pendukung untuk Replikasi, Konvergensi, dan Pelaporan Simulasi MD

Checklist implementasi:

- Input force field memiliki validator dan pesan error yang jelas.
- Parameter integrator tersimpan dalam job specification.
- Raw output untuk ensemble disimpan sebelum parsing.
- Hasil trajectory mempunyai unit dan provenance.
- Kegagalan checkpoint dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

6.13 Rangkuman Bab

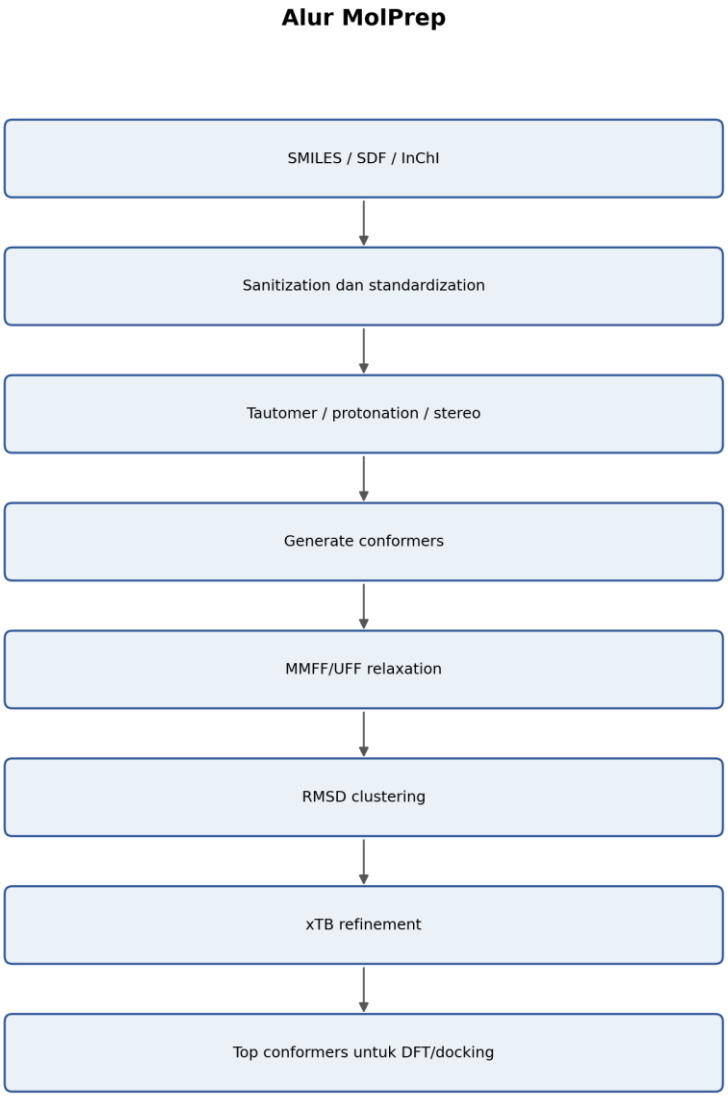
Bab 6 menempatkan MD KOMPUTASI: MOLECULAR DYNAMICS BERBASIS OPENMM sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 7 MOLPREP: CONFORMER DAN MOLECULAR PREPARATION STUDIO

Bab ini berfokus pada mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Ekosistem teknis yang menjadi perhatian utama meliputi RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 7.1. Molprep: Conformer Dan Molecular Preparation Studio

7.1 Mengapa Geometri Awal Menentukan Mutu Hasil

Mengapa Geometri Awal Menentukan Mutu Hasil harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga

apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Mengapa Geometri Awal Menentukan Mutu Hasil, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.1, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.1. Spesifikasi fungsional Mengapa Geometri Awal Menentukan Mutu Hasil

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.2. Matriks risiko Mengapa Geometri Awal Menentukan Mutu Hasil

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_1_mengapa_geometri_awal_menentukan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.2 Parsing SMILES, InChI, SDF, MOL, dan XYZ

Parsing SMILES, InChI, SDF, MOL, dan XYZ harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Parsing SMILES, InChI, SDF, MOL, dan XYZ, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.2, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.3. Spesifikasi fungsional Parsing SMILES, InChI, SDF, MOL, dan XYZ

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.4. Matriks risiko Parsing SMILES, InChI, SDF, MOL, dan XYZ

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_2_parsing_smiles_inchi_sdf_mol_dan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.

- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.3 Sanitization, Valence Check, dan Kekonsistenan Struktur

Sanitization, Valence Check, dan Kekonsistenan Struktur harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Sanitization, Valence Check, dan Kekonsistenan Struktur, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.3, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.5. Spesifikasi fungsional Sanitization, Valence Check, dan Kekonsistenan Struktur

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan format data dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk format data; simpan status, pesan, dan provenance.	Record format data, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark validasi sintaks dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk validasi sintaks; simpan status, pesan, dan provenance.	Record validasi sintaks, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi unit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk unit; simpan status, pesan, dan provenance.	Record unit, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.6. Matriks risiko Sanitization, Valence Check, dan Kekonsistenan Struktur

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_3_sanitization_valence_check_dan_k(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.4 Salt Removal, Fragment Selection, dan Standardization

Salt Removal, Fragment Selection, dan Standardization harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Salt Removal, Fragment Selection, dan Standardization, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.4, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki

nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.7. Spesifikasi fungsional Salt Removal, Fragment Selection, dan Standardization

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.8. Matriks risiko Salt Removal, Fragment Selection, dan Standardization

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.4.3 Sketsa Implementasi Python

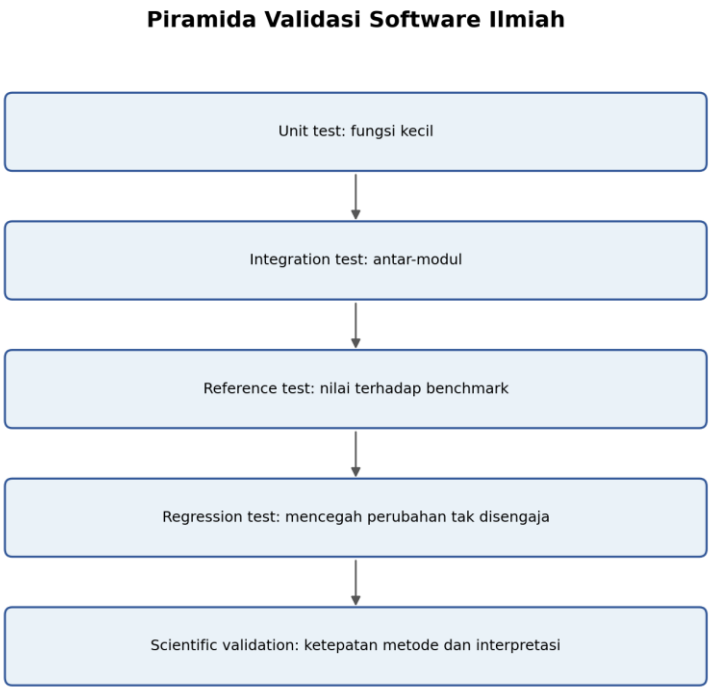
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_4_salt_removal_fragment_selection_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 7.2. Kerangka pendukung untuk Salt Removal, Fragment Selection, dan Standardization

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.5 Tautomer Generation dan Konsekuensi terhadap Energi

Tautomer Generation dan Konsekuensi terhadap Energi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF,

sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Tautomer Generation dan Konsekuensi terhadap Energi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.5, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.9. Spesifikasi fungsional Tautomer Generation dan Konsekuensi terhadap Energi

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.10. Matriks risiko Tautomer Generation dan Konsekuensi terhadap Energi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_5_tautomer_generation_dan_konsekue(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.

- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.6 Protonation State, Charge, dan Kondisi Kimia

Protonation State, Charge, dan Kondisi Kimia harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Protonation State, Charge, dan Kondisi Kimia, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.6, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.11. Spesifikasi fungsional Protonation State, Charge, dan Kondisi Kimia

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.12. Matriks risiko Protonation State, Charge, dan Kondisi Kimia

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_6_protonation_state_charge_dan_kon(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.7 Stereoisomer Enumeration dan Integritas Stereokimia

Stereoisomer Enumeration dan Integritas Stereokimia harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Stereoisomer Enumeration dan Integritas Stereokimia, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.7, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.13. Spesifikasi fungsional Stereoisomer Enumeration dan Integritas Stereokimia

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.14. Matriks risiko Stereoisomer Enumeration dan Integritas Stereokimia

Risiko	Penyebab	Mitigasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
```

```
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_7_stereoisomer_enumeration_dan_int(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.8 Conformer Generation dan Sampling Ruang Konformasi

Conformer Generation dan Sampling Ruang Konformasi harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi

keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Conformer Generation dan Sampling Ruang Konformasi, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.8, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.15. Spesifikasi fungsional Conformer Generation dan Sampling Ruang Konformasi

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.16. Matriks risiko Conformer Generation dan Sampling Ruang Konformasi

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.8.3 Sketsa Implementasi Python

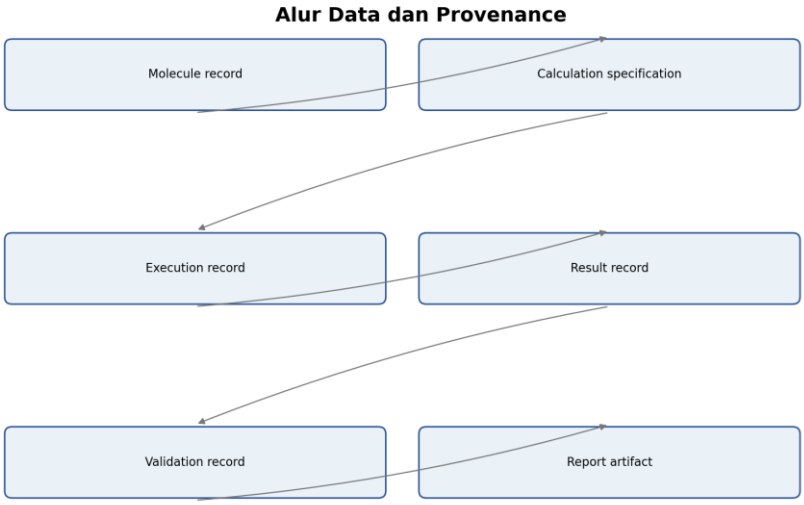
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_8_conformer_generation_dan_samplin(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 7.3. Kerangka pendukung untuk Conformer Generation dan Sampling Ruang Konformasi

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.9 UFF/MMFF Relaxation dan Energy Filtering

UFF/MMFF Relaxation dan Energy Filtering harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk UFF/MMFF Relaxation dan Energy Filtering, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.9, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki

nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.17. Spesifikasi fungsional UFF/MMFF Relaxation dan Energy Filtering

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.18. Matriks risiko UFF/MMFF Relaxation dan Energy Filtering

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_9_uff_mmff_relaxation_dan_energy_f(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.10 RMSD Clustering dan Penghapusan Duplikat

RMSD Clustering dan Penghapusan Duplikat harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk RMSD Clustering dan Penghapusan Duplikat, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.10, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.19. Spesifikasi fungsional RMSD Clustering dan Penghapusan Duplikat

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.20. Matriks risiko RMSD Clustering dan Penghapusan Duplikat

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_10_rmsd_clustering_dan_penghapusan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.

- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.11 xTB Refinement dan Seleksi Top Conformer

xTB Refinement dan Seleksi Top Conformer harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk xTB Refinement dan Seleksi Top Conformer, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.11, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.21. Spesifikasi fungsional xTB Refinement dan Seleksi Top Conformer

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.22. Matriks risiko xTB Refinement dan Seleksi Top Conformer

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi

Risiko	Penyebab	Mitigasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_11_xtb_refinement_dan_seleksi_top_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
```

```
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.12 Integrasi MolPrep dengan QMScreen dan DockFlow

Integrasi MolPrep dengan QMScreen dan DockFlow harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencegah kesalahan komputasi sejak tahap awal melalui standardisasi struktur, enumerasi keadaan kimia, konformer, clustering, dan seleksi geometri. Pada tingkat implementasi, modul memanfaatkan RDKit, MMFF/UFF, InChI/SMILES/SDF, sanitization, stereochemistry, tautomer handling, RMSD clustering dan xTB refinement. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan sanitization, tautomer, protonation, stereochemistry, conformer, RMSD, energy. Kesalahan pada salah

satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Integrasi MolPrep dengan QMScreen dan DockFlow, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 7.12, aturan kontrol yang disarankan adalah: sanitization harus memiliki nilai/keadaan yang eksplisit; tautomer harus memiliki nilai/keadaan yang eksplisit; protonation harus memiliki nilai/keadaan yang eksplisit; stereochemistry harus memiliki nilai/keadaan yang eksplisit. Sistem

perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

7.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar sanitization, tautomer, protonation, stereochemistry, conformer, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 7.23. Spesifikasi fungsional Integrasi MolPrep dengan QMScreen dan DockFlow

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi sanitization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanitization; simpan status, pesan, dan provenance.	Record sanitization, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi tautomer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk tautomer; simpan status, pesan, dan provenance.	Record tautomer, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi protonation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk protonation; simpan status, pesan, dan provenance.	Record protonation, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan stereochemistry dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk stereochemistry; simpan status, pesan, dan provenance.	Record stereochemistry, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis conformer dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk conformer; simpan status, pesan, dan provenance.	Record conformer, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor energy dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk energy; simpan status, pesan, dan provenance.	Record energy, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan method dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk method; simpan status, pesan, dan provenance.	Record method, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark basis set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk basis set; simpan status, pesan, dan provenance.	Record basis set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi SCF dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk SCF; simpan status, pesan, dan provenance.	Record SCF, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

