BAB II
KAJIAN PUSTAKA

A. Kajian Teoritis

Konsep Dasar Sistem

Sistem merupakan sekumpulan komponen yang saling berhubungan dan bekerja bersama untuk mencapai tujuan tertentu. Dalam sistem informasi, komponen tersebut meliputi masukan, proses, keluaran, penyimpanan, pengguna, teknologi, serta mekanisme umpan balik yang saling mendukung. Metekohy dan Noya (2025:261) menjelaskan bahwa kualitas sistem informasi dipengaruhi oleh kualitas sistem, kualitas informasi, dan kualitas layanan yang mampu meningkatkan kepuasan pengguna. Selanjutnya, Aswidani (2024:9) menyatakan bahwa sistem informasi yang baik harus memiliki tingkat keandalan, kemudahan penggunaan, serta mampu menghasilkan informasi yang akurat sehingga memberikan manfaat bagi penggunanya..

Berdasarkan teori tersebut, sistem dalam penelitian ini dipahami sebagai kumpulan komponen teknologi yang saling terhubung untuk mengolah data tanaman herbal menjadi informasi yang bermanfaat. Sistem tidak hanya berfungsi sebagai tempat penyimpanan data, tetapi juga sebagai sarana pencarian, penelusuran relasi, rekomendasi edukatif, dan penyajian jawaban yang dapat ditelusuri sumbernya.

a. *Website*

Situs web merupakan media digital yang dapat digunakan untuk menyajikan informasi, layanan, dan interaksi melalui jaringan internet. Subiyakto et al. (2021:6) menjelaskan bahwa kualitas situs web berkaitan dengan kegunaan, kemudahan navigasi, tampilan antarmuka, serta kemampuan sistem dalam membantu pengguna mencapai tujuannya. Dalam penelitian ini, situs web digunakan sebagai media utama agar informasi obat herbal dapat diakses secara mudah oleh mahasiswa, peneliti, tenaga farmasi, dan masyarakat umum.

b. *Chatbot*

Bot percakapan atau *chatbot* adalah sistem berbasis kecerdasan buatan yang dirancang untuk berinteraksi dengan pengguna melalui bahasa alami. Laymouna et al. (2024:1) menjelaskan bahwa *chatbot* kesehatan dapat berperan dalam penyediaan informasi, edukasi, pendampingan digital, dan dukungan awal bagi pengguna. Dalam penelitian ini, *chatbot* digunakan sebagai antarmuka interaktif agar pengguna dapat bertanya mengenai tanaman herbal, manfaat, risiko, dan hubungan ilmiah secara lebih mudah.

Kecerdasan Buatan (*Artificial Intelligence*)

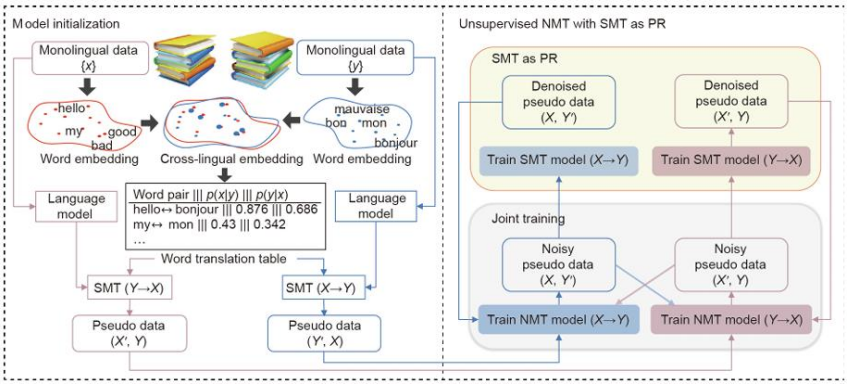
Kecerdasan buatan atau *artificial intelligence* merupakan bidang ilmu yang memungkinkan komputer meniru kemampuan manusia dalam mengenali pola, mengambil keputusan, memahami bahasa, dan menyelesaikan masalah. Secinaro et al. (2021:1) menjelaskan bahwa kecerdasan buatan banyak digunakan dalam layanan kesehatan, seperti

manajemen data, prediksi, diagnosis, dan dukungan keputusan klinis. Sejalan dengan itu, Alowais et al. (2023:11) menyatakan bahwa kecerdasan buatan mampu mempelajari data, melakukan penalaran, mengenali pola, serta mendukung proses pengambilan keputusan secara otomatis sehingga penerapannya semakin luas dalam berbagai layanan kesehatan digital.

