BAB I
PENDAHULUAN

1.1 Latar Belakang

Praktikum merupakan komponen fundamental dalam pendidikan tinggi, khususnya pada bidang sistem informasi. Melalui praktikum, mahasiswa dapat mengaplikasikan konsep-konsep teoritis yang dipelajari di kelas dalam implementasi nyata, sehingga pemahaman terhadap materi menjadi lebih komprehensif (Hwang et al., 2020). Dalam konteks pembelajaran pemrograman, praktikum menjadi sarana penting untuk melatih kemampuan *problem solving* (pemecahan masalah), logika berpikir komputasional, dan keterampilan teknis yang diperlukan di dunia industri.

Pembuatan soal praktikum yang berkualitas merupakan tantangan tersendiri bagi tenaga pengajar dan asisten laboratorium. Soal yang baik harus memenuhi beberapa kriteria, diantaranya relevan dengan materi yang diajarkan, memiliki tingkat kesulitan yang sesuai dengan kompetensi mahasiswa, mampu mengukur pencapaian tujuan pembelajaran, serta memberikan konteks yang realistis dan menarik (Li et al., 2025). Proses pembuatan soal secara manual memerlukan waktu yang signifikan, terutama ketika volume mahasiswa dan variasi mata kuliah yang harus dilayani cukup besar. Selain itu, tuntutan untuk menghasilkan soal yang bervariasi agar tidak terjadi kecurangan akademik menambah beban kerja yang harus ditanggung.

Perkembangan pesat dalam bidang kecerdasan buatan, khususnya *large language model*, telah membuka peluang baru dalam otomatisasi berbagai tugas yang bersifat *knowledge intensive* (intensif pengetahuan) (Brown et al., 2020). Model seperti GPT-4, LLaMa, dan Gemini telah menunjukkan kemampuan luar biasa dalam memahami konteks dan menghasilkan teks yang koheren, termasuk dalam domain pendidikan (Minaee et al., 2025). Penerapan LLM dalam pendidikan mencakup berbagai aplikasi seperti *virtual tutoring* (bimbingan virtual), *automated assessment* (penilaian otomatis), pembuatan ringkasan materi, dan pembuatan soal otomatis (Wang et al., 2024).

Namun demikian, penerapan LLM secara langsung untuk pembuatan soal pendidikan memiliki beberapa keterbatasan yang kritis. Pertama, LLM rentan menghasilkan informasi yang tidak akurat atau *hallucination,* di mana model menghasilkan konten yang terdengar meyakinkan namun secara faktual salah (Ji et al., 2023). Dalam konteks pendidikan di mana akurasi dan kepercayaan sangat esensial, masalah ini dapat berdampak signifikan terhadap hasil pembelajaran. Kedua, pengetahuan yang tersimpan dalam *large language model* bersifat statis, terbatas pada data yang digunakan saat pelatihan, sehingga tidak dapat merefleksikan pembaruan kurikulum atau materi terkini (Mallen et al., 2023). Ketiga, *large language model* seringkali kurang mampu menghasilkan soal yang spesifik dan kontekstual terhadap materi modul tertentu yang menjadi acuan pembelajaran.

Untuk mengatasi keterbatasan tersebut, paradigma *retrieval-augmented generation* telah diperkenalkan sebagai solusi yang menggabungkan kekuatan generatif *large language model* dengan mekanisme pencarian informasi eksternal (Lewis et al., 2021). Berbeda dengan *large language model* standar yang hanya mengandalkan pengetahuan yang telah dipelajari saat pelatihan, *retrieval-augmented generation* terlebih dahulu mengambil dokumen-dokumen yang relevan dari basis pengetahuan eksternal sebelum menghasilkan respons. Pendekatan ini meningkatkan akurasi faktual, memungkinkan

pembaruan pengetahuan yang dinamis, dan memberikan transparansi yang lebih baik karena sumber informasi dapat ditelusuri (Gao et al., 2024).

Dalam konteks pembuatan soal pendidikan, *retrieval-augmented generation* menawarkan potensi yang sangat menjanjikan. Dengan mengintegrasikan modul praktikum sebagai basis pengetahuan, sistem *retrieval-augmented generation* dapat menghasilkan soal yang benar-benar relevan dengan materi yang diajarkan (Maity et al., 2025). Penelitian terbaru menunjukkan bahwa integrasi RAG dengan LLM dapat menghasilkan soal/kuis yang lebih kontekstual dan relevan terhadap materi sumber dibandingkan pendekatan generasi tanpa konteks eksternal, termasuk pada skenario pembuatan kuis adaptif berbasis dokumen (Gopi et al., 2024). Studi sistematis oleh Li et al. (2025) mengidentifikasi bahwa *retrieval-augmented generation* telah diterapkan dalam berbagai skenario pendidikan termasuk sistem *Q&A* (tanya jawab), *chatbot* edukasi (bot percakapan), sistem *tutoring*, dan pembuatan konten pembelajaran.

Meskipun penelitian terdahulu telah menunjukkan potensi RAG dalam berbagai aplikasi pendidikan (Li et al., 2025), belum ada penelitian yang secara spesifik mengimplementasikan sistem RAG-LLM untuk pembuatan soal praktikum pemrograman yang terintegrasi dengan sistem manajemen laboratorium. Penelitian yang telah ada umumnya berfokus pada pembuatan soal pilihan ganda atau kuis sederhana, bukan soal praktis berbasis koding dengan format terstruktur yang mencakup deskripsi soal, *requirements* (kebutuhan), *expected output* (ekspektasi keluaran), dan kunci jawaban. Selain itu, implementasi RAG untuk konteks pendidikan di Indonesia, khususnya untuk materi berbahasa Indonesia, masih sangat terbatas.

Aplikasi SI-Lab merupakan platform manajemen laboratorium yang terintegrasi. Aplikasi ini mengelola berbagai aspek kegiatan praktikum mulai dari penjadwalan, penugasan, pengelolaan modul, hingga penilaian. Namun demikian, proses pembuatan soal praktikum masih dilakukan secara manual oleh asisten laboratorium, yang membutuhkan waktu dan usaha yang signifikan. Integrasi teknologi *retrieval-augmented generation* dan *large language model* ke dalam SI-Lab diharapkan dapat meningkatkan efisiensi dan konsistensi dalam pembuatan soal praktikum.

Berdasarkan latar belakang yang telah diuraikan, penelitian ini bertujuan untuk mengimplementasikan sistem *retrieval-augmented generation* yang terintegrasi dengan *large language model* untuk otomatisasi pembuatan soal praktikum pada *website* (situs web) *SI-Lab* (aplikasi web sistem informasi laboratorium). Sistem yang dikembangkan diharapkan mampu menghasilkan soal yang relevan dengan materi modul, terstruktur dengan baik, dan dapat disesuaikan tingkat kesulitannya sesuai kebutuhan pembelajaran.

Untuk menjawab pertanyaan penelitian tersebut, penelitian ini mengembangkan sistem RAG-LLM dengan arsitektur yang mencakup *text extraction* dan *semantic chunking* untuk memproses dokumen modul, *vector storage* menggunakan FAISS untuk penyimpanan *embedding*, *two-stage retrieval* dengan *cross-encoder reranking* untuk meningkatkan akurasi konteks, integrasi LLM dengan teknik *prompt engineering*, dan evaluasi kualitas melalui metrik otomatis dan evaluasi expert. Sistem yang dikembangkan diharapkan mampu menghasilkan soal yang relevan dengan materi modul, terstruktur dengan baik, dan dapat disesuaikan tingkat kesulitannya sesuai kebutuhan pembelajaran.

1.2 Rumusan Masalah

1. Bagaimana merancang arsitektur sistem *retrieval-augmented generation* yang efektif untuk mengekstrak konteks relevan dari modul praktikum pemrograman?
2. Bagaimana mengintegrasikan komponen *retrieval-augmented generation* dengan *large language model* untuk menghasilkan soal praktikum yang terstruktur, dan sesuai dengan materi modul?
3. Bagaimana performa sistem *retrieval-augmented generation* dan *large language model* dalam menghasilkan soal praktikum ditinjau dari aspek relevansi, struktur output, dan kesesuaian dengan tingkat kesulitan yang diinginkan?
4. Bagaimana implementasi sistem *retrieval-augmented generation* dan *large language model* pada aplikasi *website* SI-Lab Program Studi Sistem Informasi Universitas Hasanuddin?

1.3 Tujuan Penelitian

1. Merancang dan mengimpelementasikan pipeline *retrieval-augmented generation* yang meliputi *document processing* seperti ekstraksi dan *semantic chunking embedding*, *vector storage* menggunakan FAISS, serta *retrieval* dengan mekanisme *reranking.*
2. Mengintegrasikan pipeline *retrieval-augmented generation* dengan *large language model* melalui teknik *prompt engineering* untuk menghasilkan soal praktikum dengan format terstruktur
3. Mengevaluasi performa sistem melalui metrik kualitas *retrieval* dan evaluasi kualitas soal melalui evaluasi *expert* dengan pendekatan *formative evaluation*.
4. Mengimplementasikan sistem pada aplikasi *website* SI-Lab dengan antarmuka *REST API* (antarmuka pemrograman aplikasi berbasis REST) yang dapat diintegrasikan dengan modul-modul *existing* (yang sudah ada).

1.4 Manfaat Penelitian

1. Menyediakan referensi arsitektur sistem retrieval-augmented generation yang dapat mengekstrak konteks relevan dari dokumen modul praktikum pemrograman dan dapat diadaptasi untuk konteks pendidikan lainnya.
2. Menghasilkan sistem yang mampu memproduksi soal praktikum terstruktur secara otomatis, sehingga meningkatkan efisiensi waktu asisten laboratorium dan menjamin konsistensi format soal.
3. Menyediakan metodologi evaluasi sistem pembuatan soal otomatis yang mencakup metrik retrieval dan evaluasi expert, yang dapat dijadikan referensi penelitian selanjutnya
4. Menghasilkan sistem yang terintegrasi dengan aplikasi SI-Lab melalui REST API, menyediakan variasi soal yang beragam dan relevan dengan materi modul untuk mendukung kegiatan praktikum di Program Studi Sistem Informasi Universitas Hasanuddin

1.5 Batasan Masalah

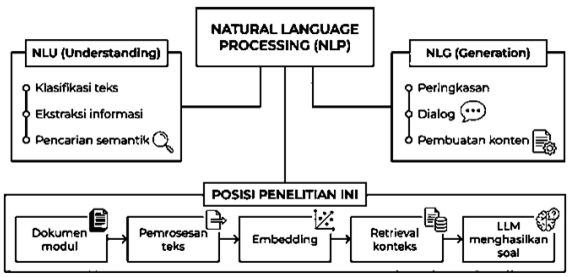
1. Pendekatan *hybrid embedding* yang menggabungkan model *text-focused* dan *code-focused* belum dieksplorasi, sehingga optimasi *retrieval* untuk konten campuran teks dan kode belum diterapkan.
2. Evaluasi berbasis pembelajaran belum dilakukan, sehingga pengukuran peningkatan pemahaman atau *learning outcome* mahasiswa setelah menggunakan soal yang dihasilkan belum menjadi cakupan pembahasan.
3. Sistem dikembangkan untuk lingkungan aplikasi *website* SI-Lab Universitas Hasanuddin dengan integrasi melalui REST API.
4. *Large language model* yang digunakan adalah model yang diakses melalui OpenRouter API, yaitu model openai/gpt-oss-120b:free.
5. Dukungan format dokumen input masih dibatasi pada format yang didukung sistem, sehingga format lain seperti PPTX, HTML, dan Markdown belum didukung, serta peningkatan kualitas ekstraksi teks untuk berbagai variasi format belum menjadi fokus pengembangan.
6. Dukungan multimodal belum tersedia, sehingga sistem belum mampu mendeteksi dan memproses gambar, diagram, maupun ilustrasi dalam dokumen modul, serta belum menghasilkan referensi visual dalam soal untuk mata kuliah yang menekankan aspek tampilan.
7. Sistem hanya menghasilkan soal tipe esai/praktikum, tidak mendukung pembuatan soal pilihan ganda atau tipe soal lainnya.

1.6 Landasan Teori

1.6.1 Kecerdasan buatan dalam pendidikan

Kecerdasan buatan (*artificial intelligence;* AI) dalam pendidikan merujuk pada penerapan teknologi AI untuk meningkatkan proses belajar mengajar (Hwang et al., 2020). Aplikasi AI dalam pendidikan mencakup sistem *tutoring* cerdas, penilaian otomatis, personalisasi pembelajaran, dan pembuatan konten pendidikan. Perkembangan AI generatif, khususnya *large language model*, telah membuka paradigma baru dalam otomatisasi tugas-tugas pendidikan yang sebelumnya memerlukan intervensi manusia secara intensif.