7.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 7.24. Matriks risiko Integrasi MolPrep dengan QMScreen dan DockFlow

Risiko	Penyebab	Mitigasi
Input tidak konsisten	sanitization salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	tautomer tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	protonation dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	stereochemistry dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau conformer tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser energy berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

7.12.3 Sketsa Implementasi Python

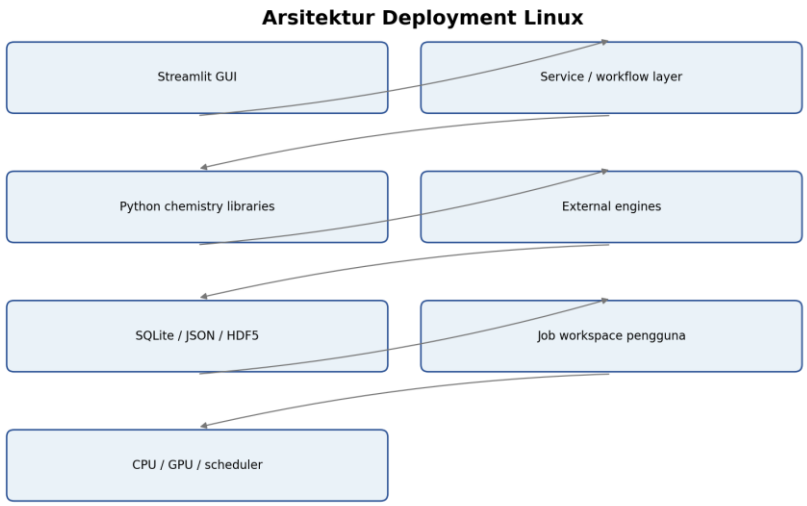
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section7Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_7_12_integrasi_molprep_dengan_qmscre(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="sanitization")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="tautomer")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk sanitization dan tautomer perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 7.4. Kerangka pendukung untuk Integrasi MolPrep dengan QMScreen dan DockFlow

Checklist implementasi:

- Input sanitization memiliki validator dan pesan error yang jelas.
- Parameter tautomer tersimpan dalam job specification.
- Raw output untuk protonation disimpan sebelum parsing.
- Hasil stereochemistry mempunyai unit dan provenance.
- Kegagalan conformer dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

7.13 Rangkuman Bab

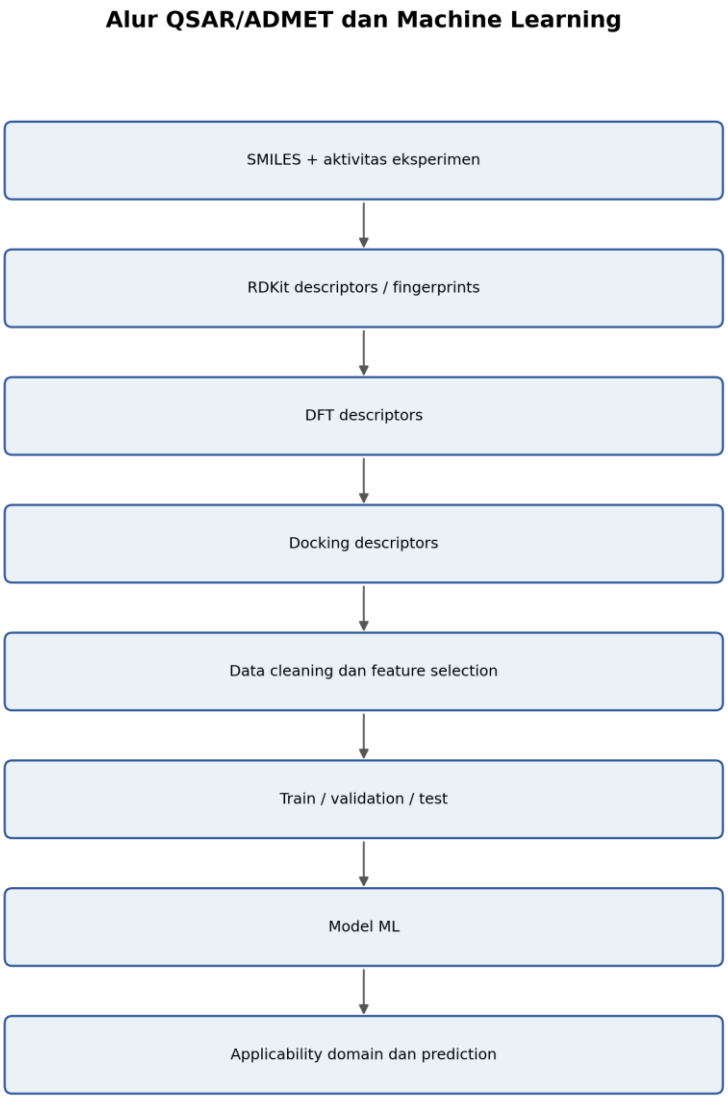
Bab 7 menempatkan MOLPREP: CONFORMER DAN MOLECULAR PREPARATION STUDIO sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 8 QSAR/ADMET STUDIO: DESKRIPTOR, MACHINE LEARNING, DAN SCREENING

Bab ini berfokus pada menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Ekosistem teknis yang menjadi perhatian utama meliputi RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 8.1. Qsar/Admet Studio: Deskriptor, Machine Learning, Dan Screening

8.1 Definisi Masalah QSAR dan Kualitas Endpoint

Definisi Masalah QSAR dan Kualitas Endpoint harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga

apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Definisi Masalah QSAR dan Kualitas Endpoint, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.1, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.1. Spesifikasi fungsional Definisi Masalah QSAR dan Kualitas Endpoint

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.2. Matriks risiko Definisi Masalah QSAR dan Kualitas Endpoint

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_1_definisi_masalah_qsar_dan_kualit(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.2 Kurasi Dataset, Duplikat, Satuan, dan Missing Value

Kurasi Dataset, Duplikat, Satuan, dan Missing Value harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Kurasi Dataset, Duplikat, Satuan, dan Missing Value, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.2, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.3. Spesifikasi fungsional Kurasi Dataset, Duplikat, Satuan, dan Missing Value

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.4. Matriks risiko Kurasi Dataset, Duplikat, Satuan, dan Missing Value

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_2_kurasi_dataset_duplikat_satuan_d(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.

- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.3 Descriptor Fisikokimia RDKit dan Fingerprints

Descriptor Fisikokimia RDKit dan Fingerprints harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Descriptor Fisikokimia RDKit dan Fingerprints, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.3, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.5. Spesifikasi fungsional Descriptor Fisikokimia RDKit dan Fingerprints

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.6. Matriks risiko Descriptor Fisikokimia RDKit dan Fingerprints

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi

Risiko	Penyebab	Mitigasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_3_descriptor_fisikokimia_rdkit_dan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
```

```
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.4 Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge

Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah

satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.4, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu

membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.7. Spesifikasi fungsional Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.8. Matriks risiko Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.4.3 Sketsa Implementasi Python

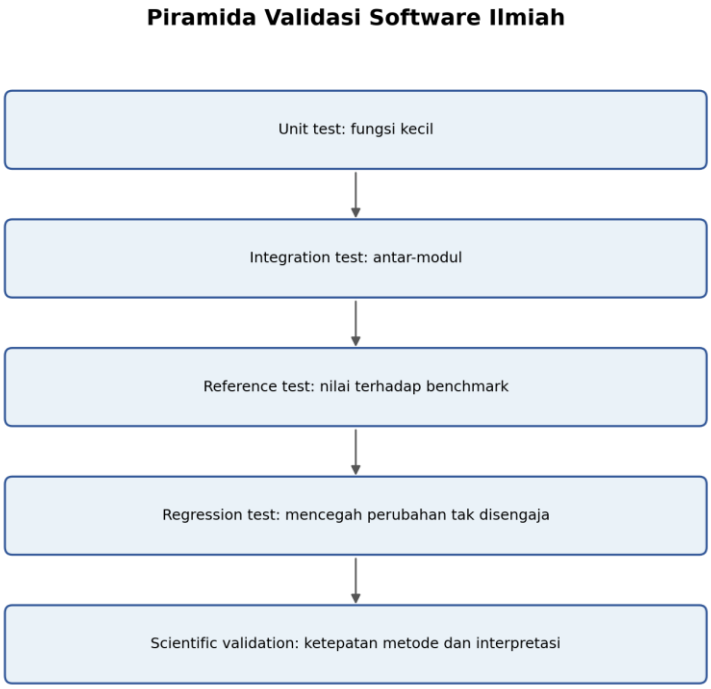
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_4_descriptor_dft_homo_lumo_gap_dip(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 8.2. Kerangka pendukung untuk Descriptor DFT: HOMO, LUMO, Gap, Dipole, dan Charge

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.5 Descriptor Docking dan Interaction Fingerprint

Descriptor Docking dan Interaction Fingerprint harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas,

NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan ligand, receptor, grid, pose, affinity, RMSD, interaction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Descriptor Docking dan Interaction Fingerprint, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.5, aturan kontrol yang disarankan adalah: ligand harus memiliki nilai/keadaan yang eksplisit; receptor harus memiliki nilai/keadaan yang eksplisit; grid harus memiliki nilai/keadaan yang eksplisit; pose harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar ligand, receptor, grid, pose, affinity, RMSD. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.9. Spesifikasi fungsional Descriptor Docking dan Interaction Fingerprint

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi ligand dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk ligand; simpan status, pesan, dan provenance.	Record ligand, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi receptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk receptor; simpan status, pesan, dan provenance.	Record receptor, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi grid dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk grid; simpan status, pesan, dan provenance.	Record grid, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan pose dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk pose; simpan status, pesan, dan provenance.	Record pose, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis affinity dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk affinity; simpan status, pesan, dan provenance.	Record affinity, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi RMSD dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk RMSD; simpan status, pesan, dan provenance.	Record RMSD, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor interaction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk interaction; simpan status, pesan, dan provenance.	Record interaction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.10. Matriks risiko Descriptor Docking dan Interaction Fingerprint

Risiko	Penyebab	Mitigasi
Input tidak konsisten	ligand salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	receptor tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	grid dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	pose dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau affinity tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	RMSD menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser interaction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_5_descriptor_docking_dan_interacti(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="ligand")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="receptor")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk ligand dan receptor perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input ligand memiliki validator dan pesan error yang jelas.
- Parameter receptor tersimpan dalam job specification.
- Raw output untuk grid disimpan sebelum parsing.
- Hasil pose mempunyai unit dan provenance.
- Kegagalan affinity dapat dilacak melalui log.
- Perubahan RMSD dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.6 Feature Cleaning, Scaling, dan Multicollinearity

Feature Cleaning, Scaling, dan Multicollinearity harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Feature Cleaning, Scaling, dan Multicollinearity, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.6, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.11. Spesifikasi fungsional Feature Cleaning, Scaling, dan Multicollinearity

Aspek	Kebutuhan	Quality control	Keluaran

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.12. Matriks risiko Feature Cleaning, Scaling, dan Multicollinearity

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_6_feature_cleaning_scaling_dan_mul(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.

- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.7 Feature Selection dan Pencegahan Data Leakage

Feature Selection dan Pencegahan Data Leakage harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Feature Selection dan Pencegahan Data Leakage, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.7, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.13. Spesifikasi fungsional Feature Selection dan Pencegahan Data Leakage

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.14. Matriks risiko Feature Selection dan Pencegahan Data Leakage

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi

Risiko	Penyebab	Mitigasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_7_feature_selection_dan_pencegahan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
```

```
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.8 Pembagian Train, Validation, Test dan Cross-Validation

Pembagian Train, Validation, Test dan Cross-Validation harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan reference calculation, benchmark set, toleransi numerik, regression test, sanity check, audit trail, cross-

check. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Pembagian Train, Validation, Test dan Cross-Validation, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.8, aturan kontrol yang disarankan adalah: reference calculation harus memiliki nilai/keadaan yang eksplisit; benchmark set harus memiliki nilai/keadaan yang eksplisit; toleransi numerik harus memiliki nilai/keadaan yang eksplisit; regression test harus memiliki nilai/keadaan yang

eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar reference calculation, benchmark set, toleransi numerik, regression test, sanity check, audit trail. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.15. Spesifikasi fungsional Pembagian Train, Validation, Test dan Cross-Validation

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan regression test dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk regression test; simpan status, pesan, dan provenance.	Record regression test, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis sanity check dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk sanity check; simpan status, pesan, dan provenance.	Record sanity check, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi audit trail dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk audit trail; simpan status, pesan, dan provenance.	Record audit trail, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor cross-check dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-check; simpan status, pesan, dan provenance.	Record cross-check, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.16. Matriks risiko Pembagian Train, Validation, Test dan Cross-Validation

Risiko	Penyebab	Mitigasi
Input tidak konsisten	reference calculation salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	benchmark set tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	toleransi numerik dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	regression test dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau sanity check tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	audit trail menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser cross-check berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.8.3 Sketsa Implementasi Python