Berdasarkan teori tersebut, kecerdasan buatan dalam penelitian ini digunakan untuk membantu sistem memahami pertanyaan pengguna, mencari informasi yang relevan, menelusuri hubungan antardata, serta menyusun jawaban edukatif. Dengan demikian, kecerdasan buatan tidak berdiri sendiri, tetapi bekerja bersama basis pengetahuan agar jawaban yang diberikan lebih terarah dan dapat ditelusuri.

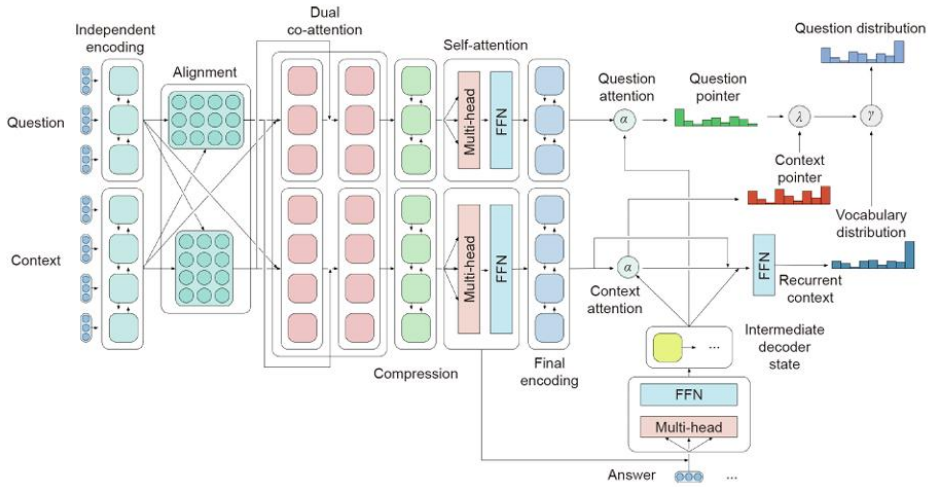
a. Pemrosesan Bahasa Alami (*Natural Language Processing* atau NLP)

Pemrosesan bahasa alami atau NLP adalah cabang kecerdasan buatan yang berfokus pada kemampuan komputer dalam memahami, memproses, dan menghasilkan bahasa manusia. Locke et al. (2021:5) menjelaskan bahwa NLP banyak digunakan dalam bidang kesehatan untuk mengolah teks bebas, mengekstraksi informasi, dan mendukung analisis dokumen medis. Dalam penelitian ini, NLP digunakan agar sistem dapat memahami pertanyaan pengguna dalam bahasa Indonesia dan menghubungkannya dengan data herbal yang tersedia.



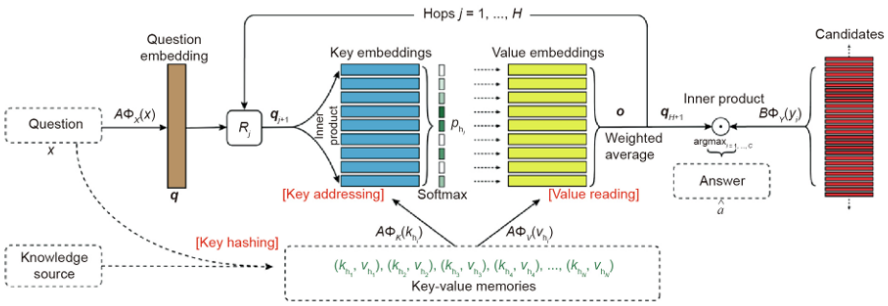
Gambar 2. 1 Alur Kerja *Unsupervised Neural Machine Translation* dalam NLP (Ren et al., 2020:243)

Gambar 2.1 menjelaskan proses *Unsupervised Neural Machine Translation* (NMT) dalam NLP, yaitu sistem penerjemahan mesin yang dilatih tanpa data pasangan terjemahan langsung. Sistem hanya menggunakan data monolingual dari dua bahasa, lalu membentuk *word embedding* dan *cross-lingual embedding* untuk mencocokkan kata yang memiliki makna serupa antarbahasa. Dari proses tersebut dibuat model awal berbasis SMT untuk menghasilkan pseudo data atau data terjemahan buatan. Pseudo data ini kemudian digunakan untuk melatih model NMT dua arah, yaitu bahasa X ke Y dan Y ke X. Karena data buatan masih bisa mengandung kesalahan, sistem melakukan proses perbaikan atau denoising agar data menjadi lebih bersih dan hasil terjemahan semakin akurat.