1.6.2 Pemrosesan bahasa alami (Natural Language Processing/NLP)



Gambar 1. Ruang lingkup pemrosesan bahasa alami dan posisi penelitian ini

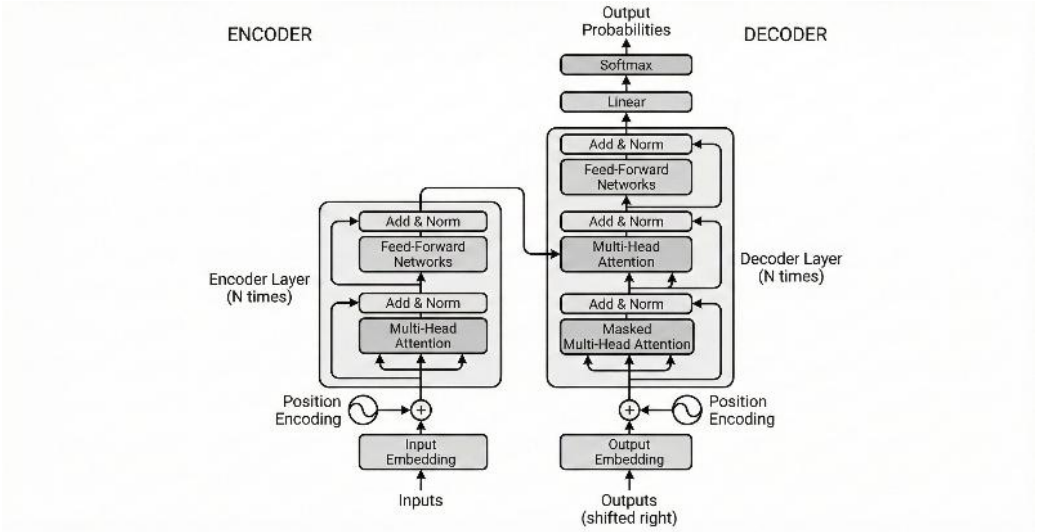
Pemrosesan Bahasa Alami (*Natural Language Processing/NLP*) adalah bidang dalam kecerdasan buatan yang mempelajari cara komputer memproses, memahami, dan menghasilkan bahasa manusia dalam bentuk teks maupun ujaran (Jurafsky & Martin, 2023). Secara umum, NLP mencakup dua spektrum utama, yaitu pemahaman bahasa (*natural language understanding*) dan pembangkitan bahasa (*natural language generation*). Pada ranah pemahaman bahasa, sistem berupaya mengekstraksi makna dan informasi dari teks, misalnya melalui klasifikasi, ekstraksi entitas, atau pencarian semantik. Pada ranah pembangkitan bahasa, sistem menghasilkan teks yang koheren dan sesuai konteks, misalnya untuk peringkasan, dialog, atau pembuatan konten.

Perkembangan NLP modern didorong oleh model berbasis *deep learning* yang mampu merepresentasikan teks ke dalam bentuk vektor berdimensi tinggi sehingga kesamaan makna dapat dihitung secara komputasional. Representasi berbasis vektor ini memungkinkan pencarian semantik, pengelompokan dokumen, dan pemetaan konteks yang relevan terhadap suatu *query* (Manning et al., 2009). Dalam sistem generatif seperti LLM, NLP tidak hanya berfokus pada pemahaman masukan, tetapi juga pada pembentukan keluaran yang terstruktur dan memenuhi tujuan tertentu melalui perancangan *prompt* serta pengaturan format keluaran.

Penelitian ini berada pada irisan antara pemahaman dan pembangkitan bahasa. Pada sisi pemahaman, sistem melakukan pemrosesan dokumen modul untuk menyiapkan basis pengetahuan, termasuk ekstraksi teks, pemotongan menjadi *chunk*, dan pembentukan representasi semantik melalui *embedding* agar sistem mampu melakukan *retrieval* konteks yang relevan. Pada sisi pembangkitan, sistem memanfaatkan LLM untuk menghasilkan soal praktikum berdasarkan konteks yang diambil, dengan struktur keluaran yang ditentukan melalui *prompt engineering*. Dengan demikian, NLP menjadi fondasi konseptual yang menghubungkan proses representasi teks, mekanisme temu kembali konteks, dan pembangkitan soal yang koheren dan sesuai materi modul.

1.6.3 Arsitektur transformer

Transformer adalah arsitektur jaringan saraf yang diperkenalkan oleh Vaswani et al. (2017) dalam makalah *Attention Is All You Need*. Arsitektur ini menjadi fondasi bagi model-model bahasa modern termasuk BERT, GPT, dan LLaMa.



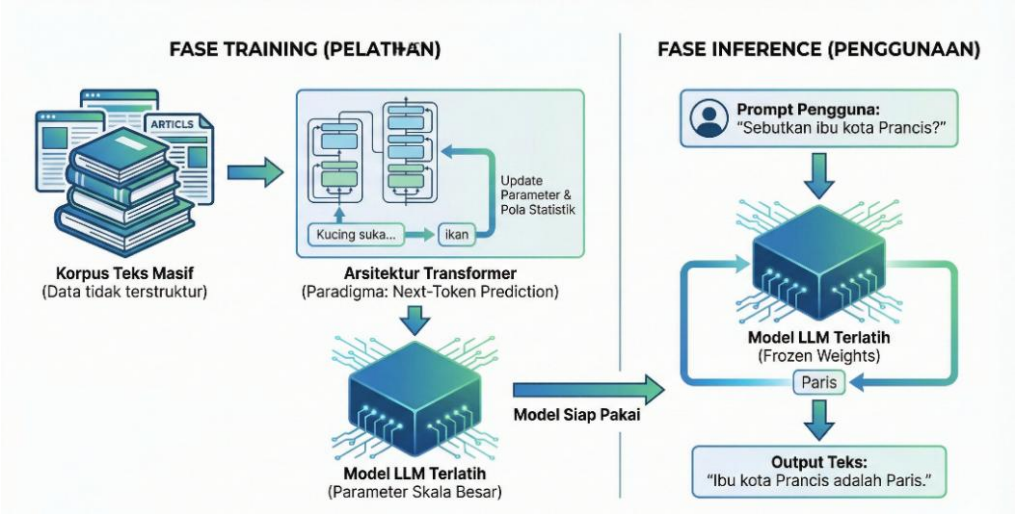
Gambar 2. Arsitektur transformer dengan komponen *encoder* (enkoder) dan *decoder* (dekoder)

Gambar 2 memperlihatkan komponen utama Transformer meliputi:

1. *Self attention mechanism* (mekanisme pelatihan diri): Memungkinkan model untuk mempertimbangkan hubungan antar semua *token* (unit kata) dalam *sequence* (urutan) secara paralel, menangkap dependensi jarak jauh yang sulit ditangani oleh arsitektur *recurrent* (rekuren).
2. *Multi head attention* (perhatian multi kepala): menjalankan beberapa operasi perhatian secara paralel dengan proyeksi yang berbeda, sehingga model dapat berfokus pada berbagai aspek representasi.
3. *Position encoding* (pengodean encoding): Memberikan informasi posisi *token* dalam *sequence* karena Transformer tidak memiliki struktur *recurrent*.
4. *Feed forward networks* (jaringan umpan maju): Layer *fully connected* (tersambung penuh) yang diterapkan pada setiap posisi secara independen untuk mentransformasi representasi fitur.

1.6.4 Large Language Model (LLM)

Large Language Model (LLM) didefinisikan sebagai model bahasa *autoregressive* berbasis *neural network* dengan skala parameter sangat besar, yang dilatih pada korpus teks internet berskala masif untuk mempelajari pola statistik dan memprediksi token berikutnya dalam urutan (Brown et al., 2020).



Gambar 3. Proses *training* (pelatihan) dan *inference* (inferensi) pada LLM

Karakteristik utama LLM:

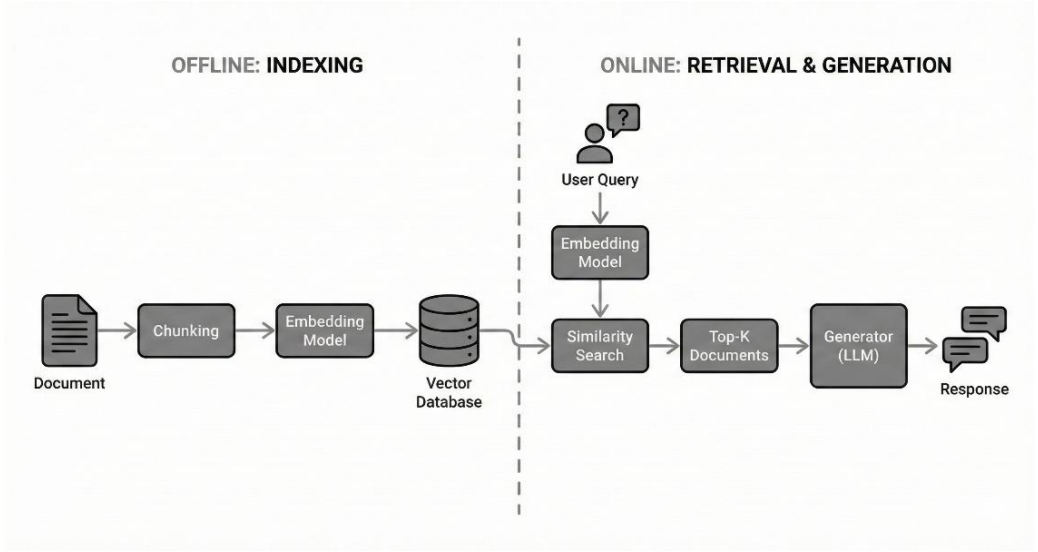
1. Skala Parameter: model dengan miliaran hingga triliunan parameter yang memungkinkan pembelajaran representasi yang kompleks.
2. Pelatihan awal *(pre-training)*: pelatihan pada data teks berskala besar untuk mempelajari pengetahuan umum.
3. Pembelajaran dalam konteks (*in-context learning kemampuan melakukan tugas baru hanya dari contoh yang diberikan dalam prompt (instruksi/masukan), tanpa fine-tuning (penalaan ulang).*
4. Kemampuan emergen (emergent abilities): kemampuan yang muncul pada skala tertentu, seperti *reasoning* (penalaran) dan *chain of thought* (alur penalaran).

Keterbatasan LLM yang relevan dengan penelitian ini:

1. *Hallucination*: Menghasilkan informasi yang terdengar meyakinkan namun tidak akurat (Ji et al., 2023).
2. Batas pengetahuan (*knowledge cutoff*): pengetahuan terbatas pada data pelatihan sehingga tidak merefleksikan informasi terkini.
3. Kurangnya keterpautan sumber (*lack of grounding*): kesulitan menghasilkan *output* (keluaran) yang spesifik berdasarkan sumber tertentu.

1.6.5 Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation adalah paradigma yang menggabungkan kemampuan generatif LLM dengan mekanisme pencarian informasi dari sumber eksternal (Lewis et al., 2021). RAG mengatasi keterbatasan LLM dengan menyediakan konteks faktual dari dokumen sumber.



Gambar 4. Diagram alur workflow retrieval-augmented generation (RAG).

Gambar 4 memperlihatkan workflow RAG terdiri dari tiga tahap:

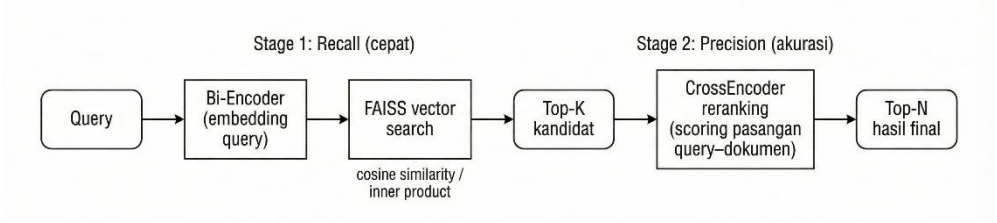
1. Pengindeksan (*indexnig*) luring (*offline*)
   - Dokumen sumber diproses dan dipecah menjadi *chunk* (potongan teks).
   - Setiap *chunk* dikonversi menjadi vektor *embedding* (penyematan).
   - Vektor disimpan dalam *vector database* (basis data vektor) untuk mendukung pencarian yang efisien.
2. Pengambilan (*retrieval*) daring (*online*)
   - *Query* (kueri) pengguna dikonversi menjadi vektor *embedding.*
   - Sistem mencari *chunk* dengan tingkat *similarity* (kemiripan) tertinggi.
   - Sejumlah *top-k* (k-teratas) *chunk* yang paling relevan diambil sebagai konteks.
3. Generasi (*generation*)—daring (*online*)
   - *Chunk* relevan dimasukkan sebagai konteks ke dalam masukan model.
   - LLM menghasilkan *response* (respons) berdasarkan konteks yang diberikan.
   - *Output* di *ground* pada informasi dari dokumen sumber.

   Keunggulan RAG dibandingkan LLM standar (Gao et al., 2024):
   - Akurasi faktual lebih tinggi karena *grounding* pada sumber.
   - Pengetahuan dapat diperbarui tanpa *re-training* (pelatihan ulang) model.
   - Transparansi melalui kemampuan *tracing* (pelacakan) sumber informasi.
   - Efisiensi biaya dibanding *fine tuning* (penalaan ulang).

Perlu dicatat bahwa arsitektur RAG yang diperkenalkan oleh (Lewis et al., 2021) merupakan RAG standar (sering disebut *naive RAG* atau *basic RAG*) yang pada dasarnya terdiri dari komponen *retrieval* berbasis *similarity* dan komponen *generation*. Pada perkembangannya, berbagai teknik optimasi dikembangkan untuk meningkatkan performa RAG, salah satunya *reranking* (pemeringkatan ulang). *Reranking* bukan bagian dari arsitektur RAG standar, melainkan peningkatan (*enhancement*) pada tahap *retrieval* untuk meningkatkan presisi hasil. Penelitian ini mengimplementasikan RAG lanjutan

(*advanced RAG*) dengan menambahkan komponen *reranking* menggunakan *cross-encoder* (enkoder-silang), sehingga berbeda dari implementasi RAG standar.

1.6.6 Temu kembali informasi (Information Retrieval/*IR*) dan Reranking



Gambar 5. Two-stage retrieval untuk pengambilan kandidat dan reranking

Temu kembali informasi (*information retrieval/IR*) adalah bidang yang berfokus pada pencarian dan pemeringkatan dokumen yang paling relevan terhadap kebutuhan informasi pengguna, umumnya dinyatakan sebagai *query* (Manning et al., 2009). Tujuan utama IR bukan sekadar menemukan dokumen yang "mengandung kata yang sama", tetapi memeringkatkan dokumen berdasarkan relevansi terhadap maksud pencarian. Dalam konteks sistem berbasis dokumen, IR umumnya menghasilkan daftar *top-k* kandidat yang dianggap paling relevan, lalu sistem downstream memanfaatkan kandidat tersebut untuk tugas lanjutan.