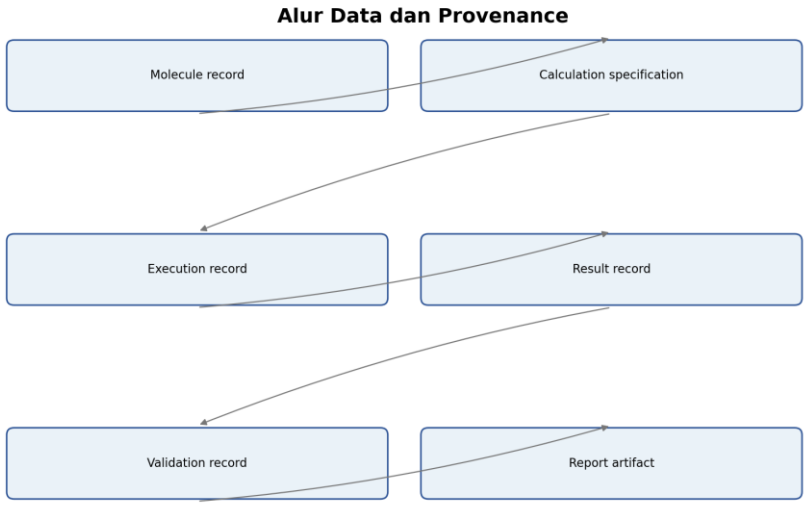
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_8_pembagian_train_validation_test_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="reference
calculation")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="benchmark set")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk reference calculation dan benchmark set perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 8.3. Kerangka pendukung untuk Pembagian Train, Validation, Test dan Cross-Validation

Checklist implementasi:

- Input reference calculation memiliki validator dan pesan error yang jelas.
- Parameter benchmark set tersimpan dalam job specification.
- Raw output untuk toleransi numerik disimpan sebelum parsing.
- Hasil regression test mempunyai unit dan provenance.
- Kegagalan sanity check dapat dilacak melalui log.
- Perubahan audit trail dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.9 Linear/PLS, Random Forest, Gradient Boosting, dan SVM

Linear/PLS, Random Forest, Gradient Boosting, dan SVM harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan

perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Linear/PLS, Random Forest, Gradient Boosting, dan SVM, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.9, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan

yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.17. Spesifikasi fungsional Linear/PLS, Random Forest, Gradient Boosting, dan SVM

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.18. Matriks risiko Linear/PLS, Random Forest, Gradient Boosting, dan SVM

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik

Risiko	Penyebab	Mitigasi
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
```

```
    multiplicity: int = 1
    method: str = "auto"

def run_8_9_linear_pls_random_forest_gradien(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.10 Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics

Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan

serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.10, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.19. Spesifikasi fungsional Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.20. Matriks risiko Evaluasi R^2, Q^2, RMSE, MAE, dan Classification Metrics

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_10_evaluasi_r_q_rmse_mae_dan_class(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.

- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.11 Y-Randomization dan Applicability Domain

Y-Randomization dan Applicability Domain harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Y-Randomization dan Applicability Domain, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.11, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.21. Spesifikasi fungsional Y-Randomization dan Applicability Domain

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.22. Matriks risiko Y-Randomization dan Applicability Domain

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_11_y_randomization_dan_applicabili(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.
- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.12 Explainability, Prediction, dan Ekspor Model

Explainability, Prediction, dan Ekspor Model harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menghubungkan descriptor cheminformatika, descriptor DFT, descriptor docking, dan data eksperimen menjadi workflow QSAR/ML yang tervalidasi. Pada tingkat implementasi, modul memanfaatkan RDKit descriptors/fingerprints, Pandas, NumPy, scikit-learn, feature selection, model validation, applicability domain dan serialisasi model. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan descriptor, feature, split, cross-validation, metric, applicability domain, prediction. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Explainability, Prediction, dan Ekspor Model, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 8.12, aturan kontrol yang disarankan adalah: descriptor harus memiliki nilai/keadaan yang eksplisit; feature harus memiliki nilai/keadaan yang eksplisit; split harus memiliki nilai/keadaan yang eksplisit; cross-validation harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

8.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar descriptor, feature, split, cross-validation, metric, applicability domain. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 8.23. Spesifikasi fungsional Explainability, Prediction, dan Ekspor Model

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi descriptor dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk descriptor; simpan status, pesan, dan provenance.	Record descriptor, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi feature dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk feature; simpan status, pesan, dan provenance.	Record feature, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi split dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk split; simpan status, pesan, dan provenance.	Record split, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan cross-validation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk cross-validation; simpan status, pesan, dan provenance.	Record cross-validation, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis metric dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metric; simpan status, pesan, dan provenance.	Record metric, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi applicability domain dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk applicability domain; simpan status, pesan, dan provenance.	Record applicability domain, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor prediction dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk prediction; simpan status, pesan, dan provenance.	Record prediction, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

8.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 8.24. Matriks risiko Explainability, Prediction, dan Ekspor Model

Risiko	Penyebab	Mitigasi
Input tidak konsisten	descriptor salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	feature tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	split dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	cross-validation dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau metric tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	applicability domain menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser prediction berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

8.12.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

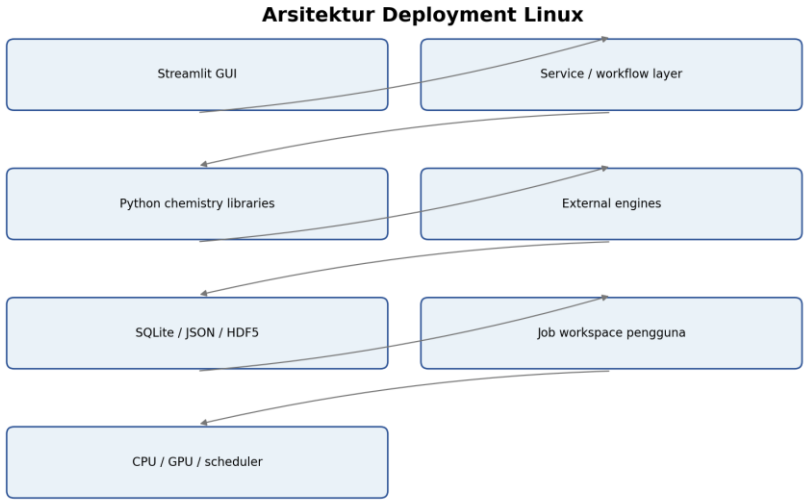
```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section8Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_8_12_explainability_prediction_dan_e(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="descriptor")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="feature")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk descriptor dan feature perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 8.4. Kerangka pendukung untuk Explainability, Prediction, dan Ekspor Model

Checklist implementasi:

- Input descriptor memiliki validator dan pesan error yang jelas.
- Parameter feature tersimpan dalam job specification.
- Raw output untuk split disimpan sebelum parsing.
- Hasil cross-validation mempunyai unit dan provenance.

- Kegagalan metric dapat dilacak melalui log.
- Perubahan applicability domain dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

8.13 Rangkuman Bab

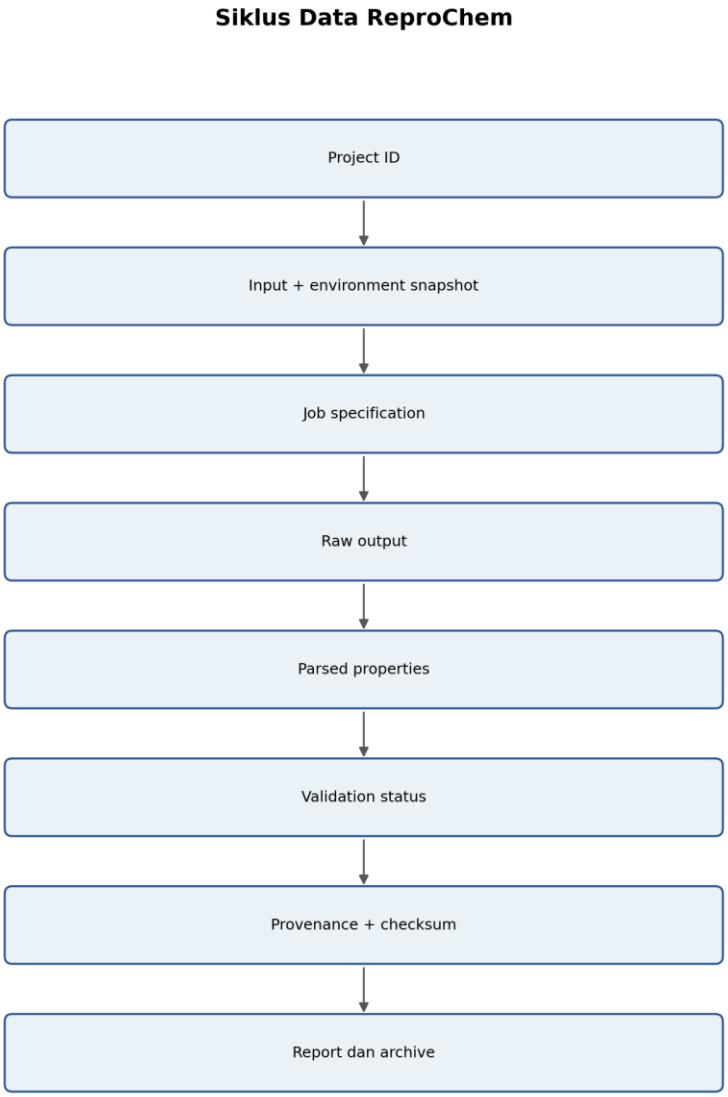
Bab 8 menempatkan QSAR/ADMET STUDIO: DESKRIPTOR, MACHINE LEARNING, DAN SCREENING sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 9 REPROCHEM: DATA, PROVENANCE, DAN REPRODUCIBLE COMPUTATIONAL RESEARCH

Bab ini berfokus pada mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Ekosistem teknis yang menjadi perhatian utama meliputi SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 9.1. Reprochem: Data, Provenance, Dan Reproducible Computational Research

9.1 Reproducibility sebagai Fitur Inti Software Ilmiah

Reproducibility sebagai Fitur Inti Software Ilmiah harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil.

Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Reproducibility sebagai Fitur Inti Software Ilmiah, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.1, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.1. Spesifikasi fungsional Reproducibility sebagai Fitur Inti Software Ilmiah

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.2. Matriks risiko Reproducibility sebagai Fitur Inti Software Ilmiah

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_1_reproducibility_sebagai_fitur_in(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.2 Project ID, Molecule ID, Calculation ID, dan Relasi Data

Project ID, Molecule ID, Calculation ID, dan Relasi Data harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Project ID, Molecule ID, Calculation ID, dan Relasi Data, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.2, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.3. Spesifikasi fungsional Project ID, Molecule ID, Calculation ID, dan Relasi Data

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.4. Matriks risiko Project ID, Molecule ID, Calculation ID, dan Relasi Data

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_2_project_id_molecule_id_calculati(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.

- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.3 Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity

Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.3, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.5. Spesifikasi fungsional Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.6. Matriks risiko Penyimpanan Koordinat, SMILES, Charge, dan Multiplicity

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_3_penyimpanan_koordinat_smiles_cha(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.4 Calculation Specification: Method, Basis, Solvent, dan Convergence

Calculation Specification: Method, Basis, Solvent, dan Convergence harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job

selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Calculation Specification: Method, Basis, Solvent, dan Convergence, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.4, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.7. Spesifikasi fungsional Calculation Specification: Method, Basis, Solvent, dan Convergence

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.8. Matriks risiko Calculation Specification: Method, Basis, Solvent, dan Convergence

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.4.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine,

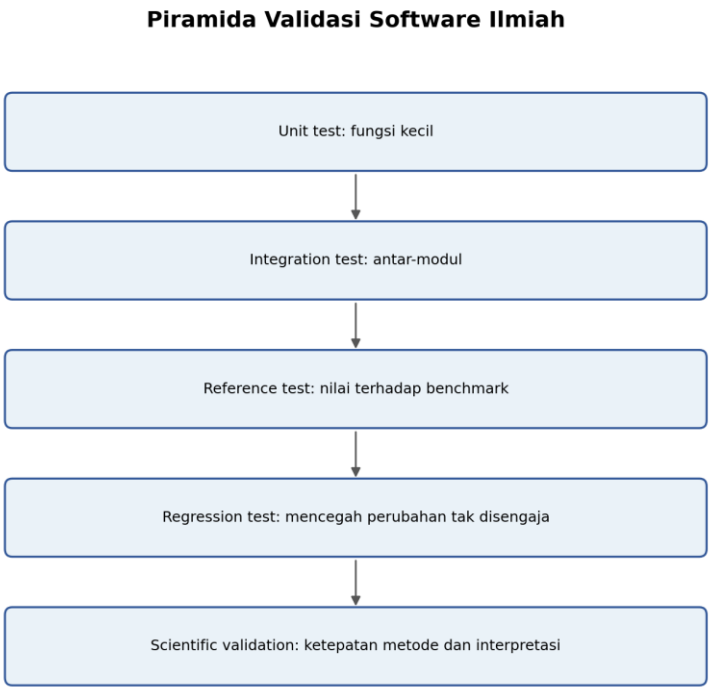
penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_4_calculation_specification_method(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 9.2. Kerangka pendukung untuk Calculation Specification: Method, Basis, Solvent, dan Convergence

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.5 Environment Snapshot: Versi Software, CPU, GPU, dan OS

Environment Snapshot: Versi Software, CPU, GPU, dan OS harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired

records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Environment Snapshot: Versi Software, CPU, GPU, dan OS, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.5, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.9. Spesifikasi fungsional Environment Snapshot: Versi Software, CPU, GPU, dan OS

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.10. Matriks risiko Environment Snapshot: Versi Software, CPU, GPU, dan OS

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_5_environment_snapshot_versi_softw(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.

- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.6 Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan

Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber.

Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.6, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.11. Spesifikasi fungsional Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.12. Matriks risiko Raw Output sebagai Bukti Primer dan Parsed Result sebagai Turunan

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_6_raw_output_sebagai_bukti_primer_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.7 Hash, Checksum, Deduplication, dan Integritas File

Hash, Checksum, Deduplication, dan Integritas File harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Hash, Checksum, Deduplication, dan Integritas File, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.7, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.13. Spesifikasi fungsional Hash, Checksum, Deduplication, dan Integritas File

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.14. Matriks risiko Hash, Checksum, Deduplication, dan Integritas File

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_7_hash_checksum_deduplication_dan_(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.8 Database Schema, Indexing, dan Query Hasil

Database Schema, Indexing, dan Query Hasil harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini,

pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Database Schema, Indexing, dan Query Hasil, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan

laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.8, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.15. Spesifikasi fungsional Database Schema, Indexing, dan Query Hasil

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.16. Matriks risiko Database Schema, Indexing, dan Query Hasil

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.8.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

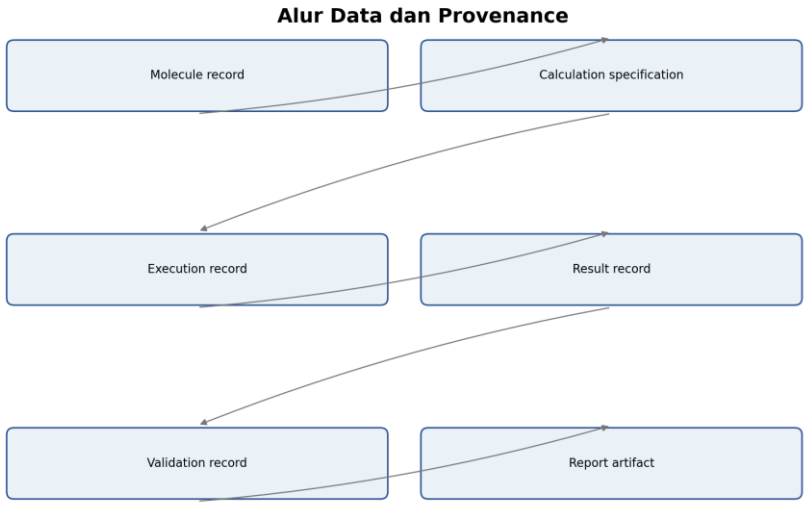
provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_8_database_schema_indexing_dan_que(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 9.3. Kerangka pendukung untuk Database Schema, Indexing, dan Query Hasil

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.9 Status Job, Error Log, Restart, dan Audit Trail

Status Job, Error Log, Restart, dan Audit Trail harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan

perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Status Job, Error Log, Restart, dan Audit Trail, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.9, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan

yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.17. Spesifikasi fungsional Status Job, Error Log, Restart, dan Audit Trail

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.18. Matriks risiko Status Job, Error Log, Restart, dan Audit Trail

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi

Risiko	Penyebab	Mitigasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_9_status_job_error_log_restart_dan(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
```

```
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.
- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.10 FAIR Data, Metadata, dan Ekspor Proyek

FAIR Data, Metadata, dan Ekspor Proyek harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur

tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk FAIR Data, Metadata, dan Ekspor Proyek, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.10, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu

membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.19. Spesifikasi fungsional FAIR Data, Metadata, dan Ekspor Proyek

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.20. Matriks risiko FAIR Data, Metadata, dan Ekspor Proyek

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_10_fair_data_metadata_dan_ekspor_p(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.11 Generator Laporan DOCX/PDF/CSV yang Reproducible

Generator Laporan DOCX/PDF/CSV yang Reproducible harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan provenance, version, environment, raw output, hash, database, report. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Generator Laporan DOCX/PDF/CSV yang Reproducible, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-

fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.11, aturan kontrol yang disarankan adalah: provenance harus memiliki nilai/keadaan yang eksplisit; version harus memiliki nilai/keadaan yang eksplisit; environment harus memiliki nilai/keadaan yang eksplisit; raw output harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar provenance, version, environment, raw output, hash, database. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.21. Spesifikasi fungsional Generator Laporan DOCX/PDF/CSV yang Reproducible

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis hash dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk hash; simpan status, pesan, dan provenance.	Record hash, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi database dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk database; simpan status, pesan, dan provenance.	Record database, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor report dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk report; simpan status, pesan, dan provenance.	Record report, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.22. Matriks risiko Generator Laporan DOCX/PDF/CSV yang Reproducible

Risiko	Penyebab	Mitigasi
Input tidak konsisten	provenance salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	version tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	environment dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	raw output dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau hash tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	database menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser report berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_11_generator_laporan_docx_pdf_csv_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="provenance")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="version")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk provenance dan version perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input provenance memiliki validator dan pesan error yang jelas.

- Parameter version tersimpan dalam job specification.
- Raw output untuk environment disimpan sebelum parsing.
- Hasil raw output mempunyai unit dan provenance.
- Kegagalan hash dapat dilacak melalui log.
- Perubahan database dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.12 Backup, Migrasi Versi, dan Retensi Data Riset

Backup, Migrasi Versi, dan Retensi Data Riset harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah mencatat seluruh konteks perhitungan sehingga setiap hasil dapat dilacak, diverifikasi, diulang, dan dibagikan dengan struktur data yang konsisten. Pada tingkat implementasi, modul memanfaatkan SQLite, JSON/QCSchema-inspired records, hashing, environment capture, versioning, project IDs, immutable raw output, report generation dan audit trail. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Backup, Migrasi Versi, dan Retensi Data Riset, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 9.12, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

9.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 9.23. Spesifikasi fungsional Backup, Migrasi Versi, dan Retensi Data Riset

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan provenance dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk provenance; simpan status, pesan, dan provenance.	Record provenance, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark environment dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk environment; simpan status, pesan, dan provenance.	Record environment, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi raw output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk raw output; simpan status, pesan, dan provenance.	Record raw output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

9.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 9.24. Matriks risiko Backup, Migrasi Versi, dan Retensi Data Riset

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

9.12.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

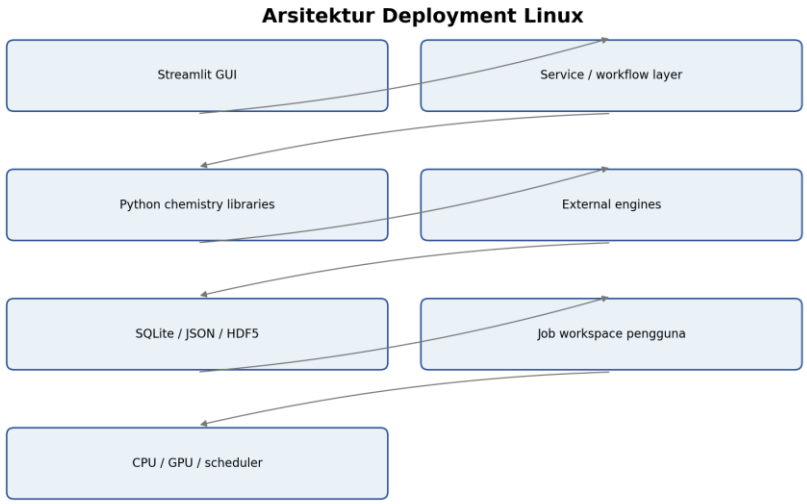
```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section9Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_9_12_backup_migrasi_versi_dan_retens(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 9.4. Kerangka pendukung untuk Backup, Migrasi Versi, dan Retensi Data Riset