Gambar 2. 2 Arsitektur Model NLP untuk *Question Answering* (Zhou et al., 2020:8)

Gambar 2.2 menjelaskan arsitektur model NLP untuk *Question Answering*, yaitu sistem yang menerima pertanyaan dan konteks teks, lalu menghasilkan jawaban yang paling sesuai. Pertama, pertanyaan dan konteks diubah menjadi representasi vektor melalui proses encoding, kemudian dilakukan alignment untuk mencocokkan bagian konteks yang relevan dengan pertanyaan. Setelah itu, model menggunakan *dual co-attention* dan *self-attention* untuk memahami hubungan antara kata dalam pertanyaan dan konteks secara lebih mendalam. Hasil pemahaman tersebut diproses oleh *decoder* dengan mekanisme *question attention, context attention, dan pointer,* sehingga model dapat menentukan apakah jawaban diambil langsung dari konteks atau dihasilkan dari kosakata. Dalam NLP, arsitektur ini digunakan agar mesin mampu memahami teks, menemukan informasi penting, dan menjawab pertanyaan secara otomatis berdasarkan konteks yang diberikan.



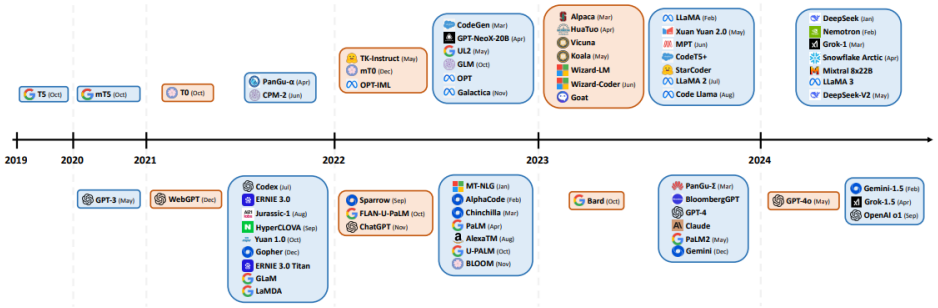
Gambar 2. 3 Arsitektur *Memory Network* dalam NLP untuk Tugas *Question Answering* (Zhou et al., 2020:12)

Gambar 2.3 menjelaskan arsitektur *Memory Network* dalam NLP untuk tugas *Question Answering*, yaitu sistem yang menjawab pertanyaan dengan mengambil informasi dari sumber pengetahuan. Pertanyaan terlebih dahulu diubah menjadi *question embedding*, kemudian dibandingkan dengan kumpulan memori berbentuk pasangan *key-value*. Bagian *key addressing* digunakan untuk mencari informasi yang paling relevan dengan pertanyaan melalui perhitungan kemiripan dan *softmax*, sedangkan bagian *value reading* mengambil isi informasi dari memori yang sesuai. Proses ini dapat dilakukan beberapa kali melalui beberapa *hops* agar model memperoleh konteks yang lebih lengkap. Setelah itu, hasil pembacaan memori digabungkan dengan representasi pertanyaan untuk memilih kandidat jawaban yang paling tepat. Dalam NLP, model seperti ini berguna untuk memahami pertanyaan, menelusuri pengetahuan, dan menghasilkan jawaban berdasarkan informasi yang tersimpan

b. Model Bahasa Besar (*Large Language Model* atau LLM)

Model bahasa besar atau LLM merupakan model kecerdasan buatan yang dilatih menggunakan data teks berskala besar untuk memahami konteks dan menghasilkan jawaban berbasis bahasa alami.