Pada pendekatan IR klasik, representasi dokumen sering bersifat *sparse* melalui indeks terbalik dan pemodelan berbasis frekuensi istilah, dengan metode populer seperti BM25 untuk menghitung skor relevansi berbasis kecocokan leksikal. Pendekatan ini efektif untuk pencarian berbasis kata kunci, tetapi dapat kurang optimal ketika pengguna menggunakan variasi istilah, sinonim, atau ketika relevansi lebih ditentukan oleh makna semantik. Sebaliknya, pendekatan IR modern banyak memanfaatkan representasi *dense* melalui *embedding* sehingga dokumen dan *query* dapat dibandingkan berdasarkan kedekatan semantik dalam ruang vektor. Pendekatan *dense retrieval* seperti DPR menggunakan *bi-encoder* untuk menghasilkan vektor *query* dan dokumen, lalu melakukan pencarian kandidat dengan skor kemiripan (misalnya *cosine similarity* atau *inner product*) (Karpukhin et al., 2020).

Karena jumlah dokumen dapat sangat besar, pencarian vektor umumnya menggunakan pendekatan *exact nearest neighbor* agar tetap efisien. Implementasi praktisnya dapat memanfaatkan FAISS dengan struktur indeks tertentu yang dirancang untuk mempercepat pencarian kandidat pada skala besar. Namun, *dense retrieval* berbasis *bi-encoder* tetap memiliki keterbatasan: karena *query* dan dokumen di-*encode* secara terpisah, informasi interaksi detail antar-token tidak sepenuhnya dimodelkan pada saat penilaian kandidat. Akibatnya, kandidat yang diambil pada tahap awal bisa kurang optimal urutannya, terutama pada kasus *query* yang ambigu atau konteks dokumen yang mirip-mirip.

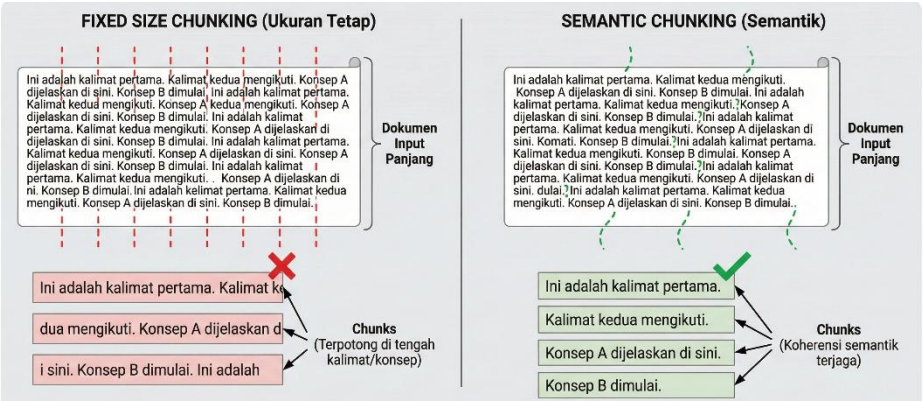
Untuk meningkatkan kualitas pemeringkatan, banyak sistem IR menerapkan *two-stage retrieval*: tahap pertama berfokus pada jangkauan kandidat (mengambil *top-k* secara efisien), sedangkan tahap kedua berfokus pada ketepatan pemeringkatan akhir melalui *reranking* (Nogueira & Cho, 2020). Pada tahap *reranking*, *cross-encoder* menilai

pasangan *query*–dokumen secara langsung sehingga mampu menangkap interaksi yang lebih kaya dan biasanya menghasilkan peringkat yang lebih akurat dibanding *bi-encoder*. Strategi ini relevan untuk sistem RAG karena kualitas konteks yang diambil sangat menentukan kualitas keluaran LLM. Dengan *reranking*, konteks yang diberikan ke LLM cenderung lebih tepat sasaran, sehingga membantu meningkatkan koherensi dan ketepatan soal yang dihasilkan, serta mengurangi risiko konteks yang tidak relevan masuk ke proses generasi (Lewis et al., 2021).

Dalam penelitian ini, IR berperan sebagai komponen inti pada tahap pengambilan konteks modul. Sistem menggabungkan pencarian kandidat berbasis FAISS pada tahap awal dan *cross-encoder reranking* pada tahap lanjutan untuk meningkatkan kualitas konteks yang masuk ke LLM. Dengan demikian, landasan IR dan *reranking* mendukung justifikasi arsitektur *two-stage retrieval* yang digunakan serta menjelaskan mengapa peningkatan kualitas konteks berimplikasi langsung pada kualitas soal praktikum yang dihasilkan.

1.6.7 Text chunking

Text chunking adalah proses memecah dokumen panjang menjadi segmen-segmen yang lebih kecil untuk diproses dalam pipeline RAG. *Chunking* yang efektif sangat penting karena mempengaruhi kualitas *retrieval* dan generasi.



Gambar 6. Perbandingan strategi *chunking*: *fixed size* vs *semantic chunking*

Tabel 1. Jenis-jenis strategi chunking

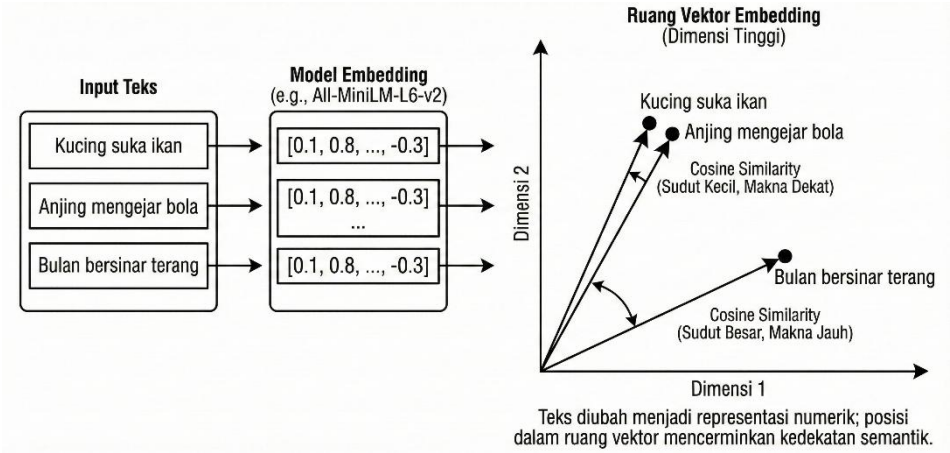
Strategi	Deskripsi	Kelebihan	Kekurangan
Fixed size	Memotong berdasarkan jumlah karakter/token tetap	Sederhana, prediktabel	Dapat memotong di tengah kalimat/konsep
Semantic	Mempertimbangkan batas paragraf dan kalimat	Menjaga koherensi semantik	Ukurang chunk bervariasi
Recursive	Memecah hierarkis dengan separator berbeda	Fleksibel	Kompleksitas implementasi

Parameter *Chunking*:

- *Chunk Size* (ukuran potongan teks): Ukuran maksimum setiap *chunk* (karakter/*token*).
- *Overlap* (tumpang tindih): Jumlah karakter tumpang tindih antar *chunk* untuk menjaga kontinuitas konteks.

1.6.8 Text embedding

Text embedding adalah representasi numerik teks dalam ruang, berdimensi tinggi yang menangkap makna semantik (Reimers & Gurevych, 2019). *Embedding* memungkinkan perhitungan *similarity* antar teks secara matematis.



Gambar 7. Representasi teks dalam ruang vektor *embedding*

Tabel 2. Model embedding yang umum digunakan

Model	Dimensi	Karakteristik
All-MiniLM-L6-v2	384	Ringan, cepat, multilingual
text-embedding-ada-002	1536	Akurasi tinggi, proprietary
BGE-large	1024	Open source, performa tinggi
Sentence-BERT	768	Optimized untuk similarity

Metrik *similarity*:

- *Cosine similarity* (kemiripan kosinus): mengukur sudut antara dua vektor; nilainya berkisar dari -1 hingga 1
- *DOT product* (inner product; hasil kali titik): menghitung hasil kali titik antara dua vektor
- *Euclidean distance* (jarak Euklides): mengukur jarak geometris antara dua vektor.

1.6.9 Vector database dan FAISS

Vector database adalah sistem penyimpanan yang dioptimalkan untuk menyimpan dan mencari vektor *embedding* secara efisien. FAISS (*Facebook AI Similarity Search*) adalah *library* (pustaka) *open source* (sumber terbuka) dari Meta AI untuk pencarian *exact nearest neighbor* (ENN; tetangga terdekat aproksimasi) pada skala besar (Johnson et al., 2021).

Gambar 8. Pencarian *nearest nighbor* dalam *vector database*

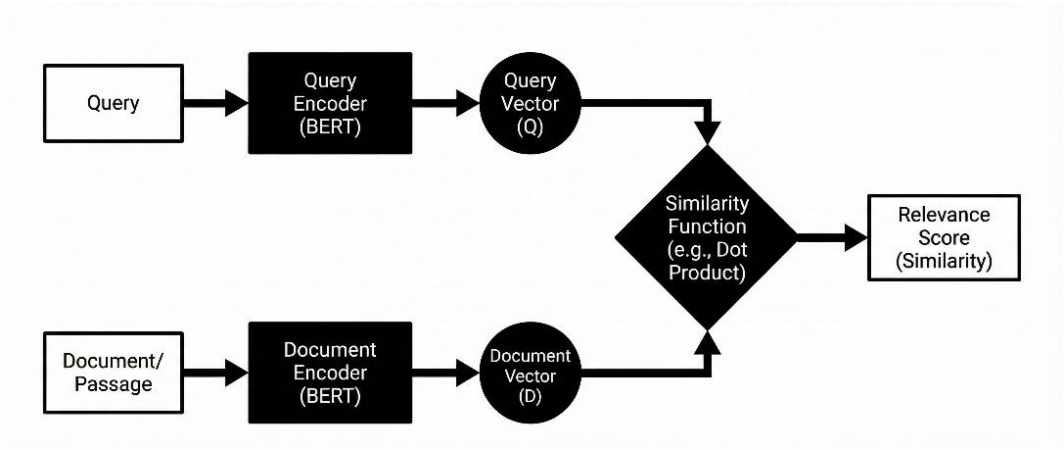
Tabel 3. Jenis-jenis FAISS Index

Index Type	Deskripsi	Use Case
Flat (FlatIP/FlatL2)	*Brute force search* (pencarian menyeluruh), akurasi 100%	Dataset kecil (<100K)
IVF	*Inverted file* (berkas terbalik) dengan *clustering* (pengelompokkan)	Dataset medium
HNSW	*Graph based* (berbasis graf) dengan navigasi hierarkis	Kecepatan tinggi
PQ	*Product quantization* (kuantitas produk) untuk kompresi	*Memory constrained* (berkendala memori)

1.6.10 Dense Passage Retrieval (DPR)

Dense passage retrieval (DPR; temu kembali lintasan padat) adalah teknik *retrieval* yang menggunakan *dual encoder* (enkoder ganda) untuk menghasilkan representasi *dense* (padat) dari *query* dan dokumen (Karpukhin et al., 2020). DPR lebih efektif menangkap

similarity semantic (kemiripan semantik) dibandingkan metode *sparse* (jarang) seperti BM25.



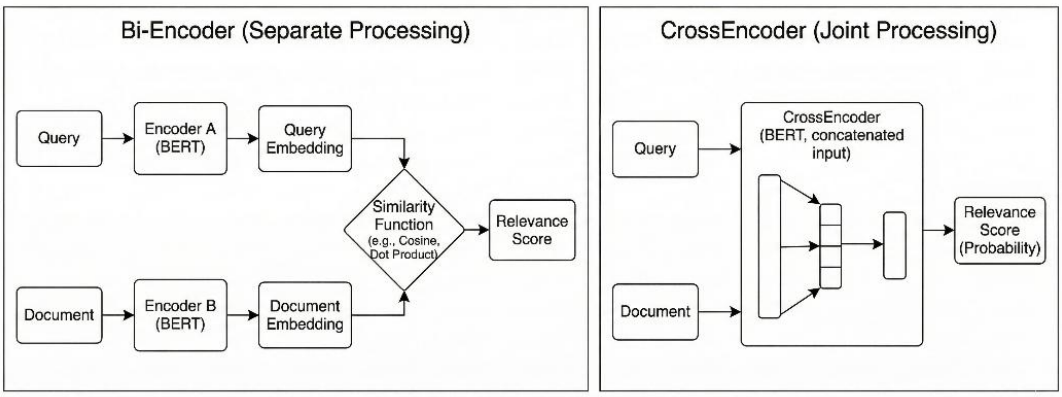
Gambar 9. Arsitektur *dense passage retrieval* dengan *dual encoder*

Komponen DPR:

- Enkoder kueri (*query encoder*): mengonversi query menjadi vektor
- Enkoder dokumen (*document encoder)*: mengonversi dokumen atau *passage* (potongan teks) menjadi vektor
- Fungsi kemiripan (*similarity function*): menghitung relevansi antara vektor kueri dan vektor dokumen.

1.6.11 Reranking dengan cross-encoder

Reranking adalah proses memeringkat ulang hasil *retrieval* awal untuk meningkatkan presisi. *Cross-encoder* mengevaluasi relevansi pasangan query dokumen secara bersamaan, berbeda dengan *bi-encoder* (enkoder ganda) yang menghitung *embedding* secara terpisah (Santhanam et al., 2022).