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.

- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

9.13 Rangkuman Bab

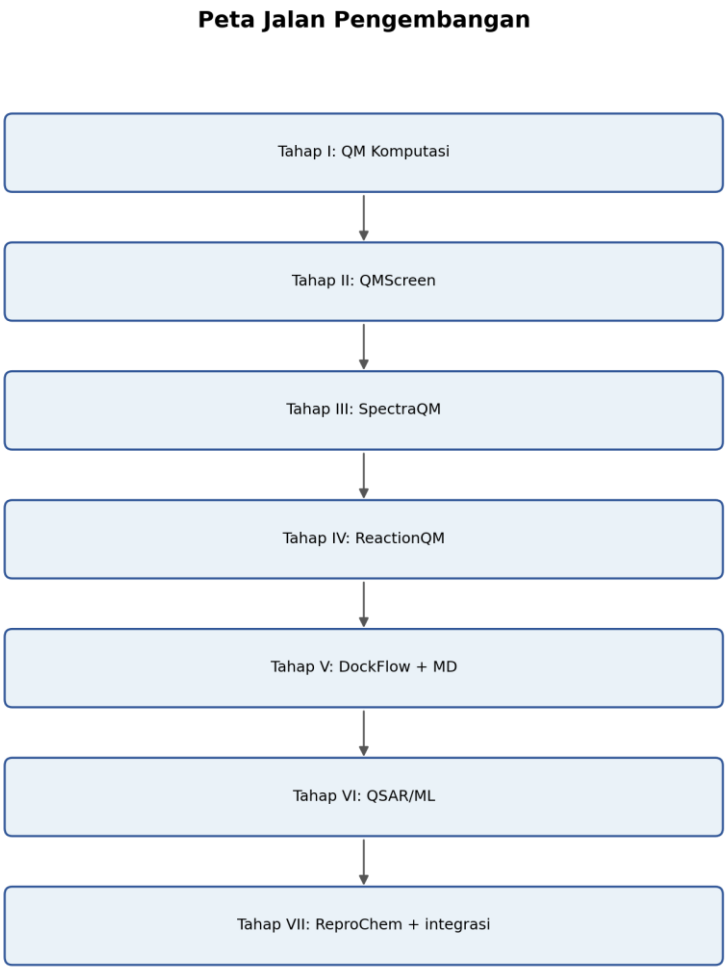
Bab 9 menempatkan REPROCHEM: DATA, PROVENANCE, DAN REPRODUCIBLE COMPUTATIONAL RESEARCH sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama, GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

BAB 10 INTEGRASI KOMPUTASI JISAMSU, IMPLEMENTASI, VALIDASI, DAN PETA JALAN

Bab ini berfokus pada menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Ekosistem teknis yang menjadi perhatian utama meliputi core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Seluruh pembahasan diarahkan pada software riset yang menyimpan konteks ilmiah, bukan hanya mengotomasi perintah. Karena itu, setiap fungsi dinilai dari empat sisi: kebenaran input, keterandalan eksekusi, kualitas interpretasi, dan kemampuan reproduksi hasil.

Prinsip desain yang digunakan sepanjang bab adalah pemisahan antara data, metode, eksekusi, hasil, dan presentasi. Struktur ini memberi fleksibilitas ketika library berubah versi atau ketika satu engine perlu diganti oleh engine lain. Di sisi pengguna, antarmuka tetap dapat dipertahankan; di sisi ilmiah, setiap kalkulasi tetap memiliki spesifikasi yang dapat ditelusuri dan diuji ulang.



Gambar 10.1. Integrasi Komputasi Jisamsu, Implementasi, Validasi, Dan Peta Jalan

10.1 Satu Core Engine dan Banyak Modul Antarmuka

Satu Core Engine dan Banyak Modul Antarmuka harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Satu Core Engine dan Banyak Modul Antarmuka, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.1, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki

nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.1.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.1. Spesifikasi fungsional Satu Core Engine dan Banyak Modul Antarmuka

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.1.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.2. Matriks risiko Satu Core Engine dan Banyak Modul Antarmuka

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.1.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_1_satu_core_engine_dan_banyak_mod(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.2 Kontrak Data Antar-Modul dan QCSchema-Inspired Interface

Kontrak Data Antar-Modul dan QCSchema-Inspired Interface harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Kontrak Data Antar-Modul dan QCSchema-Inspired Interface, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan,

sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.2, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.2.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.3. Spesifikasi fungsional Kontrak Data Antar-Modul dan QCSchema-Inspired Interface

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.2.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.4. Matriks risiko Kontrak Data Antar-Modul dan QCSchema-Inspired Interface

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.2.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_2_kontrak_data_antar_modul_dan_qc(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.3 Struktur Package Python dan Dependency Boundary

Struktur Package Python dan Dependency Boundary harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan

pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Struktur Package Python dan Dependency Boundary, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.3, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine.

Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.3.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.5. Spesifikasi fungsional Struktur Package Python dan Dependency Boundary

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan format data dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk format data; simpan status, pesan, dan provenance.	Record format data, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark validasi sintaks dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk validasi sintaks; simpan status, pesan, dan provenance.	Record validasi sintaks, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi unit dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk unit; simpan status, pesan, dan provenance.	Record unit, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.3.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.6. Matriks risiko Struktur Package Python dan Dependency Boundary

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi

Risiko	Penyebab	Mitigasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.3.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_3_struktur_package_python_dan_dep(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
```

```
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.4 Streamlit GUI, CLI, dan Service Layer

Streamlit GUI, CLI, dan Service Layer harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut

memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Streamlit GUI, CLI, dan Service Layer, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.4, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.4.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.7. Spesifikasi fungsional Streamlit GUI, CLI, dan Service Layer

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.4.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.8. Matriks risiko Streamlit GUI, CLI, dan Service Layer

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.4.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

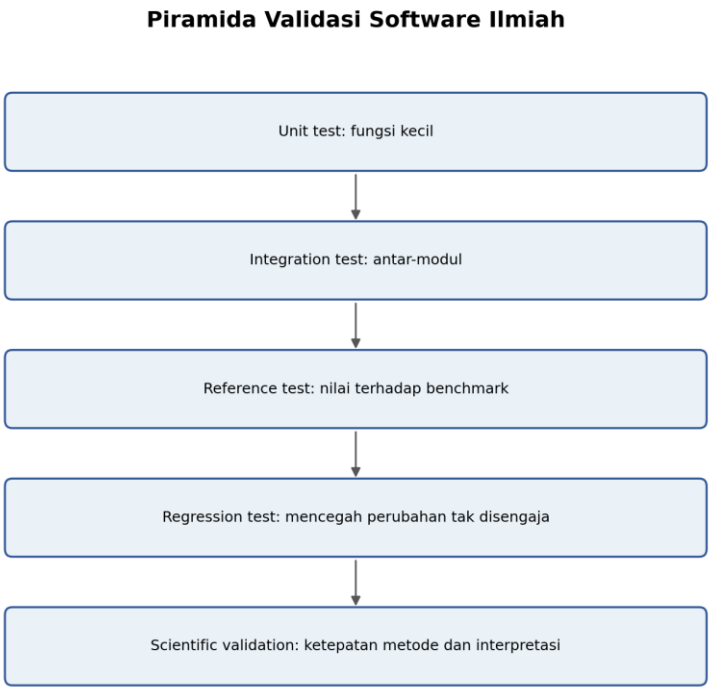
provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_4_streamlit_gui_cli_dan_service_l(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 10.2. Kerangka pendukung untuk Streamlit GUI, CLI, dan Service Layer

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.5 Workspace Pengguna, Permission Linux, dan Keamanan File

Workspace Pengguna, Permission Linux, dan Keamanan File harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface,

Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Workspace Pengguna, Permission Linux, dan Keamanan File, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti

menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.5, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.5.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.9. Spesifikasi fungsional Workspace Pengguna, Permission Linux, dan Keamanan File

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.5.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.10. Matriks risiko Workspace Pengguna, Permission Linux, dan Keamanan File

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.5.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_5_workspace_pengguna_permission_l(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.

- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.6 Packaging Conda/Pip dan Paket Debian

Packaging Conda/Pip dan Paket Debian harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Packaging Conda/Pip dan Paket Debian, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut,

perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.6, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.6.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan

memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.11. Spesifikasi fungsional Packaging Conda/Pip dan Paket Debian

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.6.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.12. Matriks risiko Packaging Conda/Pip dan Paket Debian

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema

Risiko	Penyebab	Mitigasi
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.6.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_6_packaging_conda_pip_dan_paket_d(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.7 Logging, Telemetri Lokal, dan Diagnostik Error

Logging, Telemetri Lokal, dan Diagnostik Error harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Logging, Telemetri Lokal, dan Diagnostik Error, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.7, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.7.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.13. Spesifikasi fungsional Logging, Telemetri Lokal, dan Diagnostik Error

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.7.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.14. Matriks risiko Logging, Telemetri Lokal, dan Diagnostik Error

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log

Risiko	Penyebab	Mitigasi
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.7.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_7_logging_telemetri_lokal_dan_dia(spec, workspace: Path):
```

```
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.8 Unit Test, Integration Test, Scientific Benchmark, dan Regression Test

Unit Test, Integration Test, Scientific Benchmark, dan Regression Test harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing,

documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Unit Test, Integration Test, Scientific Benchmark, dan Regression Test, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal,

dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.8, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.8.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.15. Spesifikasi fungsional Unit Test, Integration Test, Scientific Benchmark, dan Regression Test

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan reference calculation dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk reference calculation; simpan status, pesan, dan provenance.	Record reference calculation, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark benchmark set dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk benchmark set; simpan status, pesan, dan provenance.	Record benchmark set, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi toleransi numerik dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk toleransi numerik; simpan status, pesan, dan provenance.	Record toleransi numerik, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.8.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.16. Matriks risiko Unit Test, Integration Test, Scientific Benchmark, dan Regression Test

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.8.3 Sketsa Implementasi Python

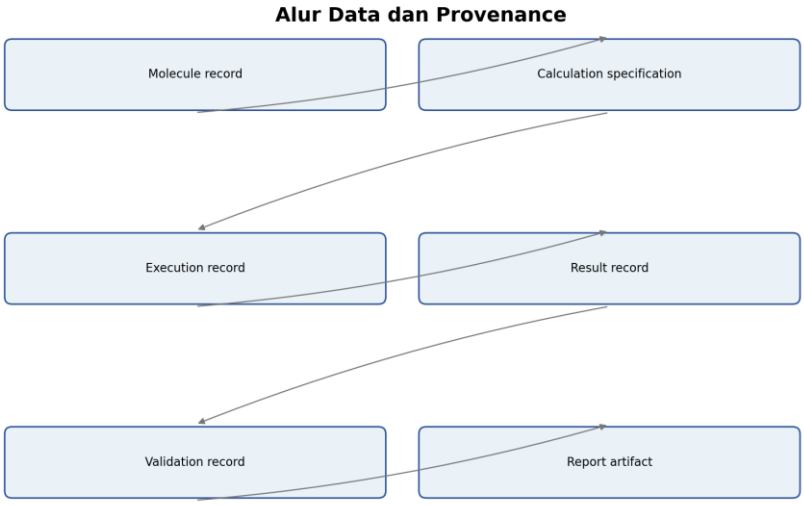
Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_8_unit_test_integration_test_scie(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 10.3. Kerangka pendukung untuk Unit Test, Integration Test, Scientific Benchmark, dan Regression Test

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.9 Dokumentasi Pengguna, Tutorial, dan Contoh Proyek

Dokumentasi Pengguna, Tutorial, dan Contoh Proyek harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job

selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Dokumentasi Pengguna, Tutorial, dan Contoh Proyek, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.9, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.9.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.17. Spesifikasi fungsional Dokumentasi Pengguna, Tutorial, dan Contoh Proyek

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.9.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.18. Matriks risiko Dokumentasi Pengguna, Tutorial, dan Contoh Proyek

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.9.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan

provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_9_dokumentasi_pengguna_tutorial_d(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.10 Versioning, Release Notes, dan Migrasi Database

Versioning, Release Notes, dan Migrasi Database harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Versioning, Release Notes, dan Migrasi Database, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan,

sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.10, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.10.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.19. Spesifikasi fungsional Versioning, Release Notes, dan Migrasi Database

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.10.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.20. Matriks risiko Versioning, Release Notes, dan Migrasi Database

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback

mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.10.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_10_versioning_release_notes_dan_m(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.

- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.11 Prioritas Implementasi Bertahap 2026 dan Sesudahnya

Prioritas Implementasi Bertahap 2026 dan Sesudahnya harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Prioritas Implementasi Bertahap 2026 dan Sesudahnya, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.11, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.11.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.21. Spesifikasi fungsional Prioritas Implementasi Bertahap 2026 dan Sesudahnya

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.11.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.22. Matriks risiko Prioritas Implementasi Bertahap 2026 dan Sesudahnya

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi

Risiko	Penyebab	Mitigasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.11.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine, penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_11_prioritas_implementasi_bertaha(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
```

```
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.12 Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan

Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan harus dipahami sebagai komponen ilmiah, bukan sekadar elemen antarmuka. Fokusnya adalah menyatukan modul menjadi platform komputasi kimia berbasis satu core engine dan project database serta menetapkan strategi implementasi, packaging, pengujian, dokumentasi, dan pengembangan berkelanjutan. Pada tingkat implementasi, modul memanfaatkan core Python package, plugin interface, Streamlit, CLI, SQLite, environment isolation, Debian packaging, CI/testing, documentation dan governance versi. Setiap keputusan desain harus menjaga keterlacakan antara input, parameter komputasi, proses eksekusi, dan hasil. Dengan pendekatan ini, pengguna dapat menilai bukan hanya apakah sebuah job selesai, tetapi juga apakah job tersebut memenuhi prasyarat ilmiah, dapat diulang, dan menghasilkan data yang layak dibandingkan dengan perhitungan lain.

Kebutuhan utama pada bagian ini berkaitan dengan schema, primary key, foreign key, metadata, checksum, version, serialization. Kesalahan pada salah satu unsur tersebut dapat merambat ke seluruh workflow. Karena itu, validasi tidak ditempatkan hanya pada akhir perhitungan, melainkan dilakukan sejak data masuk, sebelum engine dipanggil, setelah engine menghasilkan raw output, dan setelah parser membentuk data terstruktur. Pola validasi berlapis penting untuk mencegah hasil numerik yang tampak rapi tetapi berasal dari spesifikasi yang tidak sesuai.

Dari sudut rekayasa software, implementasi sebaiknya memisahkan model data, scientific service, executor, parser, analyzer, dan presentation layer. Pemisahan ini memungkinkan engine diganti tanpa mengubah keseluruhan GUI dan memungkinkan satu hasil dipakai kembali oleh modul lain. Kontrak antar-lapisan perlu sederhana: struktur molekul, spesifikasi job, lokasi workspace, status eksekusi, pesan diagnostik, dan objek hasil. Struktur seperti ini juga memudahkan pengujian otomatis karena fungsi ilmiah dapat diuji tanpa harus menjalankan antarmuka Streamlit.

Aspek reproducibility harus diwujudkan secara konkret melalui pencatatan parameter. Untuk Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan, catatan minimal mencakup identitas molekul atau sistem, koordinat awal, satuan, metode, opsi numerik, versi library, waktu eksekusi, dan status validasi. Raw output tidak seharusnya ditimpa oleh hasil parsing. Sebaliknya, parser menghasilkan record turunan yang merujuk kembali ke file sumber. Dengan mekanisme tersebut, perubahan parser pada versi berikutnya dapat dievaluasi tanpa kehilangan bukti perhitungan awal.

Efisiensi tidak identik dengan memilih metode paling cepat. Strategi yang baik menggunakan tingkat komputasi yang sesuai dengan pertanyaan ilmiah. Tahap murah dipakai untuk menyaring konfigurasi yang jelas tidak menjanjikan, sedangkan metode yang lebih mahal dipakai pada kandidat terpilih. Prinsip multi-fidelity ini menurunkan beban CPU dan memori tanpa menghilangkan transparansi keputusan, asalkan kriteria perpindahan antar-tahap disimpan sebagai bagian dari proyek.

Keluaran dari bagian ini harus bersifat machine-readable dan human-readable sekaligus. Data numerik idealnya tersimpan dalam record terstruktur untuk analisis lanjutan, sedangkan tabel, grafik, dan narasi ringkas membantu peneliti menilai hasil secara cepat. Setiap angka penting perlu memiliki unit, metode asal, dan status validasi. Dengan demikian, export CSV/XLSX, database internal, dan laporan DOCX/PDF merupakan representasi berbeda dari sumber data yang sama, bukan salinan yang dikelola secara terpisah.

Dalam spesifikasi teknis bagian 10.12, aturan kontrol yang disarankan adalah: schema harus memiliki nilai/keadaan yang eksplisit; primary key harus memiliki

nilai/keadaan yang eksplisit; foreign key harus memiliki nilai/keadaan yang eksplisit; metadata harus memiliki nilai/keadaan yang eksplisit. Sistem perlu membedakan error yang menghentikan job, warning yang masih mengizinkan job diteruskan, dan informasi yang hanya dicatat untuk audit. Pemisahan tingkat keparahan ini penting agar pengguna tidak terbiasa mengabaikan seluruh pesan. Setiap error sebaiknya memiliki kode stabil, deskripsi singkat, kemungkinan penyebab, dan tindakan korektif yang aman sehingga dokumentasi dan antarmuka dapat merujuk pada kode yang sama.

Skenario uji untuk bagian ini harus mewakili data normal, batas, dan gagal. Data normal menguji workflow utama; data batas menguji kondisi seperti ukuran sistem besar, nilai parameter di ujung rentang, atau struktur yang hampir degenerat; sedangkan data gagal memastikan software berhenti dengan pesan yang dapat ditindaklanjuti. Hasil uji dicatat bersama toleransi dan versi engine. Dengan pola ini, perubahan kode dapat dinilai melalui regression test, bukan hanya melalui pengamatan visual bahwa tombol GUI masih dapat diklik.

10.12.1 Spesifikasi Fungsional dan Scientific Guardrails

Scientific guardrails untuk bagian ini dibangun di sekitar schema, primary key, foreign key, metadata, checksum, version. Guardrail bukan pembatas yang menyulitkan pengguna, melainkan mekanisme untuk mencegah kombinasi parameter yang jelas tidak valid, memperingatkan kondisi meragukan, dan memastikan bahwa keputusan otomatis tetap dapat dilacak. Nilai default harus ditulis ke dalam job specification sehingga hasil tidak bergantung pada keadaan GUI yang tidak terekam.

Tabel 10.23. Spesifikasi fungsional Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan

Aspek	Kebutuhan	Quality control	Keluaran
Validasi	Validasi schema dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk schema; simpan status, pesan, dan provenance.	Record schema, status QC, dan metadata yang dapat ditelusuri.
Normalisasi	Normalisasi primary key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk primary key; simpan status, pesan, dan provenance.	Record primary key, status QC, dan metadata yang dapat ditelusuri.