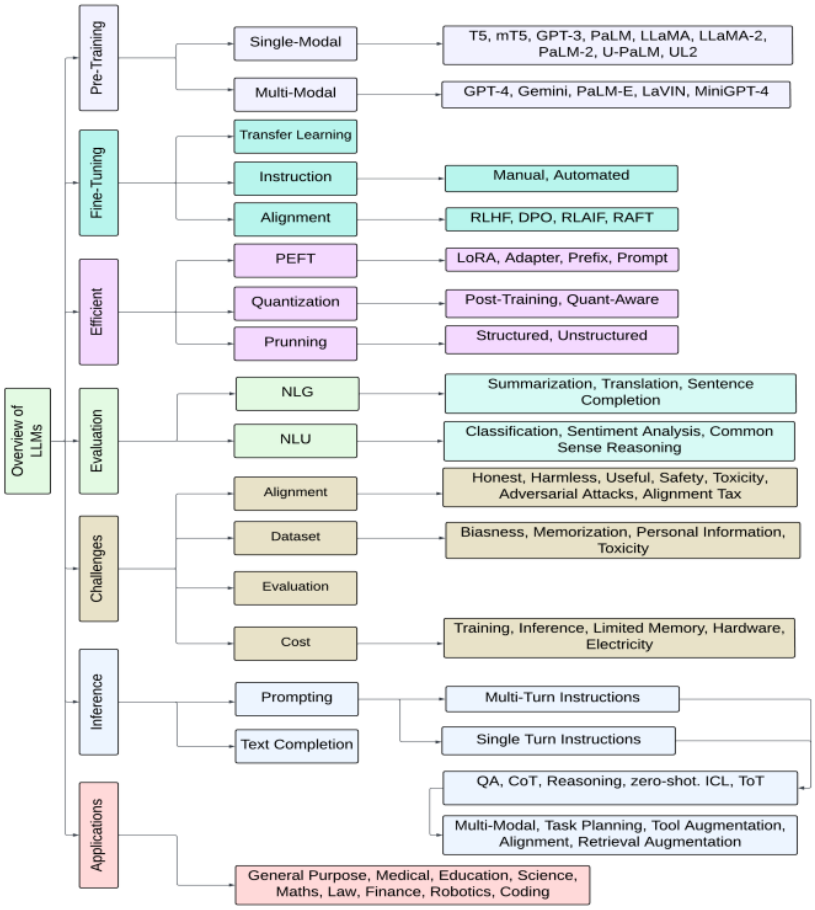
Haltaufderheide dan Ranisch (2024:2) menjelaskan bahwa *Large Language Model* memiliki potensi dalam analisis data, penyediaan informasi, dukungan pengambilan keputusan, serta berbagai aplikasi di bidang kesehatan. Namun demikian, model ini masih memiliki keterbatasan berupa bias, informasi yang tidak akurat (*hallucination*), dan kurangnya transparansi sehingga memerlukan pengawasan manusia. Oleh karena itu, penelitian ini menggunakan LLM dengan dukungan *GraphRAG* agar jawaban tidak hanya generatif, tetapi juga berbasis konteks pengetahuan yang dapat ditelusuri.



Gambar 2. 4 *Large Language Model* (LLM) dari Tahun 2019 hingga 2025 (Naveed et al., 2024:2)

Gambar 2.4 menunjukkan perkembangan *Large Language Model* (LLM) dari tahun 2019 hingga 2025 yang ditandai dengan kemunculan berbagai model besar seperti T5, GPT-3, WebGPT, Codex, ChatGPT, PaLM, LLaMA, GPT-4, Gemini, Claude, hingga DeepSeek. Dalam kajian NLP, LLM merupakan model kecerdasan buatan berbasis deep learning yang dilatih menggunakan kumpulan data teks berukuran sangat besar sehingga mampu memahami, menghasilkan, menerjemahkan, meringkas, dan menjawab pertanyaan dalam bahasa alami. Perkembangan pada gambar memperlihatkan bahwa LLM terus mengalami peningkatan dari

sisi skala parameter, kemampuan penalaran, pemahaman konteks, dukungan *multimodal*, serta penerapan pada berbagai bidang seperti *chatbot*, pencarian informasi, penulisan kode, dan sistem tanya jawab. Oleh karena itu, LLM menjadi salah satu teknologi utama dalam NLP modern karena mampu menghasilkan respons yang lebih kontekstual, fleksibel, dan menyerupai cara manusia berkomunikasi.



Gambar 2. 5 Ruang Lingkup LLM (Naveed et al., 2024:3)

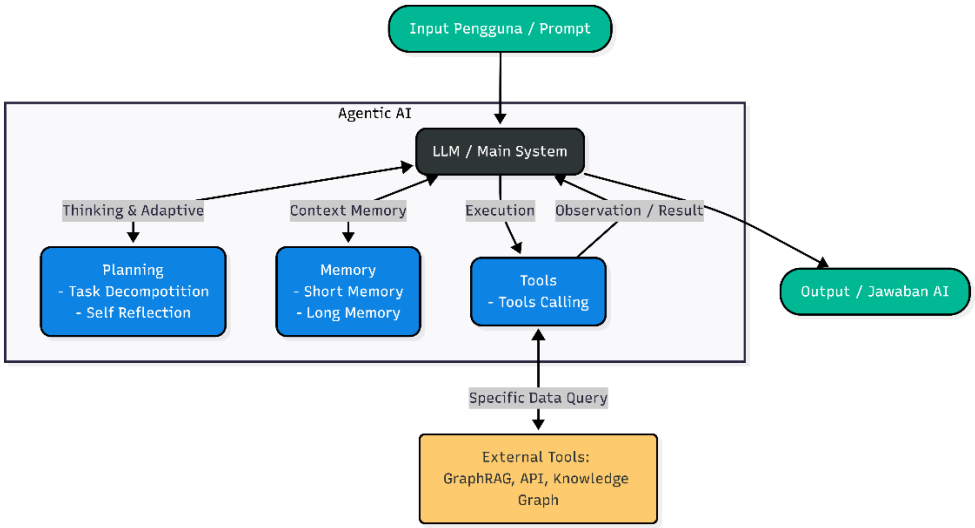
Gambar 2.5 menjelaskan ruang lingkup kajian *Large Language Model* (LLM) yang mencakup tahapan pengembangan, peningkatan efisiensi, evaluasi, tantangan, inferensi, dan penerapannya. Pada tahap *pre-training*, LLM dilatih menggunakan data berskala besar dalam bentuk