Gambar 10. Perbandingan *bi-Encoder* vs *cross-encoder* untuk penilaian relevansi (*scoring*)

Two stage retrieval pipeline (alur temu kembali dua tahap):

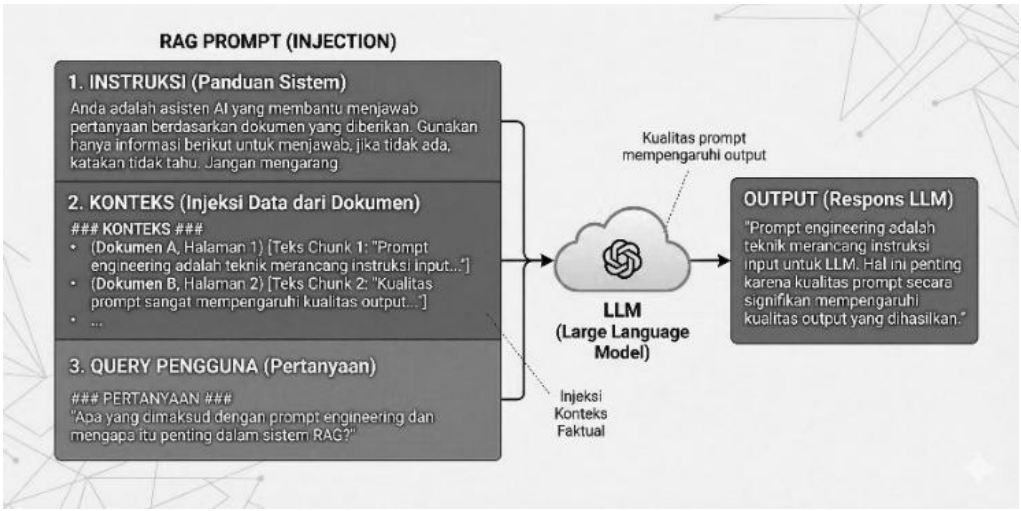
1. Tahap 1 (recall; jangkauan): *bi-Encoder* mengambil *top-K* kandidat
2. Tahap 2 (precision; ketepatan): *Cross-encoder* memeringkatkan ulang kandidat, menghasilkan *top-N* hasil akhir.

Keunggulan pendekatan dua tahap (*two-stage approach*) meliputi:

- Efisiensi: *bi-encoder* cepat untuk *filtering* (penyaringan) awal
- Akurasi: *Cross-encoder lebih akurat untuk pemeringkatan (ranking) akhir*
- Scalability (skalabilitas): mampu menangani korpus yang besar.

1.6.12 Prompt engineering

Prompt engineering adalah teknik merancang instruksi input yang efektif untuk mendapatkan *output* yang diinginkan dari LLM. Kualitas *prompt* (masukan) sangat mempengaruhi kualitas *output* yang dihasilkan.



Gambar 11. Struktur *prompt* untuk sistem RAG dengan *context injection* (penyuntikan konteks)

Komponen *prompt* dalam Sistem RAG:

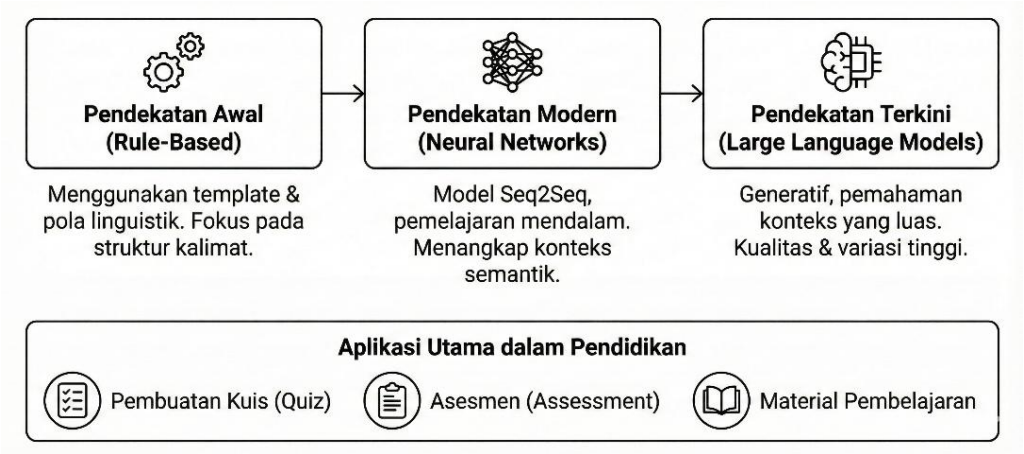
1. *System prompt*: Instruksi peran dan perilaku
2. *Context*: Hasil *retrieval* dari dokumen
3. *User instruction*: *Query*/permintaan spesifik
4. *Output format*: Spesifikasi struktur *output.*

Teknik Prompt Engineering:

- *Zero shot*: Tanpa contoh, hanya instruksi
- *Few shot*: Menyertakan beberapa contoh *input output*
- *Chain of Thought*: Meminta model menjelaskan langkah *reasoning*
- *Role based*: Mendefinisikan persona/peran untuk model.

1.6.13 Automatic Question Generation (AQG)

Automatic Question Generation (AQG; pembuatan pertanyaan otomatis) adalah bidang penelitian yang berfokus pada pembuatan pertanyaan secara otomatis dari materi teks (Maity et al., 2025). AQG memiliki aplikasi penting dalam pendidikan untuk pembuatan *quiz* (kuis), *assessment* (tugas), dan material pembelajaran.



Gambar 12. Evolusi pendekatan *automatic question generation*

Tabel 4. Evolusi pendekatan *automatic question generation*

Era	Pendekatan	Karakteristik
Tradisional	Rule based, template	Menggunakan aturan sintaks dan template
Neural	Seq2Seq (BART, T5)	End to end dengan fine tuning
LLM	GPT, LlaMa + ICL	In context learning tanpa fine tuning

Kriteria evaluasi kualitas pertanyaan. Dalam konteks AQG untuk soal praktikum pemrograman, evaluasi kualitas pertanyaan dapat dilakukan menggunakan aspek-aspek berikut (Kurdi et al., 2020)

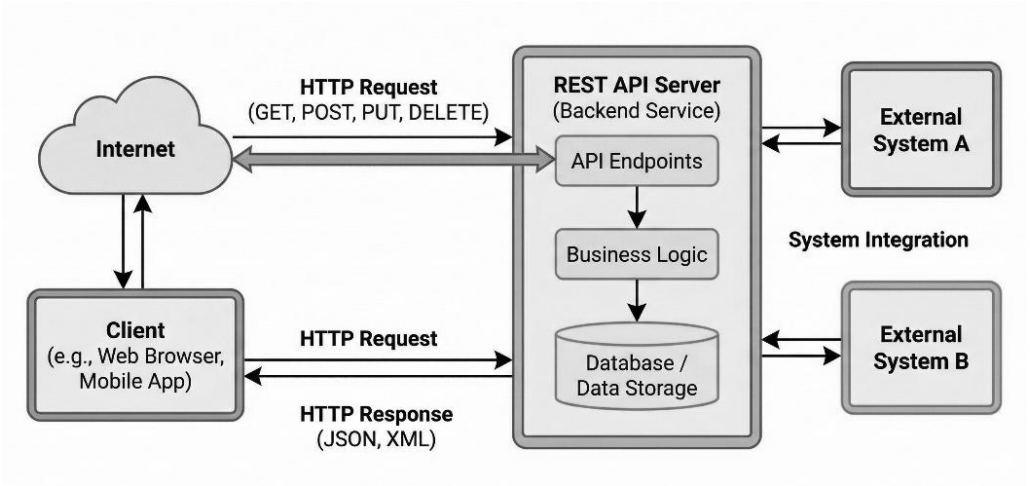
Tabel 5. Aspek-aspek evaluasi kualitas pertanyaan

Aspek	Deskripsi	Referensi
Relevance (relevansi)	Kesesuaian soal dengan materi/konteks sumber	Maity et al. (2025)
Structure (struktur)	Kelengkapan dan kejelasan komponen soal	Kurdi et al. (2020)
Difficulty Match (tingkat kesulitan)	Kesesuaian tingkat kesulitan dengan target	Anderson & Krathwohl (2001)

Aspek	Deskripsi	Referensi
Pedagogical Value (nilai pedagogis)	Nilai pembelajaran dan kemampuan mengukur kompetensi	Merrill (2002)

1.6.14 REST API dan arsitektur web service

REST (*Representational State Transfer*) adalah gaya arsitektur untuk merancang layanan web yang menggunakan protokol HTTP standar (Fielding, 2000). REST API memungkinkan komunikasi antar sistem melalui *request response* dengan format standar.



Gambar 13. Arsitektur REST API untuk integrasi sistem

Prinsip REST:

- *Stateless*: Setiap *request* independen, tidak menyimpan state client
- *Client server*: Pemisahan antara *client* dan *server*
- *Uniform interface*: Antarmuka Standar (HTTP *methods*: GET, POST, PUT, DELETE)
- *Layered system*: Arsitektur berlapis untuk *scalability.*

Format Data:

- JSON (*JavaScript Object Notation*): Format pertukaran data ringan dan mudah dibaca
- HTTP Status Codes: Kode status untuk menandakan hasil operasi (200 OK, 404 Not Founf, 500 Error).

1.6.15 Flask web framework

Flask adalah micro web framework untuk Python yang ringan dan fleksibel (Grinberg, 2018). Flask menyediakan fondasi untuk membangun aplikasi web REST API dengan pendekatan minimalis.

Karakteristik Flask:

- *Lightweight*: Tidak memaksakan struktur tertentu
- *Extensible*: Mudah diperluas dengan *extensions*
- *WSGI compliant*: Kompatibel dengan standar Python web
- *Built in development server*: Server bawaan untuk *development.*

1.6.16 MySQL database

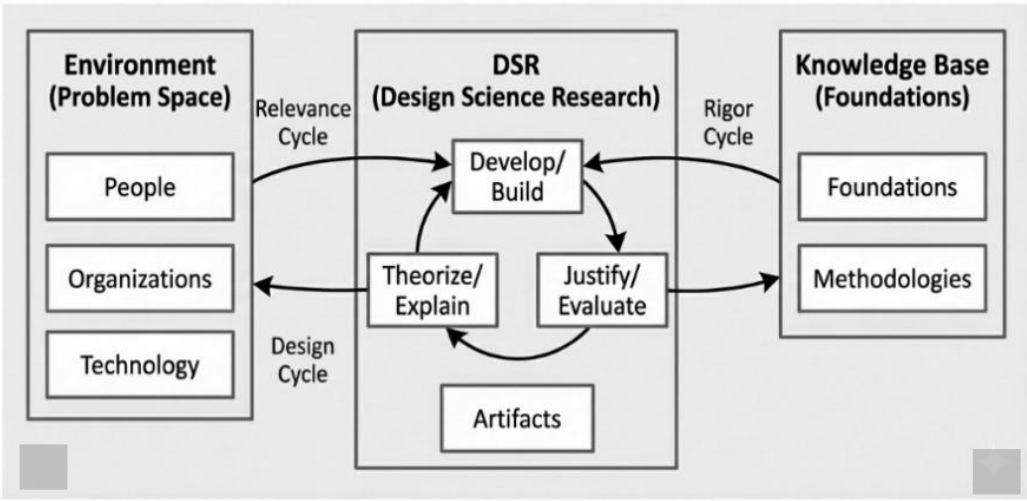
MySQL adalah sistem manajemen basis data relasional (*Relational Database Management System/*RDBMS) yang menggunakan SQL sebagai bahasa query (Oracle, 2023). MySQL digunakan untuk menyimpan data terstruktur seperti informasi mata kuliah, modul, dan hasil generasi soal.

Konsep Database Relasional:

- *Tables*: Struktur penyimpanan data dalam baris dan kolom
- *Primary key*: *Identifier* unik untuk setiap *record*
- *Foreign key*: Referensi antar tabel untuk relasi
- *SQL queries*: Bahasa untuk memanipulasi dan *querying* data.

1.6.17 Design Science Research (DSR)

Design science research (DSR; penelitian sains desain) adalah paradigma penelitian yang berfokus pada pencipataan dan evaluasi artefak TI yang bertujuan untuk menyelesaikan masalah organisasional yang teridentifikasi (Hevner et al., 2004). DSR berbeda dari penelitian *behavioral science* yang berfokus pada pengembangan dan verifikasi teori untuk menjelaskan atau memprediksi fenomena.



Gambar 14. Framework *design science research*

Tabel 6. Prinsip-prinsip DSR menurut Hevner et al. (2004)

Prinsip	Deskripsi
Design as an artifact	Penelitian DSR harus menghasilkan artefak yang viable (construct, model, method, atau instantiation)
Problem relevance	Artefak harus menyelesaikan masalah bisnis/organisasi yang penting dan relevan
Design evaluation	Utilitas, kualitas, dan efektivitas artefak harus dievaluasi secara rigorous
Research contributions	Penelitian harus memberikan kontribusi yang jelas dan dapat diverifikasi
Research rigor	Metode rigorous harus diterapkan dalam konstruksi dan evaluasi artefak
Design as a search process	Pencarian artefak yang efektif memerlukan penggunaan means yang tersedia
Communication of research	Penelitian harus dikomunikasikan kepada audience teknis dan manajerial

Dalam konteks penelitian ini, artefak yang dikembangkan adalah sistem RAG LLM Assessment Generator yang bertujuan menyelesaikan masalah pembuatan soal praktikum SI-Lab serta akan dievaluasi berdasarkan utilitas dan efektivitasnya.

1.6.18 Taksonomi Bloom

Taksonomi Bloom adalah kerangka klasifikasi yang digunakan untuk mengkategorikan tingkat kompleksitas *kognitif* dalam pembelajaran (Bloom et al., 1956). Taksonomi ini kemudian direvisi oleh Anderson & Krathwohl (2001) untuk merefleksikan perkembangan dalam teori pendidikan.

Gambar 15. Piramida taksonomi Bloom revisi*:*

Tabel 7. Tingkatan Kognitif dalam Taksonomi Bloom Revisi

Level	Kategori	Deskripsi	Kata Kerja Contoh
1	*Remembering*	Mengingat informasi faktual	Mendefinisikan, menyebutkan, mengidentifikasi
2	*Understanding*	Memahami makna informasi	Menjelaskan, merangkum, mengklasifikasi
3	*Applying*	Menggunakan informasi dalam situasi baru	Mengimplementasikan, menghitung, menjalankan
4	*Analyzing*	Memecah informasi menjadi bagian-bagian	Membedakan, mengorganisir, menghubungkan
5	*Evaluating*	Membuat penilaian berdasarkan kriteria	Mengkritik, menilai, memvalidasi
6	*Creating*	Menghasilkan sesuatu yang baru	Merancang, mengembangkan, memformulasi

Dalam konteks pembuatan soal praktikum, Taksonomi Bloom digunakan sebagai referensi untuk:

- Menentukan tingkat kesulitan soal
- Mengevaluasi kesesuaian soal dengan level kognitif yang diharapkan
- Memastikan soal mengukur kompetensi yang tepat.