Aspek	Kebutuhan	Quality control	Keluaran
Eksekusi	Eksekusi foreign key dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk foreign key; simpan status, pesan, dan provenance.	Record foreign key, status QC, dan metadata yang dapat ditelusuri.
Pencatatan	Pencatatan metadata dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk metadata; simpan status, pesan, dan provenance.	Record metadata, status QC, dan metadata yang dapat ditelusuri.
Analisis	Analisis checksum dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk checksum; simpan status, pesan, dan provenance.	Record checksum, status QC, dan metadata yang dapat ditelusuri.
Visualisasi	Visualisasi version dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk version; simpan status, pesan, dan provenance.	Record version, status QC, dan metadata yang dapat ditelusuri.
Ekspor	Ekspor serialization dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk serialization; simpan status, pesan, dan provenance.	Record serialization, status QC, dan metadata yang dapat ditelusuri.
Pemulihan	Pemulihan workflow dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk workflow; simpan status, pesan, dan provenance.	Record workflow, status QC, dan metadata yang dapat ditelusuri.
Benchmark	Benchmark parameter dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk parameter; simpan status, pesan, dan provenance.	Record parameter, status QC, dan metadata yang dapat ditelusuri.
Dokumentasi	Dokumentasi output dengan aturan eksplisit dan nilai default yang terdokumentasi.	Uji positif/negatif untuk output; simpan status, pesan, dan provenance.	Record output, status QC, dan metadata yang dapat ditelusuri.

Tabel di atas dapat diterjemahkan langsung menjadi acceptance criteria. Sebuah fitur dianggap selesai bukan ketika tombolnya telah tersedia, melainkan ketika input valid dapat diproses, input tidak valid ditolak secara informatif, metadata tersimpan, hasil dapat diekspor, dan hasil numerik lulus uji acuan dalam toleransi

yang ditetapkan. Acceptance criteria semacam ini menghubungkan kebutuhan pengguna dengan pengujian software.

10.12.2 Risiko, Failure Mode, dan Strategi Mitigasi

Tabel 10.24. Matriks risiko Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan

Risiko	Penyebab	Mitigasi
Input tidak konsisten	schema salah atau tidak lengkap	Validasi sebelum job dan pesan koreksi spesifik
Kegagalan numerik	primary key tidak konvergen atau tidak stabil	Fallback terkontrol, retry terbatas, simpan log
Salah interpretasi	foreign key dibaca tanpa konteks metode	Tampilkan unit, metode, asumsi, dan status validasi
Duplikasi komputasi	metadata dihitung ulang tanpa perubahan spesifikasi	Hash spesifikasi dan cache hasil tervalidasi
Kehilangan provenance	versi atau checksum tidak tercatat	Snapshot environment dan immutable raw output
Beban sumber daya	version menghabiskan CPU/memori	Resource limit, queue, estimasi ukuran, stop aman
Perubahan versi	hasil parser serialization berubah antar-rilis	Regression test dan migrasi schema
Ekspor tidak sinkron	laporan berbeda dengan database	Generate semua output dari record hasil yang sama

Failure mode perlu dicatat dalam log yang mudah dibaca manusia sekaligus dapat diproses mesin. Pesan error sebaiknya tidak hanya menampilkan traceback mentah. Traceback tetap disimpan untuk diagnosis pengembang, sedangkan pengguna memperoleh ringkasan konteks dan tindakan yang dapat dilakukan, misalnya memeriksa charge/multiplicity, mengubah convergence threshold, memperbaiki struktur, mengurangi ukuran job, atau menjalankan kembali dari checkpoint yang valid.

10.12.3 Sketsa Implementasi Python

Sketsa berikut bersifat arsitektural. Nama fungsi menunjukkan boundary yang sebaiknya dipertahankan: validasi, registrasi job, pemanggilan engine,

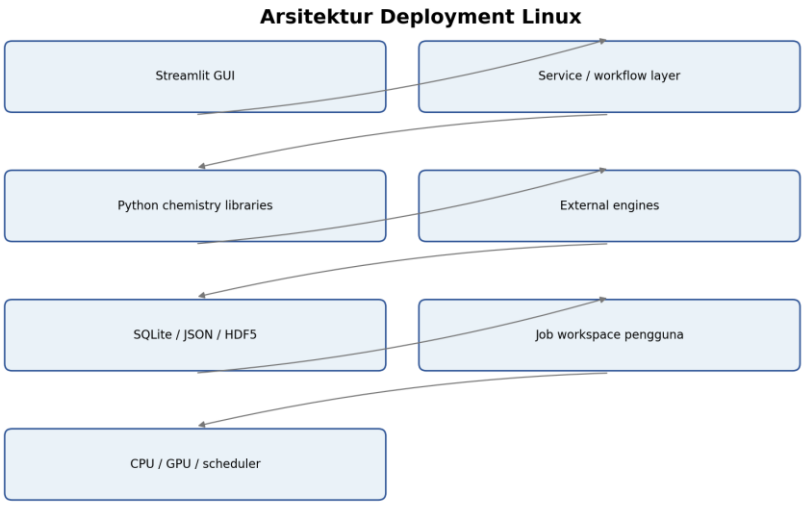
penyimpanan raw output, parsing, quality control, penyimpanan hasil, dan provenance. Detail engine dapat berbeda, tetapi aliran tanggung jawab tetap konsisten di seluruh modul.

```
from dataclasses import dataclass
from pathlib import Path

@dataclass
class Section10Spec:
    project_id: str
    molecule_id: str
    charge: int = 0
    multiplicity: int = 1
    method: str = "auto"

def run_10_12_arah_platform_komputasi_kimia_(spec, workspace: Path):
    validate_structure(spec)
    validate_charge_multiplicity(spec)
    job_id = register_job(spec, workspace)
    try:
        context = prepare_context(spec, focus="schema")
        raw = execute_engine(context)
        save_raw_output(job_id, raw)
        result = parse_result(raw)
        result = quality_control(result, key="primary key")
        store_result(job_id, result)
        return result
    except Exception as exc:
        register_failure(job_id, exc)
        raise
    finally:
        write_provenance(job_id)
```

Pada implementasi produksi, fungsi untuk schema dan primary key perlu memiliki unit test tersendiri. Objek hasil sebaiknya tidak bergantung pada komponen GUI sehingga dapat dipakai oleh CLI, batch runner, notebook, maupun antarmuka web. Dengan desain ini, sebuah perhitungan yang sama dapat dijalankan melalui beberapa antarmuka tetapi tetap menghasilkan record proyek dengan schema identik.



Gambar 10.4. Kerangka pendukung untuk Arah Platform Komputasi Kimia Terpadu untuk Riset dan Pendidikan

Checklist implementasi:

- Input schema memiliki validator dan pesan error yang jelas.
- Parameter primary key tersimpan dalam job specification.
- Raw output untuk foreign key disimpan sebelum parsing.
- Hasil metadata mempunyai unit dan provenance.
- Kegagalan checksum dapat dilacak melalui log.
- Perubahan version dilindungi regression test.
- Ekspor laporan dibangkitkan dari record hasil yang sama dengan database.
- Dokumentasi menyebut asumsi ilmiah dan batas penggunaan.

10.13 Rangkuman Bab

Bab 10 menempatkan INTEGRASI KOMPUTASI JISAMSU, IMPLEMENTASI, VALIDASI, DAN PETA JALAN sebagai komponen dari ekosistem yang lebih besar. Fokus utama tidak hanya pada keberhasilan eksekusi, melainkan pada validasi, provenance, failure recovery, dan interoperabilitas. Desain yang konsisten membuat modul dapat menggunakan project database yang sama sehingga hasil satu tahap tidak perlu dimasukkan ulang pada tahap berikutnya.

Keputusan implementasi yang paling penting adalah menjaga pemisahan antara scientific logic dan presentation layer. Dengan pola tersebut, pengujian dapat dilakukan secara otomatis, workflow batch dapat berjalan tanpa GUI, dan perubahan tampilan tidak memengaruhi validitas kalkulasi. Pada saat yang sama,

GUI tetap berperan penting untuk menampilkan parameter, progress, log, visualisasi, dan peringatan yang membantu pengguna melakukan kontrol ilmiah.

10.14 LATIHAN INTEGRATIF DAN PEMBAHASAN

1. Mengapa software kimia komputasi yang hanya bertindak sebagai kalkulator energi belum cukup untuk riset modern?

Pembahasan: Karena riset modern memerlukan rangkaian kegiatan yang lebih luas: preparasi struktur, validasi input, batch screening, pengelolaan sumber daya, analisis keluaran, penyimpanan provenance, integrasi dengan docking/MD/ML, dan pelaporan. Nilai ilmiah software terletak pada kemampuan menjaga konteks dari awal sampai akhir, bukan hanya menghasilkan satu angka energi.

2. Jelaskan perbedaan peran MolPrep dan QMScreen.

Pembahasan: MolPrep berfokus pada kualitas dan konsistensi struktur sebelum kalkulasi: sanitization, tautomer/protonation state, stereokimia, konformer, clustering, dan seleksi geometri. QMScreen menggunakan struktur yang telah siap untuk menjalankan workflow high-throughput seperti prescreening xTB, optimasi DFT, frequency, descriptor kuantum, dan ranking.

3. Mengapa GFN2-xTB sesuai ditempatkan sebagai prescreening sebelum DFT?

Pembahasan: GFN2-xTB dirancang sebagai metode tight-binding berbiaya rendah yang efisien untuk struktur dan interaksi. Dalam workflow multi-fidelity, metode ini dapat mengurangi jumlah konformer/kandidat yang perlu dihitung dengan DFT. Hasil prescreening tidak menggantikan validasi DFT ketika pertanyaan penelitian memerlukan tingkat teori lebih tinggi.

4. Apa fungsi frequency calculation setelah optimasi geometri?

Pembahasan: Frequency calculation membantu memeriksa karakter titik stasioner dan memberi informasi vibrasi serta kontribusi termokimia. Minimum yang benar secara lokal diharapkan tidak memiliki frekuensi imajiner, sedangkan transition state orde pertama secara ideal memiliki satu mode imajiner yang berkaitan dengan koordinat reaksi.

5. Mengapa satu frekuensi imajiner belum otomatis membuktikan transition state yang benar?

Pembahasan: Mode imajiner harus diperiksa apakah gerak atomnya sesuai bond forming/breaking yang diharapkan, dan konektivitas harus diverifikasi, misalnya dengan IRC dua arah. Tanpa pemeriksaan tersebut, titik stasioner dapat menghubungkan minimum yang berbeda dari mekanisme yang dimaksud.

6. Apa keuntungan memisahkan IR dan thermochemistry dalam desain SpectraQM?

Pembahasan: IR berfokus pada frekuensi vibrasi, intensitas, scaling, broadening, assignment, dan perbandingan spektrum. Thermochemistry menggunakan informasi vibrasi dan energi untuk besaran termodinamika. Keduanya berhubungan tetapi memiliki tujuan analitis berbeda; pemisahan modul menghindari kebingungan antarmuka dan interpretasi.

7. Mengapa docking score tidak boleh diperlakukan sebagai afinitas eksperimen absolut?

Pembahasan: Docking menggunakan scoring function dan sampling yang menyederhanakan banyak aspek fisika sistem. Skor berguna untuk membandingkan pose atau melakukan prioritisasi dalam protokol tervalidasi, tetapi tidak identik dengan energi bebas pengikatan eksperimen. Karena itu perlu redocking, pemeriksaan pose, interaction analysis, dan bila sesuai tahap lanjutan seperti MD atau metode energi bebas.

8. Apa tujuan redocking dalam DockFlow?

Pembahasan: Redocking menguji apakah protokol mampu mereproduksi pose ligan yang diketahui dari struktur kompleks. RMSD antara pose hasil docking dan pose referensi menjadi salah satu indikator kinerja pose prediction. Hasil redocking juga membantu mengevaluasi grid, protonasi, parameter, dan search settings.

9. Mengapa checkpoint penting pada simulasi MD?

Pembahasan: MD dapat berjalan lama dan rentan terhadap interupsi sistem. Checkpoint menyimpan keadaan yang diperlukan untuk melanjutkan simulasi sehingga kehilangan waktu komputasi dapat diminimalkan. Resume harus tetap menyimpan provenance agar segmen trajectory dapat dilacak dan digabung secara benar.

10. Bedakan RMSD dan RMSF pada analisis MD.

Pembahasan: RMSD menilai deviasi struktur secara keseluruhan terhadap struktur referensi sepanjang waktu, sedangkan RMSF menilai besar fluktuasi tiap atom/residu terhadap posisi rata-ratanya. Keduanya menjawab pertanyaan berbeda dan perlu dibaca bersama dengan parameter lain seperti radius of gyration, SASA, kontak, dan kestabilan observabel.

11. Mengapa charge dan multiplicity harus menjadi input eksplisit pada software QM?

Pembahasan: Charge dan multiplicity menentukan jumlah elektron dan keadaan spin yang digunakan engine electronic structure. Kesalahan pada keduanya dapat

menyebabkan kegagalan SCF atau, lebih berbahaya, menghasilkan keadaan elektronik yang bukan sistem yang dimaksud. Karena itu software perlu menyediakan nilai default yang transparan dan kemampuan koreksi langsung.

12. Bagaimana mencegah kalkulasi duplikat pada QMScreen?

Pembahasan: Bangun hash dari spesifikasi canonical yang mencakup struktur, charge, multiplicity, metode, basis, opsi numerik, dan parameter relevan. Sebelum job dijalankan, database diperiksa untuk hasil identik yang sudah tervalidasi. Jika ada, hasil dapat dipakai kembali dengan tetap mencatat provenance penggunaan ulang.

13. Mengapa raw output harus disimpan walaupun hasil sudah diparsing?

Pembahasan: Parser dapat memiliki bug atau berubah antar-versi. Raw output adalah bukti primer dari engine dan memungkinkan parsing ulang serta audit. Parsed result adalah representasi terstruktur yang memudahkan analisis, tetapi tidak seharusnya menggantikan sumber primer.