single-modal seperti teks maupun *multi-modal* yang menggabungkan teks, gambar, atau data lain, dengan contoh model seperti T5, GPT-3, PaLM, LLaMA, GPT-4, dan Gemini. Setelah itu, model dapat ditingkatkan melalui proses *fine-tuning*, seperti *transfer learning*, *instruction tuning*, dan *alignment* agar respon model lebih sesuai dengan instruksi pengguna serta aman digunakan. Gambar tersebut juga menunjukkan teknik efisiensi seperti PEFT, *quantization*, dan *pruning* untuk mengurangi kebutuhan komputasi tanpa menurunkan kinerja secara signifikan. Evaluasi LLM dilakukan pada aspek *Natural Language Generation* (NLG) dan *Natural Language Understanding* (NLU), seperti kemampuan merangkum, menerjemahkan, klasifikasi, analisis sentimen, dan penalaran umum. Selain itu, LLM masih memiliki tantangan terkait bias data, halusinasi, privasi, keamanan, biaya pelatihan, serta kebutuhan perangkat keras yang besar. Dalam penerapannya, LLM digunakan melalui *prompting* dan *text completion* untuk berbagai bidang, seperti pendidikan, kesehatan, sains, hukum, keuangan, robotika, dan pemrograman.

Sistem *Agentic AI*

Sistem *Agentic AI* merupakan pendekatan kecerdasan buatan yang memungkinkan model bertindak seperti agen, yaitu memahami tujuan, merencanakan langkah, menggunakan alat, memanfaatkan memori, dan mengevaluasi hasil. Wang et al. (2024:1) menjelaskan bahwa agen berbasis *Large Language Model* merupakan sistem otonom yang mampu memahami lingkungan, melakukan penalaran, menyusun perencanaan, mengambil keputusan, memanfaatkan berbagai alat (*tool use*), serta berinteraksi dengan

lingkungan untuk menyelesaikan tugas secara mandiri. Dalam penelitian ini, *Agentic AI* digunakan untuk mengatur alur kerja sistem mulai dari pemahaman pertanyaan, pencarian informasi, pemeriksaan keamanan, sampai penyusunan jawaban.



Gambar 2. 6 *Agentic AI Workflow* (Wang et al., 2024:2)

Gambar 2.6 mengilustrasikan komponen fundamental dan siklus alur kerja dari *Agentic AI* berbasis *Large Language Model* (LLM). Dalam arsitektur tersebut, LLM difungsikan sebagai "otak" (*brain*) pengontrol sentral yang berinteraksi secara dinamis dengan tiga modul eksternal utama:

a). Perencanaan (*Planning*): Modul yang memungkinkan agen untuk melakukan penalaran berantai (seperti metode *Chain-of-Thought*). Agen memecah permintaan pengguna yang rumit menjadi beberapa tahapan langkah logis dan melakukan refleksi diri (*self-reflection*) untuk mengevaluasi apakah rencana tersebut masuk akal.

b). Memori (*Memory*): Terdiri dari memori jangka pendek (menyimpan riwayat percakapan saat ini) dan memori jangka panjang (menyimpan

fakta atau pengetahuan masa lalu). Ini memastikan agen tidak kehilangan konteks di tengah proses penelusuran data yang panjang.

c). Penggunaan Alat (*Tool Use* / *Action*): Agen tidak sekadar menghasilkan teks, melainkan melakukan tindakan nyata. Agen difasilitasi kemampuan untuk memanggil fungsi (*function calling*), mengeksekusi skrip komputasi, atau menelusuri *Knowledge Graph* medis untuk mencari literatur spesifik.

Siklus *observasi-pemikiran-tindakan* (*observe-thought-action*) inilah yang memastikan sistem pada penelitian ini dapat menelusuri data fitofarmakologi secara cermat dan memverifikasi informasi obat herbal sebelum disajikan kepada pengguna.