1.6.19 Skala Likert dan interpretasinya

Skala Likert adalah skala psikometrik yang umum digunakan dalam penelitian survei untuk mengukur sikap, opini, atau persepsi (Likert, 1932). Skala ini meminta responden untuk menunjukkan tingkat persetujuan atau penilaian terhadap suatu pernyataan.

Karakteristik Skala Likert:

- *Ordinal*: Data bersifat ordinal (ada urutan tapi jarak antar nilai tidak sama)
- *Bipolar* atau *Unipolar*: Dapat mengukur dari ekstrem negatif ke positif, atau dari tidak ada ke maksimal
- Jumlah Poin: Umumnya 5 atau 7 poin untuk memberikan variasi yang cukup.

Tabel 8. Interpretasi Skala Likert 5 Poin (Vagias, 2006)

Rentang Nilai	Kategori	Interpretasi untuk Evaluasi Kualitas
≥ 1.00	Sangat Buruk	Tidak dapat diterima
≥ 1.80	Buruk	Di bawah standar
≥ 2.60	Cukup	Memenuhi standar minimum
≥ 3.40	Baik	Memenuhi ekspektasi
≥ 4.20	Sangat Baik	Melebihi Ekspektasi

Dalam penelitian ini, skala Likert 1-5 digunakan untuk evaluasi *expert* kualitas soal yang dihasilkan sistem. Interpretasi berdasarkan Vagias (2006) memberikan panduan objektif untuk menentukan kategori kualitas berdasarkan rata-rata skor yang diperoleh.

1.6.20 Nielsen's usability response time

Jason Nielsen (1993) dalam bukunya *Usability Engineering* mengidentifikasi tiga batas waktu respons (*response time threshold*) yang penting dalam desain sistem interaktif:

Tabel 9. Batas Waktu Respons Nielsen (1993)

Threshold	Waktu	Deskripsi
Instantaneous	0.1 detik (100 ms)	Pengguna merasakan respons sebagai instan; tidak ada jeda yang dirasakan
Unintrrupted	1 detik	Batas agar pengguna tetap fokus pada tugas tanpa merasa menunggu
Attention limit	10 detik	Batas maksimum sebelum pengguna kehilangan perhatian dan meninggalkan sistem

Pada Tabel 9 Dalam konteks sistem RAG-LLM, *threshold* 100 ms menjadi target penting untuk waktu respons *retrieval pipeline*. Jika waktu respons melebihi batas ini, pengguna akan merasakan *delay* (keterlambatan) yang dapat mengurangi pengalaman pengguna. Penelitian ini menggunakan *threshold* 100 ms sebagai standar untuk mengevaluasi performa *retrieval* sistem, memastikan bahwa proses pengambilan konteks dari *vector store* tidak menimbulkan keterlambatan yang dapat dirasakan pengguna.

1.6.21 Formative evaluation

Formative evaluation (evaluasi formatif) adalah jenis evaluasi yang dilakukan selama proses pengembangan sistem dengan tujuan utama mengidentifikasi area perbaikan dan mengoptimalkan kinerja sistem sebelum digunakan secara luas (Scriven, 1967). Istilah ini pertama kali diperkenalkan oleh Michael Scriven dengan mendeskripsikan evaluasi yang berfokus pada *improvement* (perbaikan) daripada penilaian akhir.

Karakteristik Formative Evaluation Menurut Tessmer (2005), formative evaluation memiliki empat karakteristik utama:

1. *Improvement oriented* (Berorientasi Perbaikan): Tujuan utama adalah menemukan dan memperbaiki kelemahan, bukan memberikan penilaian final
2. *Iterative* (Berulang): Dapat dilakukan beberapa kali selama siklus pengembangan
3. *Contextual* (Kontekstual): Mempertimbangkan konteks dan kondisi penggunaan sistem
4. *Feedback driven* (Berbasis Umpan Balik): Umpan balik dari evaluator langsung digunakan untuk perbaikan.

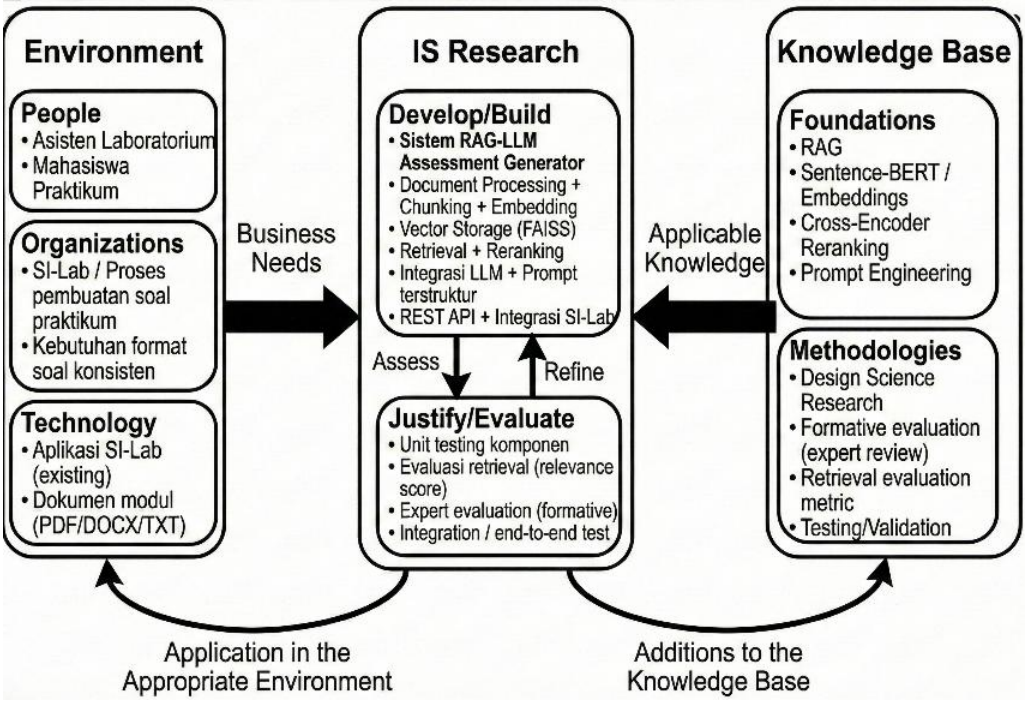
Prinsip Utama Dalam *formative evaluation* adalah evaluasi ulang setelah perbaikan tidak wajib dilakukan karena tujuan evaluasi sudah tercapai ketika area perbaikan teridentfikasi dan solusi telah diimplementasikan. Pendekatan ini mengutamakan efisiensi dalam siklus pengembangan (Stufflebeam & Coryn, 2014).

Dalam konteks penelitian ini, *formative evaluation* diterapkan untuk mengevaluasi kualitas soal yang dihasilkan sistem RAG-LLM, dimana *feedback* dari evaluator digunakan langsung untuk perbaikan sistem sebelum *deployment* operasional.

BAB II
METODE PENELITIAN

2.1 Jenis dan Pendekatan Penelitian

Penelitian ini merupakan penelitian pengembangan (*Research and Development*) dengan pendekatan *Design Science Research* (DSR). DSR merupakan paradigma penelitian yang berfokus pada penciptaan dan evaluasi artefak TI yang bertujuan untuk menyelesaikan masalah organisasional yang teridentifikasi (Hevner et al., 2004). Dalam konteks penelitian ini, artefak yang dikembangkan adalah sistem RAG-LLM untuk otomatisasi pembuatan soal praktikum. Seperti pada Gambar 16 berikut.



Gambar 16. Penerapan framework Design Science Research dalam penelitian ini

Pendekatan DSR dipilih karena beberapa pertimbangan. Pertama, penelitian ini bertujuan menghasilkan solusi teknologi yang dapat diterapkan untuk menyelesaikan masalah nyata di lingkungan laboratorium. Kedua, proses pengembangan sistem melibatkan siklus iteratif yang mencakup desain, implementasi, dan evaluasi. Ketiga, evaluasi artefak dilakukan melalui metrik kinerja teknis dan validasi praktis oleh pengguna.

2.2 Lokasi dan Waktu Penelitian

Penelitian ini dilaksanakan di Laboratorium Sistem Informasi, Departemen Matematika. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Hasanuddin, Makassar. Pemilihan lokasi didasarkan pada ketersediaan infrastruktur aplikasi SI-Lab yang menjadi platform target implementasi sistem.

2.2.1 Waktu penelitian

Penelitian dilaksanakan selama 4 Bulan, terhitung dari bulan September sampai dengan bulan Desember 2025.

2.3 Alat dan Bahan Penelitian

2.3.1 Perangkat keras

Tabel 10. Spesifikasi perangkat keras penelitian

Komponen	Spesifikasi
Processor	Intel Core i5 Gen 10
RAM	16 GB
Storage	SSD 512 GB
GPU	Nvidia Geforce GTX 1650

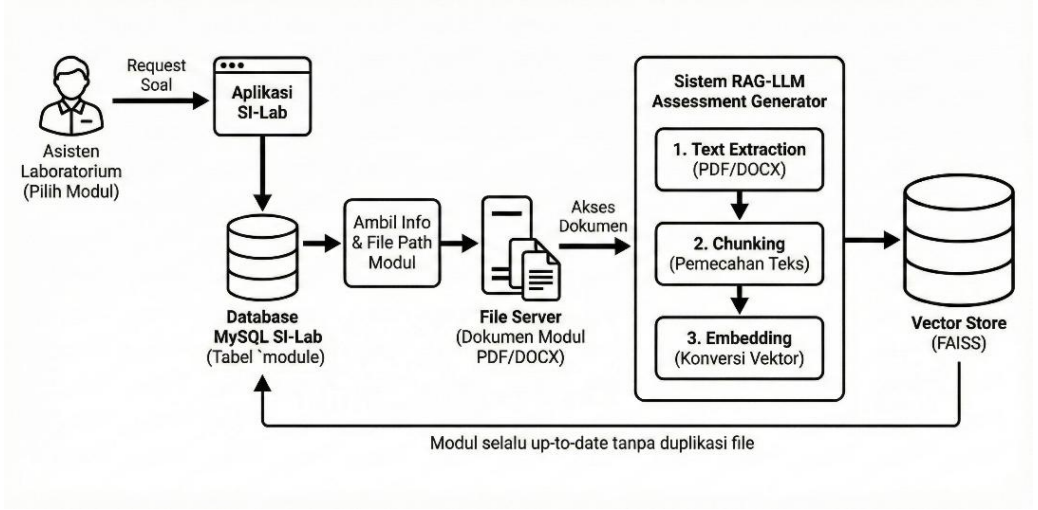
2.3.2 Perangkat lunak

Tabel 11. Teknologi dan framework yang digunakan

Komponen	Teknologi	Versi
Runtime	Python	3.10+
Web Framwork	Flask	3.0+
Database	MySQL	8.0+
Vector Store	FAISS	Latest
Embedding	Sentence-transformers	Latest
Reranker	*Cross-encoder*	ms-macro-TinyBert-L-2lv2
LLM Gateaway	OpenRouter	-
Text Extraction	pypdf, python-docx	Latest

2.3.3 Dataset

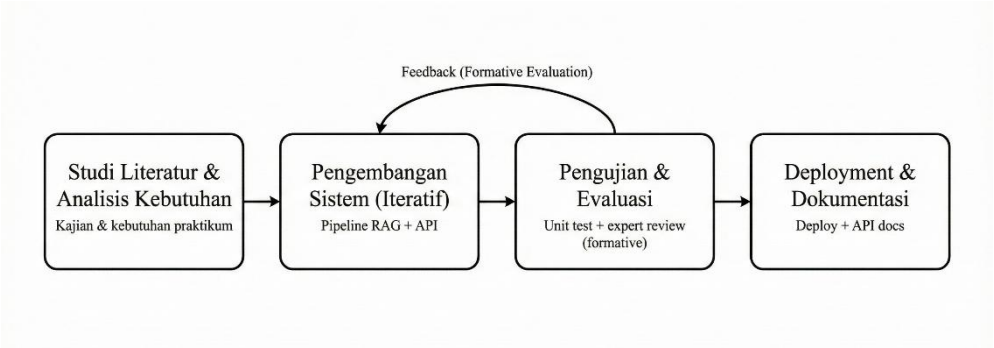
Dataset yang digunakan dalam penelitian ini berupa modul praktikum dari lima mata kuliah yang tersimpan dalam *database* (basis data) aplikasi SI-Lab. Modul-modul ini merupakan materi resmi yang digunakan dalam kegiatan praktikum di Laboratorium Sistem Informasi. Sistem mengakses modul secara dinamis dari *database* MySQL SI-Lab melalui table *module*, dimana setiap modul memiliki *file_path* yang mengarah ke dokumen PDF atau DOCX yang tersimpan di server.



Gambar 17. Diagram alur pengambilan modul dari database SI-Lab

Gambar 17 menunjukkan diagram alur pengambilan modul dari database SI-Lab yang dimulai dari permintaan (*request*) soal oleh Asisten Laboratorium melalui antarmuka aplikasi. Sistem kemudian mengakses Database MySQL SI-Lab untuk mengambil informasi serta jalur berkas (*file path*) modul, yang dilanjutkan dengan pengambilan dokumen fisik dari *File Server*. Dokumen tersebut kemudian diproses oleh Sistem RAG-LLM Assessment Generator melalui tiga tahapan utama: *Text Extraction* untuk membaca berkas PDF/DOCX, *Chunking* untuk pemecahan teks, dan *Embedding* untuk konversi ke bentuk vektor sebelum akhirnya disimpan ke dalam *Vector Store* (FAISS) guna memastikan data modul selalu mutakhir tanpa adanya duplikasi file.