14. Apa yang dimaksud data leakage dalam QSAR dan bagaimana mencegahnya?

Pembahasan: Data leakage terjadi ketika informasi dari test set atau target secara tidak semestinya memengaruhi training atau preprocessing. Pencegahannya mencakup split sebelum fitting transformasi, penggunaan pipeline, feature selection di dalam cross-validation, pengelompokan scaffold bila relevan, dan pemisahan set eksternal untuk evaluasi akhir.

15. Mengapa R^2 saja tidak cukup untuk menilai model QSAR?

Pembahasan: R^2 dapat tinggi pada training tetapi tidak menunjukkan kemampuan generalisasi. Evaluasi perlu mempertimbangkan cross-validation, Q^2 atau metrik validasi, RMSE/MAE untuk regresi, metrik klasifikasi bila sesuai, Y-randomization, applicability domain, serta performa pada test set yang tidak dipakai membangun model.

16. Apa fungsi applicability domain?

Pembahasan: Applicability domain menjelaskan wilayah ruang descriptor/kimia tempat prediksi model dianggap didukung oleh data training. Prediksi di luar domain perlu diberi peringatan karena ketidakpastiannya lebih besar. Software sebaiknya menyimpan indikator domain bersama setiap prediksi.

17. Mengapa satu core engine lebih baik daripada sembilan aplikasi yang benar-benar terpisah?

Pembahasan: Core engine bersama menghindari duplikasi kode untuk molecule model, job handling, database, logging, export, dan validation. Modul dapat berbagi record dan workflow sehingga data dari satu tahap langsung digunakan

tahap lain. Pemeliharaan juga lebih mudah karena perbaikan pada fungsi inti berlaku untuk seluruh modul.

18. Apa perbedaan unit test, integration test, regression test, dan scientific validation?

Pembahasan: Unit test menguji fungsi kecil secara terisolasi; integration test menguji komunikasi antar-komponen; regression test memastikan perilaku yang sudah benar tidak berubah tanpa sengaja; scientific validation membandingkan metode dan hasil terhadap acuan ilmiah untuk memastikan software bukan hanya benar secara pemrograman tetapi juga layak untuk tujuan riset.

19. Bagaimana Linux permission harus ditangani untuk workspace aplikasi yang dipasang melalui paket Debian?

Pembahasan: Aplikasi tidak seharusnya menulis hasil runtime ke direktori instalasi seperti /opt yang biasanya tidak dapat ditulis user biasa. Kode program dapat berada di /opt atau lokasi sistem, sedangkan workspace/run data diletakkan di home user, XDG data directory, atau lokasi yang secara eksplisit writable. Ini juga memisahkan file aplikasi dari data penelitian.

20. Susun urutan prioritas implementasi platform berdasarkan sumber dan jelaskan logikanya.

Pembahasan: Urutan yang rasional adalah mempertahankan QM Komputasi sebagai fondasi, kemudian QMScreen untuk throughput, SpectraQM untuk analisis, ReactionQM untuk mekanisme, DockFlow dan MD untuk drug discovery/biokomputasi, QSAR/ML untuk integrasi data dan prediksi, lalu ReproChem sebagai lapisan integrasi provenance. Urutan dapat disesuaikan sumber daya, tetapi dependensi data antar-modul sebaiknya tetap diperhatikan.

10.15 SOAL PENGEMBANGAN DAN PROYEK

- 1. Merancang schema SQLite untuk molecule, calculation, result, file, dan provenance serta menjelaskan primary/foreign key.
- 2. Membuat prototype QMScreen yang menerima CSV SMILES dan melakukan validasi, deduplication, dan queue tanpa menjalankan DFT.
- 3. Membuat parser generik untuk membaca status konvergensi, energi, dan pesan error dari dua contoh engine berbeda.
- 4. Merancang benchmark set minimal untuk menguji optimasi geometri dan HOMO-LUMO pada software QM.
- 5. Merancang workflow transition-state yang memaksa pemeriksaan mode imajiner dan IRC sebelum status TS dinyatakan valid.

- 6. Membuat dashboard SpectraQM yang menyimpan data puncak sebagai tabel dan menggambar spektrum dari data yang sama.
- 7. Merancang protokol redocking yang mencatat grid, exhaustiveness, random seed, RMSD, dan pose referensi.
- 8. Membuat workflow OpenMM yang memiliki checkpoint periodik dan dapat melanjutkan simulasi tanpa menimpa trajectory lama.
- 9. Membuat pipeline QSAR dengan train/test split, preprocessing di dalam pipeline, cross-validation, dan applicability-domain warning.
- 10. Merancang format project export yang dapat dipindahkan ke komputer lain dan tetap mempertahankan raw output, metadata, hash, dan versi environment.

GLOSARIUM

A

Applicability Domain: Wilayah ruang kimia/descriptor tempat prediksi model QSAR dianggap memiliki dukungan data yang memadai.

Atom Mapping: Pemetaan identitas atom antara reaktan dan produk untuk mengikuti perubahan ikatan dalam reaksi.

Audit Trail: Catatan kronologis perubahan status, eksekusi, dan tindakan terhadap sebuah job atau proyek.

B

Basis Set: Kumpulan fungsi basis untuk merepresentasikan orbital molekul pada perhitungan electronic structure.

Batch Calculation: Eksekusi banyak job dengan spesifikasi terstruktur tanpa memasukkan data satu per satu.

Benchmark: Kumpulan kasus acuan untuk mengukur akurasi, stabilitas, atau kinerja software/metode.

C

Cache: Penyimpanan hasil yang dapat digunakan kembali ketika spesifikasi perhitungan identik.

Charge: Muatan total sistem yang menentukan jumlah elektron pada kalkulasi kuantum.

Checkpoint: Keadaan tersimpan yang memungkinkan proses panjang dilanjutkan setelah interupsi.

Conformer: Konformasi tiga dimensi berbeda dari molekul dengan konektivitas sama.

D

Data Leakage: Masuknya informasi test/target secara tidak sah ke proses training atau preprocessing model.

Descriptor: Representasi numerik sifat atau struktur molekul untuk analisis statistik/ML.

DFT: Density Functional Theory, metode electronic structure berbasis kerapatan elektron.

Docking: Prediksi pose dan skor interaksi ligan pada situs pengikatan reseptor.

E

Electronic Structure: Deskripsi keadaan elektron dalam molekul atau material.

Ensemble: Kondisi termodinamika simulasi MD, misalnya NVT atau NPT.

Exhaustiveness: Parameter pencarian pada AutoDock Vina yang memengaruhi intensitas eksplorasi pose.

Executor: Lapisan software yang menjalankan engine komputasi berdasarkan job specification.

F

FAIR Data: Prinsip data Findable, Accessible, Interoperable, dan Reusable.

Failure Recovery: Mekanisme pemulihan job dari kegagalan tanpa kehilangan konteks dan data yang valid.

Feature Selection: Pemilihan subset descriptor untuk membangun model yang lebih sederhana dan stabil.

Force Field: Fungsi energi potensial beserta parameter yang digunakan dalam mekanika molekuler/MD.

G

Geometry Optimization: Proses mencari geometri dengan gradien energi yang memenuhi kriteria konvergensi.

GFN2-xTB: Metode extended tight-binding yang efisien untuk struktur, energi, dan eksplorasi konformasi.

GPU: Graphics Processing Unit, perangkat komputasi paralel yang dapat mempercepat jenis kalkulasi tertentu.

Grid: Ruang pencarian atau representasi diskret yang digunakan pada docking/analisis spasial.

H

Hash: Sidik digital dari data atau spesifikasi untuk pemeriksaan integritas dan deduplication.

Hessian: Matriks turunan kedua energi terhadap koordinat; digunakan dalam analisis frekuensi dan pencarian TS.

HOMO: Highest Occupied Molecular Orbital.

Hydrogen Bond: Interaksi nonkovalen terarah yang sering dianalisis pada docking dan MD.

I

InChI: International Chemical Identifier, representasi teks terstandar untuk struktur kimia.

Integrator: Algoritma numerik untuk memajukan posisi/kecepatan sistem sepanjang simulasi MD.

Interaction Fingerprint: Representasi diskret pola interaksi ligan-residu.

IRC: Intrinsic Reaction Coordinate, lintasan yang menghubungkan transition state dengan minimum di kedua arah.

J

Job: Unit kerja komputasi dengan input, spesifikasi, status, workspace, output, dan provenance.

Job Queue: Antrean job yang mengatur urutan serta alokasi sumber daya eksekusi.

JSON: Format data teks yang umum dipakai untuk konfigurasi, schema, dan pertukaran record.

Justifikasi Metode: Alasan ilmiah pemilihan metode berdasarkan sifat sistem dan pertanyaan penelitian.

K

Kalkulasi Multi-Fidelity: Strategi menggabungkan metode murah dan mahal secara bertingkat.

Kandidat: Molekul/struktur yang dinilai dalam screening.

Konvergensi: Kondisi ketika iterasi numerik memenuhi kriteria perubahan yang ditetapkan.

Koreksi Dispersi: Tambahan atau model untuk menangani interaksi dispersi yang kurang ditangkap oleh pendekatan tertentu.

L

Ligand: Molekul yang berinteraksi dengan target/reseptor dalam konteks docking atau simulasi.

Log: Catatan pesan proses, status, warning, dan error selama eksekusi.

LUMO: Lowest Unoccupied Molecular Orbital.

Löwdin Charge: Skema analisis populasi untuk menurunkan muatan atom dari fungsi gelombang/basis.

M

Machine Learning: Metode pemodelan berbasis data untuk klasifikasi, regresi, atau prediksi.

MEP: Molecular Electrostatic Potential, peta potensial elektrostatik di sekitar molekul.

Metadata: Data yang menjelaskan asal, parameter, versi, unit, dan konteks sebuah hasil.

Multiplicity: Besaran 2S+1 yang menyatakan keadaan spin total sistem.

N

NMR Shielding: Besaran komputasi yang berkaitan dengan perlindungan magnetik inti dan chemical shift.

NPT: Ensemble dengan jumlah partikel, tekanan, dan suhu dikontrol.

NVT: Ensemble dengan jumlah partikel, volume, dan suhu dikontrol.

Normal Mode: Pola gerak kolektif atom pada analisis vibrasi harmonik.

O

OpenMM: Toolkit simulasi molecular dynamics dengan Python API dan dukungan akselerasi perangkat keras.

Orbital Molekul: Fungsi satu-elektron yang digunakan untuk menafsirkan struktur elektronik.

Oscillator Strength: Ukuran intensitas transisi elektronik yang penting pada simulasi UV-Vis.

Output Parser: Komponen yang mengubah keluaran engine menjadi record terstruktur.

P

PAGEREF: Field Microsoft Word untuk menampilkan nomor halaman sebuah bookmark.

Parallelism: Pelaksanaan tugas secara bersamaan pada beberapa core, thread, process, atau perangkat.

PDOS: Projected Density of States, kontribusi bagian tertentu terhadap density of states.

Provenance: Catatan asal-usul data, parameter, versi, dan transformasi yang menghasilkan suatu hasil.

Q

QCSchema: Schema data umum untuk merepresentasikan input dan output perhitungan kimia kuantum.

QM: Quantum Mechanics atau komputasi kimia kuantum.

QM/MM: Pendekatan hibrida quantum mechanics/molecular mechanics.

QSAR: Quantitative Structure-Activity Relationship, pemodelan hubungan struktur dan aktivitas/properti.

R

Raman Activity: Besaran yang berkaitan dengan intensitas hamburan Raman suatu mode vibrasi.

Redocking: Docking ulang ligan referensi ke reseptor untuk mengevaluasi reproduksi pose.

Regression Test: Tes yang memastikan pembaruan kode tidak merusak perilaku yang sebelumnya benar.

RMSD: Root-Mean-Square Deviation, ukuran perbedaan struktur/koordinat.

S

Sanitization: Pemeriksaan dan normalisasi struktur kimia pada cheminformatics.

SASA: Solvent Accessible Surface Area.

SCF: Self-Consistent Field, prosedur iteratif pada metode mean-field/DFT.

SMILES: Representasi teks untuk konektivitas dan stereokimia molekul.

Stereoisomer: Molekul dengan konektivitas sama tetapi susunan ruang berbeda.

T

Tautomer: Isomer yang saling berkonversi melalui perpindahan proton dan ikatan rangkap.

TDDFT: Time-Dependent Density Functional Theory untuk keadaan tereksitasi dan spektrum elektronik.

Thermochemistry: Perhitungan besaran seperti entalpi dan energi bebas dari energi elektronik dan kontribusi termal.

Transition State: Titik stasioner orde pertama yang menghubungkan jalur reaksi tertentu.

U

UFF: Universal Force Field untuk optimasi geometri mekanika molekuler.

Unit Test: Tes otomatis terhadap fungsi atau komponen kecil secara terisolasi.

Unit Conversion: Transformasi besaran antarsatuan dengan aturan yang terdokumentasi.

UV-Vis: Spektroskopi ultraviolet-tampak yang berkaitan dengan transisi elektronik.

V

Validation: Proses memeriksa apakah software/metode memenuhi acuan dan tujuan yang ditentukan.

Versioning: Pengelolaan versi kode, schema, model, dan data.

Virtual Screening: Penyaringan komputasional sejumlah besar kandidat molekul.

Vibrational Frequency: Frekuensi mode normal yang digunakan pada IR/Raman dan karakter titik stasioner.