Berdasarkan teori tersebut, *Agentic AI* dipahami sebagai pengatur proses berpikir sistem agar jawaban tidak langsung dihasilkan tanpa dasar. Agen berperan sebagai penghubung antara pertanyaan pengguna, *Knowledge Graph*, proses *retrieval*, dan model bahasa sehingga sistem dapat memberikan jawaban yang lebih relevan, runtut, dan sesuai kebutuhan pengguna.

a. *Model Bahasa Qwen (Qwen LLM*)

Qwen merupakan keluarga model bahasa besar yang dikembangkan untuk memahami instruksi, menjawab pertanyaan, melakukan penalaran, dan mendukung berbagai tugas berbasis bahasa. Yang et al. (2024:5) menjelaskan bahwa Qwen dikembangkan melalui proses *pre-training* dan *post-training* dengan peningkatan kemampuan pengetahuan umum, instruksi, dan penalaran. Dalam penelitian ini, *Qwen3-4B-Instruct-2507-Q6_K* digunakan sebagai model bahasa lokal

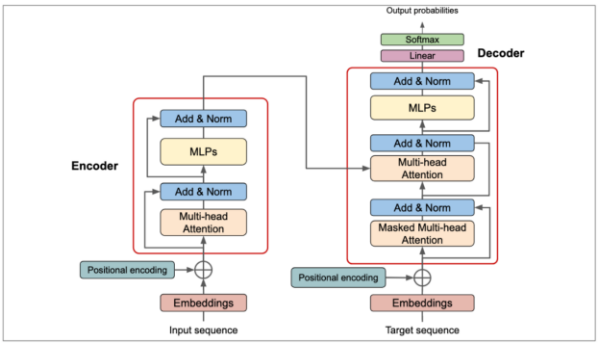
untuk menyusun jawaban berdasarkan konteks yang diperoleh dari *GraphRAG.*

b. Model Visi-Bahasa Qwen (*Qwen-VL)*

Qwen-VL merupakan model visi-bahasa yang dirancang untuk memahami teks dan gambar dalam satu sistem. Bai et al. (2023:3) menjelaskan bahwa Qwen-VL mampu melakukan pemahaman gambar, pembacaan teks visual, *grounding*, dan percakapan berbasis gambar. Dalam penelitian ini, model *Qwen3-VL-4B-Instruct-Q4_K_M* dan *mmproj-Qwen3VL-4B-Instruct-Q8_0* dapat digunakan untuk mendukung pengembangan sistem apabila terdapat kebutuhan membaca gambar tanaman, dokumen, atau informasi visual terkait herbal.

c. *Transformer*

Transformer adalah arsitektur pembelajaran mendalam yang menggunakan mekanisme *self-attention* untuk memproses hubungan antarkata atau token dalam suatu urutan teks. Vaswani et al. (2023:2) menjelaskan bahwa *Transformer* tidak bergantung pada struktur rekuren, tetapi memanfaatkan perhatian antarelemen sehingga lebih efisien untuk tugas bahasa. Arsitektur ini menjadi dasar penting bagi banyak LLM modern, termasuk model yang digunakan dalam sistem berbasis *Agentic AI.*



Gambar 2. 7 *Transformer Workflow* (Vaswani et al., 2023:3)

Gambar 2.7 menampilkan skema arsitektur Transformer yang terbagi menjadi bagian kiri (*Encoder*) dan kanan (*Decoder*). Pada bagian *Encoder*, input teks (*Inputs*) mula-mula melalui tahap *Input Embedding* dan *Positional Encoding* sebelum diproses ke dalam blok yang berisi lapisan *Multi-Head Attention* dan *Feed Forward*. Proses ini diulang sebanyak *Nx* kali (tergantung kedalaman model) untuk mengekstraksi makna mendalam. Pada bagian *Decoder*, hasil dari *Encoder* diterima melalui lapisan *Multi-Head Attention* sekunder (yang menghubungkan *Encoder-Decoder*). *Decoder* mengolah *Output (shifted right)* yang diberi perlakuan *Masked Multi-Head Attention* agar model tidak melihat token selanjutnya selama pelatihan. Hasil akhirnya melewati lapisan linear dan *Softmax* untuk mengeluarkan probabilitas - probabilitas prediksi kata berikutnya (*Output Probabilities).*

d. *Embedding*

Embedding adalah teknik representasi data yang mengubah kata, kalimat, dokumen, atau entitas menjadi vektor numerik agar dapat dihitung kedekatan maknanya oleh komputer. Asudani et al. (2023:1) menjelaskan bahwa *embedding* membantu model memahami hubungan semantik dalam

teks dan meningkatkan kinerja analisis bahasa. Dalam penelitian ini, *embedding* digunakan untuk membantu pencarian konteks dan pencocokan makna antara pertanyaan pengguna dengan data herbal.