2.4 Tahapan Penelitian



Gambar 18. Diagram alur tahapan penelitian

2.4.1 Studi literatur dan analisis kebutuhan

Tahap pertama penelitian dimulai dengan kajian literatur yang komprehensif tentang teknologi RAG, LLM, dan penerapannya dalam domain pendidikan. Kajian ini mencakup *paper-paper* fundamentel seperti Lewis et al. (2021) yang memperkenalkan konsep RAG, serta survey terbaru Li et al. (2025) tentang penerapan RAG dalam konteks

pendidikan. Pemahaman mendalam *terhadap state of the art* teknologi ini menjadi fondasi dalam perancangan sistem yang akan dikembangkan.

Selanjutnya, dilakukan analisis kebutuhan berdasarkan observasi langsung oleh peneliti yang juga berperan sebagai asisten laboratorium, serta diskusi informal dengan asisten lain yang menghadapi tantangan serupa dalam pembuatan soal praktikum. Hasil analisis menunjukkan bahwa proses manual memerlukan wkatu 30-60 menit per assessment dan seringkali menghasilkan format yang tidak konsisten antar asisten.

Berdasarkan hasil kajian literatur dan analisis kebutuhan, dilakukan identifikasi fitur dan spesifikasi teknis yang diperlukan. Fitur utama yang diidentifikasi, yaitu kemampuan memproses multi format dokumen, *retrieval* konteks yang akurat, generasi soal dengan format terstruktur, dan integrasi dengan sistem *existing* (yang sudah ada). Tahap ini diakhiri dengan perancangan arsitektur sistem yang terdokumentasi dalam diagram arsitektur (Gambar 19).

2.4.2 Pengembangan sistem

Pengembangan sistem dilakukan secara iteratif mengikuti prinsip *design science research*. Pada iterasi pertama, fokus diberikan pada implementasi komponen *document processing* yang mencakup *extractor* untuk berbagai format dokumen (PDF, DOCX, TXT), *semantic chunker* dengan algoritma yang mempertimbangkan batas paragraf dan kalimat, serta embedder menggunakan model all-MiniLM-L6-v2. Setiap komponen dikembangkan dan diuji secara modular sebelum diintegrasikan.

Iterasi kedua berfokus pada implementasi komponen *vector storage* menggunakan FAISS dengan index type FlatIP. Pada tahap ini, dikembangkan juga mekanisme *disk persistence* agar *index* dapat disimpan dan di *load* kembali tanpa perlu melakukan *re-indexing*. Hal ini penting untuk efisiensi operasional sistem di lingkungan produksi.

Iterasi ketiga mencakup implementasi komponen *retrieval* dengan *Cross-encoder reranking*. *Two-stage retrieval pipeline* dikembangkan dengan parameter yang dioptimalkan melalui eksperimen: initial_k dan top_k yang dioptimasi secara dinamis berdasarkan tingkat kesulitan soal, dan score_threshold=0.3. Model *cross-encoder* yang dipilih adalah ms-marco-TinyBERT-L-2-v2 yang memberikan keseimbangan antara akurasi dan kecepatan.

Iterasi keempat melibatkan integrasi LLM melalui OpenRouter API dengan teknik *prompt engineering*. Dikembangkan konfigurasi *prompt* yang spesifik untuk setiap mata kuliah, dengan *template* yang menghasilkan *output* terstruktur menggunakan tag SOAL, REQUIREMENTS, EXPECTED OUTPUT, dan KUNCI JAWABAN.

Pada iterasi kelima, dikembangkan REST API menggunakan Flask yang menyediakan *endpoint* untuk mengakses fungsionalitas sistem. API ini dirancang dengan mempertimbangkan kemudahan integrasi dengan frontend SI-Lab. Untuk menjaga keamanan dan stabilitas layanan, API dilengkapi dengan mekanisme keamanan yang meliputi: (1) *API Key Authentication* melalui header HTTP `X-API-Key` untuk mencegah akses tidak sah, dan (2) *Rate Limiting* dengan pembatasan 10 request per menit per pengguna untuk mencegah penyalahgunaan dan melindungi kuota API eksternal (OpenRouter). Terakhir, dilakukan integrasi dengan database MySQL SI-Lab untuk mengakses data mata kuliah dan modul.

2.4.3 Pengujian dan evaluasi

Pengujian sistem dimulai dengan *testing* (pengujan) fungsional pada setiap komponen secara individual. *Unit test* dikembangkan untuk memvalidasi fungsi *extractor* pada berbagai format dokumen, kebenaran algoritma *chunking*, konsistensi *embedding*, dan akurasi *retrieval*. Setiap komponen harus lulus *testing* sebelum diintegrasikan ke dalam *pipeline*.

Evaluasi performa *retrieval* dilakukan dengan memberikan *query* berupa pertanyaan terkait modul praktikum ke sistem, kemudian mengukur seberapa relevan *chunks* yang berhasil diambil. Sesuai dengan pendekatan evaluasi yang dijelaskan pada Section 2.6.1, evaluasi menggunakan *Relevance Score* dari *Cross-encoder* sebagai *proxy metric* untuk mengukur kualitas *retrieval*.

Evaluasi kualitas soal dilakukan melalui evaluasi *expert* dengan pendekatan *formative evaluation* (Scriven, 1967). Asisten laboratorium menilai setiap soal menggunakan rubrik dengan empat aspek, yaitu *relevance*, *structure*, *difficulty match*, dan *pedagogical value*. Hasil penilaian dikumpulkan dan dianalisis untuk mengidentifikasi area perbaikan sistem.

Berdasarkan temuan dari evaluasi *expert*, dilakukan perbaikan iteratif pada sistem. *Feedback* dari evaluator digunakan untuk menyempurnakan komponen-komponen yang memerlukan peningkatan. Sesuai dengan prinsip *formative evaluation* (Tessmer, 2005), evaluasi ulang setelah perbaikan tidak wajib dilakukan karena tujuan evaluasi sudah tercapai ketika area perbaikan teridentifikasi dan solusi telah diimplementasikan.

Pengujian integrasi dengan SI-Lab mencakup verifikasi format *request/response* API, validasi struktur JSON yang dikembalikan, dan *end-to-end testing* dari *input* parameter hingga diterimanya *response* soal oleh *frontend* SI-Lab. Sistem RAG-LLM berperan sebagai *assessment generator service* yang mengembalikan *response* ke aplikasi SI-Lab, sementara proses penyimpanan ke *database* ditangani oleh *backend* SI-Lab. *Feedback* dari pengguna awal (asisten laboratorium) juga dikumpulkan pada tahap ini.

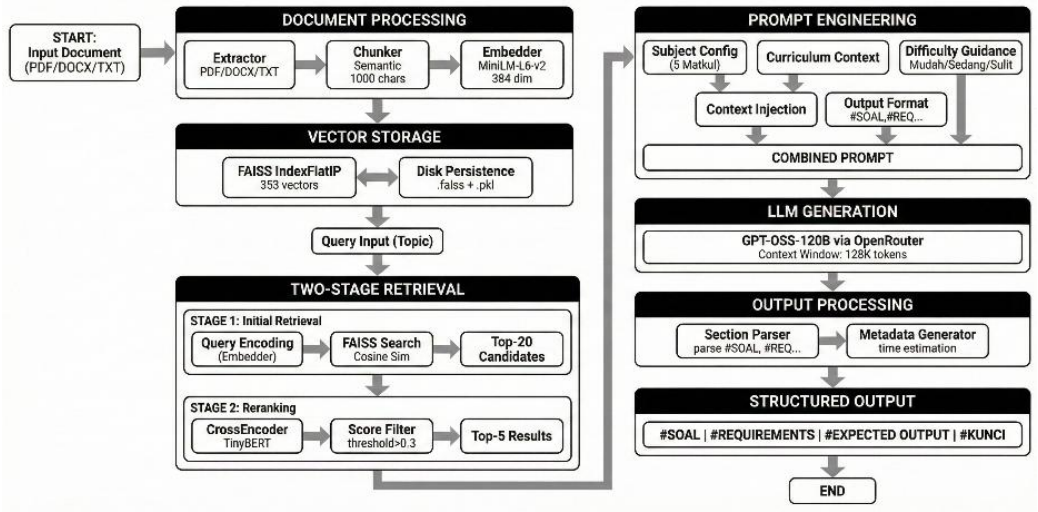
2.4.4 Deployment dan dokumentasi

Deployment sistem ke server SI-Lab dilakukan setelah seluruh tahap pengujian berhasil diselesaikan. Proses *deployment* mencakup konfigurasi *environment variables*, *setup database connection*, dan konfigurasi web server. Sistem di *deploy* sebagai *microservice* yang dapat diakses oleh aplikasi SI-Lab melalui REST API.

Penyusunan dokumentasi teknis mencakup API *documentation* yang menjelaskan setiap *endpoint*, format *request/response*, dan contoh penggunaan. Dokumentasi ini ditujukan untuk *developer* yang akan mengintegrasikan sistem dengan komponen lain dari SI-Lab.

2.5 Perancangan Arsitektur Sistem

2.5.1 Gambaran umum arsitektur



Gambar 19. Arsitektur sistem RAG LLM Assessment Generator

2.5.2 Komponen document processing

Komponen *document processing* bertanggung jawab untuk mengolah dokumen modul praktikum menjadi representasi yang dapat digunakan dalam proses *retrieval*. Komponen ini terdiri dari tiga subkomponen.

Text extraction. *Text extraction* merupakan proses pengambilan teks dari berbagai format dokumen. Sistem mendukung tiga format dokumen:

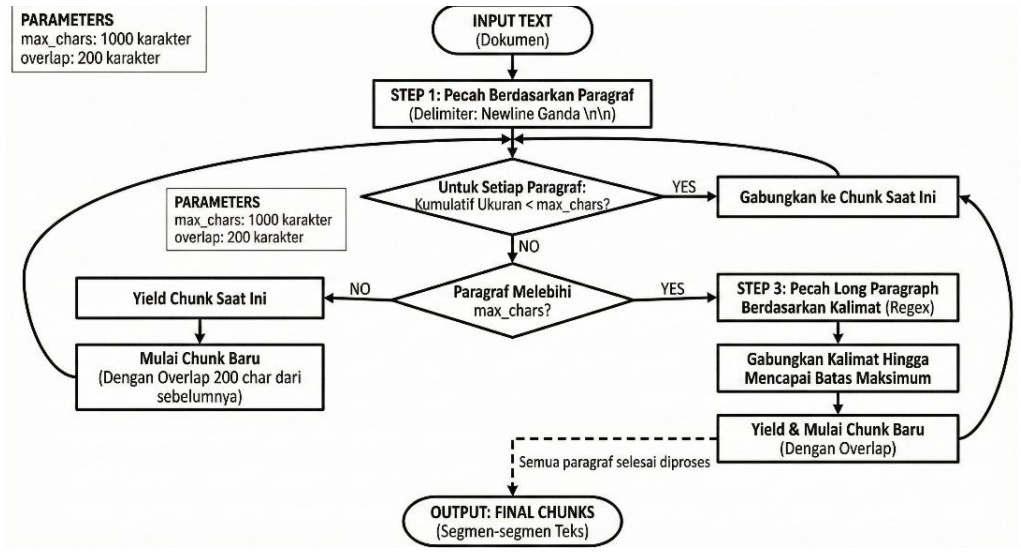
1. PDF: Menggunakan *library pypdf* untuk mengekstrak teks dari dokumen PDF. *Library* ini dipilih karena kemampuannya menangani berbagai jenis *encoding* dan struktur PDF
2. DOCX: Menggunakan *library python-docx* untuk mengekstrak teks dari dokumen Microsoft Word
3. TXT: Pembacaan langsung file teks dengan penanganan berbagai jenis *encoding*.

Semantic chunking. *Semantic chunking* merupakan proses pemecahan teks menjadi potongan-potongan yang lebih kecil dengan mempertimbangkan struktur semantik dokumen. Pendekatan ini lebih baik dibandingkan *fixed size chunking* karena menghindari pemotongan di tengah kalimat, mempertimbangkan batas paragraf, dan menjaga konteks yang lebih koheren.

Parameter chunking yang digunakan adalah ukuran maksimum 1000 karakter dengan overlap 200 karakter. Pemilihan ukuran chunk mengacu pada hasil eksperimen Wang et al. (2024) yang menunjukkan bahwa chunk size sekitar 256 tokens (~1000 karakter) memberikan tingkat relevancy tertinggi (97,78%) dibandingkan ukuran yang lebih besar atau lebih kecil. Overlap sebesar 200 karakter (20% dari ukuran chunk) digunakan untuk

memastikan kontinuitas konteks antar chunk, sesuai dengan best practice dalam sistem retrieval yang merekomendasikan overlap 10-20% dari ukuran chunk.

Pendekatan chunking berbasis karakter dipilih karena tiga pertimbangan utama. Pertama, pendekatan ini lebih sederhana secara implementasi karena tidak memerlukan library tokenizer tambahan. Kedua, pendekatan ini bersifat model-agnostic sehingga tidak bergantung pada tokenizer spesifik dari model tertentu. Ketiga, model embedding yang digunakan (all-MiniLM-L6-v2) memiliki kapasitas 512 tokens yang setara dengan sekitar 2000 karakter, sementara ukuran chunk dibatasi 1000 karakter sehingga masih berada jauh di bawah limit model. Token-based chunking akan lebih relevan untuk skenario dengan LLM yang memiliki batasan context window ketat atau untuk estimasi biaya API berbasis token, yang tidak menjadi pertimbangan dalam penelitian ini karena model LLM yang digunakan memiliki context window 128.000 tokens dan diakses melalui tier gratis.