W

Workflow: Urutan langkah terstruktur yang menghubungkan input, eksekusi, analisis, dan output.

Workspace: Direktori kerja terisolasi tempat file job disimpan.

Warning: Pesan kondisi meragukan yang tidak selalu menghentikan job tetapi harus diperhatikan.

WSL: Windows Subsystem for Linux, lingkungan Linux yang berjalan pada Windows.

X

XLSX: Format spreadsheet yang umum digunakan untuk ekspor tabel hasil.

XML: Extensible Markup Language, format yang juga menjadi dasar struktur internal DOCX.

xTB: Keluarga metode extended tight-binding untuk perhitungan berbiaya rendah.

XDG Directory: Konvensi direktori pada Linux yang dapat digunakan untuk menempatkan data/config pengguna secara benar.

Y

Y-Randomization: Uji dengan mengacak target untuk menilai apakah performa QSAR muncul secara kebetulan.

YAML: Format konfigurasi human-readable yang dapat digunakan untuk spesifikasi workflow.

Yield of Workflow: Ukuran operasional proporsi job yang berhasil dan lolos QC dari seluruh job yang masuk.

Yoked Parameters: Parameter yang harus berubah secara konsisten karena memiliki dependensi logis dalam workflow.

Z

Z-Matrix: Representasi geometri berdasarkan panjang ikatan, sudut, dan dihedral.

Zero-Point Energy: Energi vibrasi dasar yang tetap ada pada 0 K dalam pendekatan kuantum.

Z-Score: Standardisasi statistik yang dapat digunakan pada analisis descriptor.

Zone of Applicability: Istilah konseptual untuk wilayah input di mana metode/model memiliki dukungan validasi yang memadai.

DAFTAR PUSTAKA

Abraham, M. J., Murtola, T., Schulz, R., Páll, S., Smith, J. C., Hess, B., & Lindahl, E. (2015). GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX, 1-2, 19-25.

Baker, J. (1986). An algorithm for the location of transition states. Journal of Computational Chemistry, 7, 385-395.

Bannwarth, C., Ehlert, S., & Grimme, S. (2019). GFN2-xTB—An accurate and broadly parametrized self-consistent tight-binding quantum chemical method with multipole electrostatics and density-dependent dispersion contributions. Journal of Chemical Theory and Computation, 15, 1652-1671. https://doi.org/10.1021/acs.jctc.8b01176

Becke, A. D. (1993). Density-functional thermochemistry. III. The role of exact exchange. Journal of Chemical Physics, 98, 5648-5652.

Boys, S. F. (1960). Construction of some molecular orbitals to be approximately invariant for changes from one molecule to another. Reviews of Modern Physics, 32, 296-299.

Boys, S. F., & Bernardi, F. (1970). The calculation of small molecular interactions by the differences of separate total energies. Some procedures with reduced errors. Molecular Physics, 19, 553-566.

Breiman, L. (2001). Random forests. Machine Learning, 45, 5-32.

Casida, M. E. (1995). Time-dependent density functional response theory for molecules. In D. P. Chong (Ed.), Recent Advances in Density Functional Methods. World Scientific.

Cherkasov, A., Muratov, E. N., Fourches, D., Varnek, A., Baskin, I. I., Cronin, M., Dearden, J., Gramatica, P., Martin, Y. C., Todeschini, R., Consonni, V., Kuz'min, V. E., Cramer, R., Benigni, R., Yang, C., Rathman, J., Terfloth, L., Gasteiger, J., Richard, A., & Tropsha, A. (2014). QSAR modeling: Where have you been? Where are you going to? Journal of Medicinal Chemistry, 57, 4977-5010.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20, 273-297.

Cramer, C. J. (2004). Essentials of Computational Chemistry: Theories and Models (2nd ed.). Wiley.

Eastman, P., Swails, J., Chodera, J. D., McGibbon, R. T., Zhao, Y., Beauchamp, K. A., Wang, L.-P., Simmonett, A. C., Harrigan, M. P., Stern, C. D., Wiewiora, R. P., Brooks, B. R., & Pande, V. S. (2017). OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. PLOS Computational Biology, 13, e1005659. https://doi.org/10.1371/journal.pcbi.1005659

Eberhardt, J., Santos-Martins, D., Tillack, A. F., & Forli, S. (2021). AutoDock Vina 1.2.0: New docking methods, expanded force field, and Python bindings. Journal of Chemical Information and Modeling, 61, 3891-3898.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29, 1189-1232.

Fukui, K. (1981). The path of chemical reactions - the IRC approach. Accounts of Chemical Research, 14, 363-368.

Gramatica, P. (2007). Principles of QSAR models validation: Internal and external. QSAR & Combinatorial Science, 26, 694-701.

Grimme, S., Antony, J., Ehrlich, S., & Krieg, H. (2010). A consistent and accurate ab initio parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu. Journal of Chemical Physics, 132, 154104.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning (2nd ed.). Springer.

Helgaker, T., Jørgensen, P., & Olsen, J. (2000). Molecular Electronic-Structure Theory. Wiley.

Henkelman, G., Uberuaga, B. P., & Jónsson, H. (2000). A climbing image nudged elastic band method for finding saddle points and minimum energy paths. Journal of Chemical Physics, 113, 9901-9904.

Hirshfeld, F. L. (1977). Bonded-atom fragments for describing molecular charge densities. Theoretica Chimica Acta, 44, 129-138.

Hoover, W. G. (1985). Canonical dynamics: Equilibrium phase-space distributions. Physical Review A, 31, 1695-1697.

Huang, J., Rauscher, S., Nawrocki, G., Ran, T., Feig, M., de Groot, B. L., Grubmüller, H., & MacKerell, A. D. (2017). CHARMM36m: An improved force field for folded and intrinsically disordered proteins. Nature Methods, 14, 71-73.

Jensen, F. (2017). Introduction to Computational Chemistry (3rd ed.). Wiley.

Jorgensen, W. L., Chandrasekhar, J., Madura, J. D., Impey, R. W., & Klein, M. L. (1983). Comparison of simple potential functions for simulating liquid water. Journal of Chemical Physics, 79, 926-935.

Koch, W., & Holthausen, M. C. (2001). A Chemist's Guide to Density Functional Theory (2nd ed.). Wiley-VCH.

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

Leach, A. R. (2001). Molecular Modelling: Principles and Applications (2nd ed.). Pearson.

Lee, C., Yang, W., & Parr, R. G. (1988). Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density. Physical Review B, 37, 785-789.

Löwdin, P.-O. (1950). On the non-orthogonality problem connected with the use of atomic wave functions in the theory of molecules and crystals. Journal of Chemical Physics, 18, 365-375.

McGibbon, R. T., Beauchamp, K. A., Harrigan, M. P., Klein, C., Swails, J. M., Hernández, C. X., Schwantes, C. R., Wang, L.-P., Lane, T. J., & Pande, V. S. (2015). MDTraj: A modern open library for the analysis of molecular dynamics trajectories. Biophysical Journal, 109, 1528-1532.

Michaud-Agrawal, N., Denning, E. J., Woolf, T. B., & Beckstein, O. (2011). MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. Journal of Computational Chemistry, 32, 2319-2327.

Mulliken, R. S. (1955). Electronic population analysis on LCAO-MO molecular wave functions. Journal of Chemical Physics, 23, 1833-1840.

Nosé, S. (1984). A unified formulation of the constant temperature molecular dynamics methods. Journal of Chemical Physics, 81, 511-519.

OECD. (2007). Guidance Document on the Validation of (Quantitative) Structure-Activity Relationship [(Q)SAR] Models. OECD Series on Testing and Assessment No. 69.

Parr, R. G., & Yang, W. (1989). Density-Functional Theory of Atoms and Molecules. Oxford University Press.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Perdew, J. P., Burke, K., & Ernzerhof, M. (1996). Generalized gradient approximation made simple. Physical Review Letters, 77, 3865-3868.

Pracht, P., Bohle, F., & Grimme, S. (2020). Automated exploration of the low-energy chemical space with fast quantum chemical methods. Physical Chemistry Chemical Physics, 22, 7169-7192.

Roe, D. R., & Cheatham, T. E. (2013). PTRAJ and CPPTRAJ: Software for processing and analysis of molecular dynamics trajectory data. Journal of Chemical Theory and Computation, 9, 3084-3095.

Runge, E., & Gross, E. K. U. (1984). Density-functional theory for time-dependent systems. Physical Review Letters, 52, 997-1000.

Sandve, G. K., Nekrutenko, A., Taylor, J., & Hovig, E. (2013). Ten simple rules for reproducible computational research. PLOS Computational Biology, 9, e1003285.

Schlegel, H. B. (1982). Optimization of equilibrium geometries and transition structures. Journal of Computational Chemistry, 3, 214-218.

Stephens, P. J., Devlin, F. J., Chabalowski, C. F., & Frisch, M. J. (1994). Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields. Journal of Physical Chemistry, 98, 11623-11627.

Stratmann, R. E., Scuseria, G. E., & Frisch, M. J. (1998). An efficient implementation of time-dependent density-functional theory for the calculation of excitation energies of large molecules. Journal of Chemical Physics, 109, 8218-8224.

Sun, Q., Berkelbach, T. C., Blunt, N. S., Booth, G. H., Guo, S., Li, Z., Liu, J., McClain, J., Sayfutyarova, E. R., Sharma, S., Wouters, S., & Chan, G. K.-L. (2018). PySCF: The Python-based simulations of chemistry framework. WIREs Computational Molecular Science, 8, e1340. https://doi.org/10.1002/wcms.1340

Sun, Q., Zhang, X., Banerjee, S., Bao, P., Barbry, M., Blunt, N. S., Bogdanov, N. A., Booth, G. H., Chen, J., Cui, Z.-H., Eriksen, J. J., Gao, Y., Guo, S., Hermann, J., Hermes, M. R., Koh, K., Koval, P., Lehtola, S., Li, Z., Liu, J., Mardirossian, N., McClain, J. D., Motta, M., Mussard, B., Pham, H. Q., Pulkin, A., Purwanto, W., Robinson, P. J., Ronca, E., Sayfutyarova, E. R., Scheurer, M., Schurkus, H. F., Smith, J. E. T., Sun, C., Sun, S.-N., Upadhyay, S., Wagner, L. K., Wang, X., White, A., Whitfield, J. D., Williamson, M. J., Wouters, S., Yang, J., Yu, J. M., Zhu, T., Berkelbach, T. C., Sharma, S., Sokolov, A. Y., & Chan, G. K.-L. (2020). Recent developments in the PySCF program package. Journal of Chemical Physics, 153, 024109.

Szabo, A., & Ostlund, N. S. (1996). Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory. Dover.

Tkatchenko, A., & Scheffler, M. (2009). Accurate molecular van der Waals interactions from ground-state electron density and free-atom reference data. Physical Review Letters, 102, 073005.

Todeschini, R., & Consonni, V. (2009). Molecular Descriptors for Chemoinformatics. Wiley-VCH.

Trott, O., & Olson, A. J. (2010). AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of Computational Chemistry, 31, 455-461. https://doi.org/10.1002/jcc.21334

Wang, J., Wolf, R. M., Caldwell, J. W., Kollman, P. A., & Case, D. A. (2004). Development and testing of a general AMBER force field. Journal of Computational Chemistry, 25, 1157-1174.

Wang, L.-P., Song, C., & Martínez, T. J. (2016). Developing a robust and efficient algorithm for computational chemistry geometry optimization. Journal of Chemical Physics, 144, 214108.

Weigend, F., & Ahlrichs, R. (2005). Balanced basis sets of split valence, triple zeta valence and quadruple zeta valence quality for H to Rn. Physical Chemistry Chemical Physics, 7, 3297-3305.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., et al. (2016). The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, 3, 160018.

Wilson, G., Aruliah, D. A., Brown, C. T., et al. (2014). Best practices for scientific computing. PLOS Biology, 12, e1001745.

Zhao, Y., & Truhlar, D. G. (2008). The M06 suite of density functionals for main group thermochemistry, thermochemical kinetics, noncovalent interactions, excited states, and transition elements. Theoretical Chemistry Accounts, 120, 215-241.

AutoDock Vina Documentation. (2026). AutoDock Vina user and Python documentation. https://autodock-vina.readthedocs.io/ (diakses 22 Agustus 2026).

MolSSI. (2026). QCSchema, QCElemental, QCEngine, and QCArchive documentation. https://molssi.github.io/ (diakses 22 Agustus 2026).

OpenMM Documentation. (2026). Python API and user guide. https://docs.openmm.org/ (diakses 22 Agustus 2026).

PySCF Documentation. (2026). PySCF user guide and API documentation. https://pyscf.org/ (diakses 22 Agustus 2026).

RDKit Documentation. (2026). RDKit 2026.03.5 documentation. https://www.rdkit.org/docs/ (diakses 22 Agustus 2026).

Scikit-learn Developers. (2026). Scikit-learn user guide. https://scikit-learn.org/ (diakses 22 Agustus 2026).

xTB-Python Documentation. (2026). Python API for the extended tight binding program. https://xtb-python.readthedocs.io/ (diakses 22 Agustus 2026).


+ HCl
computation
computing
- KASMUI -
Dosen Kimia, Komputasi, IT, AI UNNES
https://hisabmu.com/
https://kasmui.cloud/
Ketua PCM Gunungpati 2
Anggota Majelis Tabligh PDM Kota Semarang & PWM Jawa Tengah
Tim Pengembang Software KHGT MTT PP Muhammadiyah
Praktisi Ilmu Falak