e. *Chunking*

Chunking adalah proses memecah dokumen atau teks panjang menjadi bagian-bagian kecil agar informasi lebih mudah disimpan, dicari, dan digunakan dalam proses *retrieval*. Gomez-Cabello et al. (2025:2) menjelaskan bahwa strategi *chunking* yang tepat berpengaruh terhadap kualitas jawaban dalam sistem RAG karena potongan teks yang kurang tepat dapat mengurangi ketepatan konteks. Dalam penelitian ini, *chunking* digunakan untuk mengolah sumber referensi herbal agar dapat diambil kembali secara lebih efisien.

f. *Reasoning*

Reasoning adalah kemampuan sistem untuk menyusun langkah berpikir, menghubungkan informasi, dan menghasilkan kesimpulan berdasarkan data yang tersedia. Plaat et al. (2025:8) menjelaskan bahwa agen berbasis LLM tidak hanya menghasilkan teks, tetapi juga dapat melakukan penalaran, tindakan, dan interaksi. Dalam penelitian ini, *reasoning* digunakan untuk membantu sistem menjelaskan hubungan antara tanaman, senyawa aktif, aktivitas farmakologis, dan risiko penggunaan.

g. *Orchestrator*

Orchestrator adalah komponen pengatur yang mengelola alur kerja berbagai komponen dalam aplikasi LLM, seperti pemrosesan

pertanyaan, pemanggilan alat, pengambilan data, dan penyusunan jawaban. Plaat et al. (2025:27) menjelaskan bahwa dalam sistem *Agentic AI*, mekanisme orkestrasi (*orchestration*) berperan mengoordinasikan proses *reasoning*, *planning*, penggunaan *tools*, serta interaksi antarkomponen agar setiap tugas dapat dieksekusi secara terstruktur, efisien, dan sesuai dengan tujuan sistem Dalam penelitian ini, *orchestrator* berperan mengatur kerja agen, *Knowledge Graph*, *GraphRAG*, dan model bahasa.

h. Persona

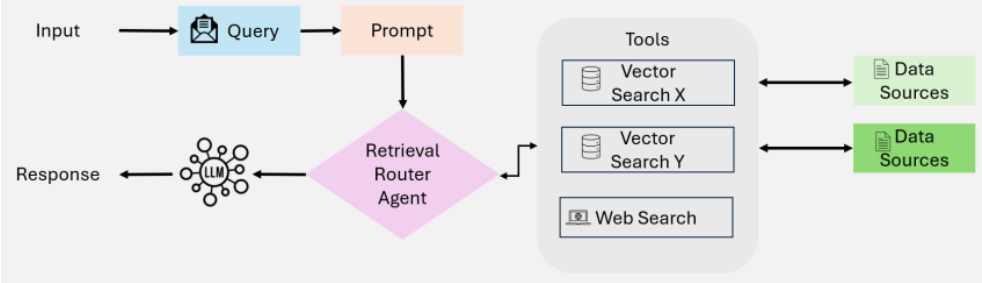
Persona dalam sistem percakapan digunakan untuk menyesuaikan gaya jawaban dengan karakteristik pengguna. Davar (2025:9) menjelaskan bahwa *chatbot* pendidikan perlu memperhatikan kebutuhan pengguna agar interaksi lebih sesuai dengan konteks pembelajaran. Dalam penelitian ini, persona digunakan agar jawaban dapat disesuaikan untuk masyarakat umum, mahasiswa, peneliti, maupun tenaga farmasi.

Graph Retrieval-Augmented Generation (GraphRAG)

Graph Retrieval-Augmented Generation atau *GraphRAG* merupakan pengembangan dari RAG yang memanfaatkan struktur graf untuk mengambil informasi berdasarkan hubungan antardata. Peng et al. (2024:3) menjelaskan bahwa *GraphRAG* menggabungkan proses pengindeksan berbasis graf, pengambilan informasi berbasis relasi, dan pembangkitan jawaban berbasis konteks. Pendekatan ini berbeda dari RAG biasa karena

tidak hanya mencari teks yang mirip, tetapi juga mempertimbangkan hubungan antarnode seperti tanaman, senyawa, penyakit, dan referensi.

Dalam bidang biomedis, *GraphRAG* dinilai penting karena informasi kesehatan sering memiliki hubungan yang kompleks. Soman et al. (2024:5) menunjukkan bahwa pendekatan *Knowledge Graph-based Retrieval-Augmented Generation* (KG-RAG) mampu mengintegrasikan *Large Language Model* dengan *Knowledge Graph* biomedis sehingga proses pengambilan informasi menjadi lebih terarah, menghasilkan jawaban yang didukung oleh pengetahuan terstruktur, serta menyertakan asal informasi (*provenance*) yang dapat ditelusuri. Wu et al. (2025:8) juga menjelaskan bahwa *Medical Graph RAG* dapat meningkatkan keamanan jawaban medis dengan memanfaatkan graf dan sumber tepercaya.