Gambar 20. *Diagram flow* algoritma *chunking* yang diimplementasikan

Embedding. *Embedding* merupakan proses konversi teks menjadi representasi vektor numerik yang menangkap makna semantik. Sistem menggunakan model *sentence-transformers/all-MiniLM-L6-v2* dengan karakteristik:

Tabel 12. Karakteristik model sentence-transformers/all-MiniLM-L6-v2

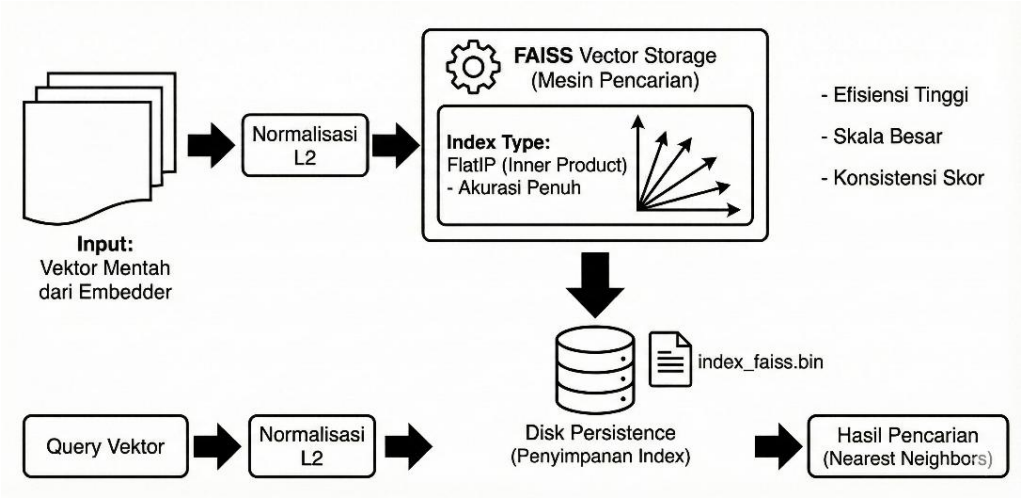
Parameter	**Nilai**
Dimensi vektor	384
Model size	~90 MB
Jumlah layer	6
Sequence length	256 tokens
Karakteristik	Multilingual, semantic similarity

Seperti yang ada di Tabel 12, model ini dipilih berdasarkan keseimbangan optimal antara performa dan efisiensi komputasi. Dengan ukuran file yang ringkas (~90 MB, dapat dikuantisasi hingga ~22 MB untuk int8) dan arsitektur yang efisien (6 layers, 384 dimensi), model ini memberikan response time yang sangat cepat untuk inferensi di CPU tanpa mengorbankan akurasi. Panjang sekuens maksimum 256 token cukup untuk meng-*encode* chunks modul praktikum setelah semantic *chunking*.

Kemampuan generalisasinya sangat baik untuk *corpus* modul praktikum yang balanced antara teks penjelasan dan kode pemrograman. Model ini juga mampu menangani konten *bilingual* (Indonesia-Inggris) yang umum pada materi teknis, serta telah production-ready dengan komunitas pengguna yang luas dan dokumentasi yang komprehensif.

2.5.3 Komponen vector storage

Vector storage menggunakan FAISS (*Facebook AI Similarity Search*) sebagai mesin pencarian vektor. FAISS dipilih karena kemampuannya melakukan pencarian *exact nearest neighbor* dengan efisiensi tinggi pada data berskala besar (Johnson et al., 2021).



Gambar 21. Konfigurasi FAISS yang diimplementasikan

Normalisasi L2. Model all-MiniLM-L6-v2 secara otomatis menerapkan normalisasi L2 pada output *embedding*. Normalisasi L2 adalah proses mengubah vektor agar panjangnya (magnitude) menjadi 1 tanpa mengubah arahnya. Proses ini krusial untuk *semantic search* karena dua alasan utama. Pertama, normalisasi memastikan bahwa *similarity measurement* murni berdasarkan arah vektor (semantic meaning) dan tidak dipengaruhi oleh *magnitude* vektor. Tanpa normalisasi, *chunks* dengan *semantic content* yang padat (banyak konsep berbeda) akan memiliki *magnitude* vektor yang lebih besar dibanding *chunks* yang repetitif atau mengandung banyak markup. Kedua, setelah normalisasi L2, *cosine similarity* menjadi ekuivalen dengan *dot product*, sehingga FAISS dapat melakukan pencarian similarity yang lebih efisien tanpa perlu menghitung pembagi magnitude secara eksplisit.

Mekanisme caching vector store. Sistem mengimplementasikan mekanisme *caching* untuk mengoptimalkan waktu respons pada *request* berulang. Cache disimpan dalam direktori data/*vectorstore* dengan dua file per modul, yaitu .faiss yang berisi FAISS index dan .pkl yang berisi metadata *chunks*.

Alur mekanisme:

1. *Cache check*: Ketika *request* diterima, sistem memeriksa apakah *cache* ada untuk modul yang diminta
2. *Cache Hit*: Jika cache tersedia, *vector store* langsung dimuat dari disk
3. *Cache Miss*: Jika *cache* belum tersedia, sistem menjalankan proses lengkap dan menyimpan hasilnya sebagai *cache*
4. *Cache Reuse*: *Cache* dapat digunakan kembali untuk *request* selanjutnya pada modul yang sama tanpa proses komputasi ulang.

Pendekatan ini sangat efisien karena proses *embedding generation* yang *computationally expensive* hanya dilakukan sekali per modul, sementara *request* selanjutnya dapat dilayani dengan latensi minimal.

2.5.4 Komponen retrieval dengan reranking

Berbeda dengan struktur arsitektur RAG standar yang hanya menggunakan *single stage retrieval* berbasis *similarity*, sistem yang dikembangkan dalam penelitian ini mengimplementasikan pendekatan *Advanced* RAG dengan *two stage retrieval*. Penambahan komponen *reranking* ini merupakan optimasi untuk meningkatkan presisi hasil *retrieval*, yang bukan merupakan bagian dari arsitektur RAG *original* (Lewis et al., 2021).

Pendekatan two-stage retrieval dengan *Cross-encoder reranking* diadopsi berdasarkan hasil penelitian terdahulu yang telah membuktikan efektivitasnya dalam meningkatkan kualitas *retrieval*. Nogueira & Cho (2020) menunjukkan bahwa penggunaan BERT sebagai reranker meningkatkan MRR (Mean Reciprocal Rank) pada dataset MS MARCO dari 0.277 (BM25 baseline) menjadi 0.365, peningkatan sebesar 31.8%. Santhanam et al. (2022) dalam ColBERTv2 mengkonfirmasi bahwa arsitektur two-stage retrieval secara konsisten mengungguli single-stage retrieval pada BEIR Benchmark. Dengan demikian, penelitian ini mengadopsi pendekatan *reranking* yang telah terbukti efektif pada berbagai *benchmark retrieval*, dengan ekspektasi bahwa peningkatan kualitas konteks akan berdampak positif pada kualitas soal yang dihasilkan.

Tahap 1: Initial retrieval. Pada tahap ini, sistem mengambil kandidat teratas (*initial_k)* berdasarkan *cosine similarity* antara vektor *query* dan vektor dokumen dalam FAISS. Jumlah kandidat yang lebih besar pada tahap awal memungkinkan *reranking* untuk mempertimbangkan lebih banyak opsi.

Tahap 2: Cross-encoder reranking. Kandidat yang diperoleh dari tahap pertama kemudian peringkatkan ulang menggunakan *cross-encoder.* Berbeda dengan *b-encoder* yang menghitung *embdedding query* dan dokumen secara terpisah, *cross-encoder* menghitung skor relevansi dengan memproses pasangan *query* dokumen secara bersamaan, menghasilkan penilaian yang lebih akurat (Santhanam et al., 2022).

Model *cross-encoder* yang digunakan adalah *cross-encoder/ms-macro-TinyBERT-L-2-v2.*

Dynamic retrieval berdasarkan tingkat kesulitan. Sistem mengimplementasikan penyesuaian parameter *retrieval* secara dinamis berdasarkan tingkat kesulitan soal yang diminta. Pendekatan ini dirancang untuk memastikan bahwa soal dengan kesulitan lebih tinggi memperoleh konteks yang lebih komprehensif.

Tabel 13. Konfigurasi Parameter *retrieval* per tingkat kesulitan

Tingkat Kesulitan	Initial K	Top K	Karakteristik Query
Mudah	15	5	Fokus pada konsep dasar dan contoh sederhana
Sedang	25	8	Materi berserta variasi dan konsep terkait
Sulit	50	15	Semua materi lengkap, seluruh variasi sintaks dan teknik

Seperti yang terlihat di Tabel 13, Untuk tingkat kesulitan “Sulit”, *query* dirancang untuk mengambil seluruh materi modul terkait topik, termasuk setiap sub-topik, variasi sintaks, dan implementasi yang ada.

2.5.5 Komponen LLM generation

LLM provider dan model. Sistem menggunakan OpenRouter sebagai *gateaway* API untuk mengakses model LLM. OpenRouter dipilih karena memberikan akses ke multiple model dengan satu API key dan mendukung *fallback* jika model utama tidak tersedia. Model utama yang digunakan adalah *openai/gpt-oss-120b:free.*

Prompt engineering. *Prompt engineering* merupakan aspek kritis dalam menghasilkan output yang berkualitas dari large language model. Desain prompt yang tepat dapat secara signifikan mempengaruhi relevansi, konsistensi, dan akurasi respons model terhadap kebutuhan domain spesifik (Brown et al., 2020; Reynolds & McDonell, 2021).

Sistem mengimplementasikan konfigurasi prompt yang terdiri dari sembilan komponen utama yang bekerja secara terintegrasi:

1. Konfigurasi per mata kuliah (Subject Configs)
2. Curriculum *aware context*
3. Identitas dan konteks
4. Instruksi tugas
5. *Dynamic difficulty guidance*
6. Spesifikasi format output
7. Kebebasan metode
8. Module *content injection*
9. *Custom notes* injection.

Sistem mengadopsi lima teknik prompting utama yang diintegrasikan dalam satu arsitektur prompt:

1. *Role prompting*: Mendefinisikan persona/peran AI untuk mengarahkan gaya respons (Reynolds & McDonell, 2021)
2. *Few shot prompting*: Memberikan contoh output yang diharapkan dan yang salah (Brown et al., 2020)
3. *Instruction Following with Constraints*: Pembatasan eksplisit untuk mengontrol output berdasarkan kurikulum (Ouyang et al., 2022)
4. *Structured output prompting*: Spesifikasi format output yang konsisten
5. *In-Context Learning*: Injeksi konteks relevan dari RAG Pipeline untuk meningkatkan relevansi output (Wei et al., 2023).

Struktur output. Pada Gambar 22 memperlihatkan *Output* LLM diformat dengan tag khusus untuk memudahkan *parsing.*

```
#SOAL
[Judul dan deskripsi lengkap soal praktikum]
→
#REQUIREMENTS
1. [Requirement teknis 1]
2. [Requirement teknis 2]
3. [Requirement teknis 3]
...
→
#EXPECTED OUTPUT
[Contoh output program yang diharapkan]
→
#KUNCI JAWABAN
[Source code lengkap sebagai jawaban referensi]
```

Gambar 22. *Output* format sistem untuk *query large language model*

2.6 Metode Evaluasi

2.6.1 Evaluasi performa retrieval

Evaluasi performa *retrieval* dilakukan untuk mengukur kemampuan sistem dalam mengambil konteks yang relevan dari modul praktikum sebelum dikirim ke LLM.

Normalisasi skor cross-encoder. *Cross-encoder* menghasilkan skor relevansi dalam bentu *logit* yang tidak memiliki batas, sehingga dapat bernilai negatif maupun positif tanpa range yang tetap. Untuk menginterpretasi skor ini sebagai probabilitas relevansi, digunakan fungsi sigmoid.

$$\boldsymbol{P(relevant) = \sigma(x) = 1/(1 + e^{-x})} \quad (1)$$

dimana:

- x = skor *Cross-encoder* (raw logit)
- e = bilangan Euler (≈ 2,71828)
- σ = fungsi sigmoid
- $P(relevant)$ = probabilitas relevansi (0-1).

Tabel 14. Normalisasi Skor cross-encoder

Cross-encoder Score	Sigmoid	P(relevant)	Interpretasi
-2,0	0,12	12%	Hampir pasti tidak relevan
0,0	0,15	50%	Netral (threshold)
2,0	0,88	88%	Kemungkinan besar relevan
4,0	0,98	98%	Hampir pasti relevan

Metrik evaluasi retrieval (Model Confidence Distribution). Evaluasi pada tahap ini difokuskan untuk mengukur distribusi keyakinan model. Evaluasi ini bertujuan mengetahui seberapa tinggi tingkat keyakinan (*confidence level*) sistem terhadap konteks yang diambil. Meskipun tidak menggantikan evaluasi akurasi berbasis *ground truth* manual, metrik ini krusial untuk memastikan bahwa *pipeline* RAG tidak bekerja secara acak dan memiliki keyakinan internal yang kuat terhadap dokumen yang dipilih.

Oleh karena itu, probabilitas relevansi digunakan sebagai indikator utama untuk mengukur seberapa tepat ("yakin") sistem bahwa konteks yang diambil sesuai dengan *query*, sementara *response time* mengukur efisiensi sistem, sebagaimana rincian parameter evaluasi yang disajikan pada Tabel 15 di bawah ini.

Tabel 15. Metrik Evaluasi Retrieval

Metrik	Definisi	Target
P(relevant)	Rata-rata probabilitas relevansi	≥ 70%
P(relevant) Top-1	Probabilitas relevansi hasil pertama	≥ 70%
Success Rate	Persentase query dengan P(relevant) ≥ 50%	100%

Metrik	Definisi	Target
Response Time	Waktu pemrosesan retrieval	< 100 ms

Interpretasi probabilitas relevansi. Interpretasi probabilitas relevansi mengadopsi prinsip probabilistik dalam information retrieval (Manning et al., 2009) yang menyatakan bahwa probabilitas di atas 0,5 mengindikasikan relevansi positif. Threshold yang digunakan dalam penelitian ini ditetapkan berdasarkan konvensi umum dalam evaluasi sistem retrieval:

Tabel 16. Interpretasi Probabilitas Relevansi

Range P(relevant)	Kategori	Interpretasi
≥ 90%	Sangat Relevan	Konteks sangat sesuai dengan query
70% - 89%	Relevan	Konteks sesuai dengan query
50% - 69%	Cukup Relevan	Konteks cukup sesuai, masih dapat digunakan
< 50%	Kurang Relevan	Konteks kurang sesuai dengan query

Seperti pada Tabel 16, Threshold 50% dipilih sebagai batas minimal karena nilai ini merepresentasikan titik netral pada fungsi sigmoid (σ(0) = 0,5), dimana model memiliki keyakinan yang sama antara relevan dan tidak relevan.

2.6.2 Evaluasi kualitas assessment

Kualitas soal yang dihasilkan dievaluasi dengan dua pendekatan komplementer, yaitu metrik otomatis dan evaluasi *expert*, mengikuti pendekatan evaluasi sistem generasi teks otomatis (Novikova et al., 2017).

Metrik otomatis.Evaluasi otomatis dilakukan melalui *parsing* dan analisis programatik terhadap output sistem. Metrik yang digunakan adalah *structure compliance*, yaitu kelengkapan komponen output yang dihasilkan. Sistem *parsing* secara otomatis memeriksa kehadiran tiga komponen utama (SOAL, REQUIREMENTS, EXPECTED OUTPUT) dalam setiap *output* dan menghitung persentase keberhasilan format. Komponen KUNCI JAWABAN bersifat opsional sehingga tidak menjadi kriteria wajib dalam penilaian struktur.

$$StructureCompliance = \frac{N_{lengkap}}{N_{total}} \quad (2)$$

dimana:

- $Nlengkap$ = jumlah output yang memiliki semua komponen
- $Ntotal$ = jumlah total output yang diuji.

Evaluasi expert. Evaluasi *expert* merupakan evaluasi kualitatif yang dilakukan oleh asisten laboratorium yang berpengalaman, mengikuti pendekatan human evaluation dalam penelitian NLP (van der Lee et al., 2019). Setiap evaluator menilai soal menggunakan rubrik penilaian dengan skala Likert 1-5 pada lima aspek (Likert, 1932).

a. Aspek *Relevance* menilai kesesuaian soal dengan materi modul yang menjadi sumber konteks (Maity et al., 2025). Evaluator mempertimbangkan apakah soal mencerminkan konsep-konsep yang terdapat dalam modul dan apakah konteks digunakan secara tepat
b. Aspek *Structure* menilai kelengkapan dan kejelasan komponen soal berdasarkan kriteria kualitas pertanyaan dalam *automatic question (Kurdi et al., 2020)*. Evaluator memeriksa apakah deskripsi soal jelas, *requirements* terperinci dan dapat diukur, *expected output* membantu pemahaman, dan kunci jawaban lengkap
c. Aspek *Difficulity Match* menilai kesesuaian tingkat kesulitan soal dengan parameter yang diberikan, mengacu pada Taksonomi Bloom untuk klasifikasi level kognitif (Anderson & Krathwohl, 2001). Untuk soal dengan parameter "mudah", evaluator menilai apakah soal memang sesuai tingkat pemula
d. Aspek *Pedagogical Value* menilai nilai pembelajaran dan kemampuan soal untuk mengukur kompetensi mahasiswa berdasarkan prinsip-prinsip *instructional design* (Merrill, 2002). Evaluator mempertimbangkan apakah soal melatih keterampilan yang relevan dan memiliki tingkat tantangan yang sesuai.

$$Overall = \frac{R+S+D+P}{4} \quad (3)$$

dimana:

- R = skor aspek relavance
- S = skor aspek structure
- D = skor aspek difficulty match
- P = skor aspek pedagogical value

Tabel 17. Rubrik evaluasi *expert*

Aspek	Kriteria Penilaian	Skala
Relevance	Kesesuaian soal dengan materi modul	1-5
Structure	Kelengkapan dan kejelasan komponen soal	1-5
Difficulity Match	Kesesuaian tingkat kesulitan dengan parameter	1-5
Pedagogical Value	Nilai pembelajaran dan kemampuan mengukur kompetensi	1-5

Pendekatan formative evaluation. Evaluasi *expert* dalam penelitian ini menggunakan pendekatan *formative evaluation* (Scriven, 1967), yaitu evaluasi yang berfokus pada identifikasi area perbaikan selama proses pengembangan, bukan penilaian akhir. Karakteristik utama pendekatan ini adalah orientasi pada perbaikan sistem bukan *judgment* akhir (Stufflebeam & Coryn, 2014).

Hasil evaluasi *expert* digunakan sebagai dasar untuk melakukan perbaikan iteratif pada sistem sebelum *deployment* operasional. Alur perbaikan yang diterapkan:

1. Generate sampel soal lalu *expert* mengevaluasi dengan rubrik
2. Identifikasi area perbaikan dari *feedback*
3. Implementasi perbaikan pada sistem
4. *Deploy* sistem *final* yang telah diperbaiki.

Sesuai dengan prinsip *formative evaluation*, evaluasi ulang setelah perbaikan tidak wajib dilakukan karena tujuan evaluasi sudah tercapai ketika area perbaikan teridentifikasi dan perbaikan telah diimplementasikan (Tessmer, 2005).

Sampel evaluasi. Untuk memastikan validitas hasil evaluasi, penelitian ini menggunakan sampel dengan komposisi sebagai berikut:

Tabel 18. Komposisi sampel evaluasi

Komponen	Jumlah	Keterangan
Request generasi soal	120 request	Data hasil sistem untuk analisis performa
Query untuk evaluasi retrieval	100 query	20 query per mata kuliah x 5 mata kuliah
Soal untuk evaluasi expert	26 soal	Sampel dari 5 mata kuliah dengan 3 tingkat kesulitan
Evaluator	9 orang	Asisten laboarotium SI-Lab
Total evaluasi expert	50 evaluasi	Setiap soal dievaluasi oleh minimal 1 evaluator

Pemilihan sampel dilakukan dengan mempertimbangkan representasi dari seluruh mata kuliah praktikum dan tingkat kesulitan yang berbeda (mudah, sedang, sulit). *Query* untuk evaluasi *retrieval* dirancang berdasarkan topik-topik utama yang terdapat dalam modul praktikum, sementara soal untuk evaluasi expert dipilih secara acak dari hasil generasi sistem.

2.6.3 Analisis Data

Prosedur pengumpulan data. Data penelitian dikumpulkan melalui beberapa tahapan:

1. Data *Retrieval*: Sistem dijalankan dengan 100 query yang telah dirancang, kemudian skor relevansi dari Bi-Encoder dan *cross-encoder* dicatat secara otomatis oleh sistem ke dalam file log
2. Data Generasi Soal: Sistem dijalankan untuk menghasilkan 120 request generasi soal, dengan mencatat waktu respons, status keberhasilan, dan output yang dihasilkan
3. Data Evaluasi *Expert*: Evaluator mengakses aplikasi evaluator berbasis Streamlit untuk menilai soal menggunakan rubrik yang telah disediakan. Hasil penilaian tersimpan secara otomatis dalam format JSON.

Prosedur pengolahan data. Data yang telah dikumpulkan diolah menggunakan Python dengan library pandas untuk manipulasi data dan numpy untuk perhitungan statistik:

1. Preprocessing: Data mentah dibersihkan dari entri yang tidak valid atau tidak lengkap
2. Agregasi: Data dikelompokkan berdasarkan mata kuliah, tingkat kesulitan, dan evaluator untuk analisis per kategori
3. Perhitungan Statistik: Statistik deskriptif dihitung menggunakan rumus standar
4. Visualisasi: Hasil disajikan dalam bentuk tabel dan grafik untuk memudahkan interpretasi.

Statistik deskriptif. Data hasil evaluasi dianalisis menggunakan beberapa pendekatan statistik sesuai dengan metodologi penelitian penelitian kuantitatif (Creswell & Creswell, 2018). Statistik deskriptif dihitung untuk setiap metrik, meliputi mean (rata-rata), standar deviasi, nilai minimum, dan nilai maksimum. Statistik ini memberikan gambaran umum tentang distribusi performa sistem.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4}$$

dimana:

- x_i = nilai data ke-i,
- n = jumlah total data,
- i = indeks data

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}} \tag{5}$$

dimana:

- σ = standar deviasi
- x_i = nilai data ke-i
- $\bar{x}$ = nilai rata-rata (mean)
- $(x_i - \bar{x})$ = selisih antara nilai data dengan rata-rata
- $(x_i - \bar{x})^2$ = kuadrat dari selisih tersebut
- n = jumlah total data.

Perbandingan performa antar konfigurasi sistem dilakukan untuk mengidentifikasi pengaturan optimal. Misalnya, perbandingan performa dengan dan tanpa *reranking*, atau perbandingan antar nilai parameter *chunking* yang berbeda. Hasil perbandingan disajikan dalam bentuk tabel dan grafik untuk memudahkan interpretasi.

Analisis kualitatif terhadap *feedback expert* dilakukan untuk mengidentifikasi pola kekuatan dan kelemahan sistem menggunakan pendekatan content analysis (Krippendorff, 2019). Komentar dan catatan dari evaluator dan dikategorisasi dan dianalisis untuk memberikan *insights* yang tidak dapat ditangkap oleh metrik kuantitatif.

Hasil analisis ini menjadi dasar untuk implementasi perbaikan sistem sesuai dengan alur *iterative improvement* yang telah dijelaskan.

2.6.4 Kriteria keberhasilan dan interpretasi skala

Untuk menentukan apakah sistem dapat dianggap berhasil dan bermanfaat, ditetapkan kriteria keberhasilan berdasarkan interpretasi skala evaluasi yang digunakan.

Interpretasi skala Likert. Hasil evaluasi expert menggunakan skala likert 1-5 diinterpretasikan berdasarkan rentang nilai rata-rata sebagai berikut (Vagias, 2006):

Tabel 19. Interpretasi Skala Likert

Rentang Nilai	Kategori	Interpretasi
≥ 1.00	Sangat Buruk	Tidak dapat diterima, perlu perbaikan fundamental
≥ 1.80	Buruk	Di bawah standar, memerlukan perbaikan signifikan
≥ 2.60	Cukup	Memenuhi standar minimum, dapat digunakan dengan perbaikan
≥ 3.40	Baik	Memenuhi ekspektasi, dapat digunakan dengan sedikit perbaikan
≥ 4.20	Sangat Baik	Melebihi ekspektasi, siap digunakan langsung

Rentang nilai pada Tabel diperoleh dari pembagian skala Likert 5-poin secara proporsional menjadi lima kategori dengan interval 0.8, yang merupakan praktik standar dalam interpretasi skala ordinal (Likert, 1932). Adapun label interpretasi (Sangat Buruk hingga Sangat Baik) mengadaptasi *Likert type scale response anchors* dari Vagias (2006).

Nilai 3.41 merupakan *cut off point* yang memisahkan respons netral/cukup (2.61-3.40) dengan respons positif/baik (3.41-4.20). Skor pada rentang ini merepresentasikan kecenderungan persetujuan (*agree*) dari evaluator terhadap kualitas yang dinilai.

Sistem dianggap berhasil jika rata-rata skor *overall* dari evaluasi *expert* mencapai minimal 3.41 (kategori Baik), yang menunjukkan bahwa soal yang dihasilkan memenuhi ekspektasi dan dapat digunakan dalam praktikum.

Evaluasi kualitas output. Untuk metrik otomatis terkait kualitas output, digunakan pendekatan deskriptif yang mengukur proporsi output yang memenuhi kriteria:

Tabel 20. Metrik kualitas *output*

Metrik	Deskripsi	Kriteria Keberhasilan
Structure compliance	Kelengkapan komponen output	Mayoritas (> 80%) output memiliki struktur lengkap

Kriteria “mayoritas” didefinisikan sebagai lebih dari 80% dari total sampel yang diuji, sesuai dengan prinsip bahwa sistem otomatis tetap memerlukan *review* manusia sebelum penggunaan akhir.

Response time threshold. Batas waktu respons (response time threshold) mengacu pada kriteria yang ditetapkan oleh Nielsen (1993) dalam bukunya Usability Engineering. Nielsen mengidentifikasi tiga batas waktu respons yang penting dalam desain sistem interaktif:

Tabel 21. Batas waktu respons dalam desain sistem interaktif (Nielsen, 1993)

Threshold	Waktu	Deskripsi
Instantaneous	0.1 detik (100 ms)	Pengguna merasakan respons sebagai instan; tidak ada jeda yang dirasakan
Uninterrupted	1 detik	Batas agar pengguna tetap fokus pada tugas tanpa merasa menunggu
Attention Limit	10 detik	Batas maksimum sebelum pengguna kehilangan perhatian dan meninggalkan sistem

Untuk sistem RAG-LLM assessment generator yang menghasilkan soal praktikum, task generasi soal bersifat asinkron (tidak memerlukan interaksi real-time). Oleh karena itu, batas waktu yang digunakan adalah 60 detik, yang masih berada dalam batas wajar untuk task asinkron sesuai prinsip responsivitas Nielsen.

Ringkasan kriteria keberhasilan sistem. Secara keseluruhan, sistem RAG LLM Assessment Generator dianggap berhasil dan bermanfaat jika memenuhi kriteria berikut:

Tabel 22. Ringkasan kriteria keberhasilan sistem

No	Metrik	Kriteria Keberhasilan
1	P(relevant) Retrieval	≥ 70% (kategori relevan)
2	Success Rate Retrieval	100% (P ≥ 50%)
3	Structure Compliance	Mayoritas output memiliki struktur lengkap (≥ 80%)
4	Expert Evaluation – Overall	Rata-rata skor expert ≥ 3,41 (kategori Baik)
5	End to end Success Rate	Persentase keberhasilan generate ≥ 95%
6	Avg. Response Time	< 60 detik untuk task asinkron
7	Deployment Status	Sistem aktif dan siap digunakan oleh SI-Lab

Sistem yang memenuhi seluruh kriteria di atas dapat direkomendasikan untuk penggunaan operasional di SI-Lab sebagai alat bantu pembuatan soal praktikum. Pendekatan perbandingan relatif yang digunakan memungkinkan evaluasi yang lebih objektif terhadap kontribusi spesifik dari penelitian ini, yaitu implementas.