Gambar 2. 8 Alur Kerja *GraphRAG* (Parihar et al., 2026:11)

Gambar 2.8 menjelaskan alur kerja GraphRAG dalam sistem berbasis Large Language Model (LLM), yaitu proses pengambilan informasi yang lebih terarah sebelum model menghasilkan jawaban. Input dari pengguna diubah menjadi query dan prompt, kemudian diteruskan ke retrieval router agent yang berfungsi menentukan sumber informasi paling relevan. Agen ini dapat memilih beberapa alat pencarian, seperti vector search pada basis data tertentu maupun web search, yang terhubung dengan berbagai

data sources. Informasi yang berhasil diambil kemudian diberikan kepada LLM sebagai konteks tambahan agar jawaban yang dihasilkan lebih akurat, relevan, dan berbasis data. Dalam GraphRAG, proses retrieval tidak hanya bergantung pada kemiripan teks, tetapi juga dapat memanfaatkan struktur hubungan antar data atau pengetahuan, sehingga sistem mampu memahami konteks pertanyaan dengan lebih baik dan menghasilkan respons yang lebih terpercaya.

Berdasarkan teori tersebut, *GraphRAG* dalam penelitian ini digunakan sebagai metode pengambilan informasi dari *Knowledge Graph* sebelum jawaban diberikan kepada pengguna. Dengan cara ini, sistem diharapkan mampu memberikan jawaban yang lebih relevan, transparan, dan dapat ditelusuri dibandingkan model bahasa yang hanya mengandalkan pengetahuan hasil pelatihan.

Database dan Penyimpanan

Database dan penyimpanan berfungsi untuk mengelola data agar dapat disimpan, diakses, diperbarui, dan diamankan secara terstruktur. Freitas et al. (2025:801) menjelaskan bahwa manajemen data menjadi penting karena organisasi perlu mengelola informasi dalam jumlah besar secara efisien dan andal. Dalam penelitian ini, database dan penyimpanan digunakan untuk mengelola data pengguna, percakapan, dokumen referensi, data tanaman herbal, dan relasi pengetahuan.

Berdasarkan teori tersebut, database tidak hanya berperan sebagai tempat menyimpan data, tetapi juga sebagai fondasi agar sistem dapat berjalan konsisten. Penyimpanan yang baik membantu sistem menjaga

integritas data, mempercepat akses informasi, dan mendukung keterlacakan sumber jawaban.

a. *Graph Database*

Database graf adalah jenis basis data yang menyimpan data dalam bentuk node dan relasi. Coimbra et al. (2025:2) menjelaskan bahwa database graf sesuai untuk data yang kompleks dan saling terhubung, seperti jejaring sosial, bioinformatika, dan sistem rekomendasi. Dalam penelitian ini, database graf digunakan karena data herbal memiliki hubungan antarkomponen, misalnya tanaman, senyawa aktif, penyakit, efek samping, dan referensi.

b. *Knowledge Graph*

Knowledge Graph merupakan representasi pengetahuan yang menghubungkan entitas dan relasi sehingga data dapat dipahami secara kontekstual. Ji et al. (2021:8) menjelaskan bahwa *Knowledge Graph* digunakan untuk akuisisi pengetahuan, representasi, pelengkapan data, penalaran, dan aplikasi cerdas. Dalam penelitian ini, *Knowledge Graph* digunakan sebagai basis pengetahuan utama untuk menghubungkan informasi fitofarmakologi.

c. Neo4j

Neo4j adalah sistem manajemen database graf yang digunakan untuk menyimpan dan menelusuri data berbasis node dan relasi. Prataba et al. (2024:898) Neo4j digunakan untuk mengelola dan menyimpan data *database graph* yang berupa *node* dan *relationship* untuk memperoleh kompleksitas pengetahuan yang lebih mendalam. Neo4j digunakan untuk

merepresentasikan hubungan tanaman herbal, senyawa aktif, aktivitas farmakologis, penyakit, dan referensi ilmiah.

d. Supabase

Supabase merupakan platform *backend-as-a-service* berbasis PostgreSQL yang menyediakan fitur database, autentikasi, API, dan penyimpanan. Kolesnikov et al. (2024:100) menjelaskan bahwa Supabase sebagai *platform Backend as a Service* (BaaS) menyediakan layanan backend terkelola seperti autentikasi, basis data PostgreSQL, penyimpanan objek, serta API otomatis sehingga dapat mempercepat proses pengembangan aplikasi berbasis web. Dalam penelitian ini, Supabase digunakan untuk mengelola autentikasi, profil pengguna, riwayat percakapan, dan data aplikasi yang bersifat relasional.

e. MinIO

MinIO merupakan sistem penyimpanan objek yang kompatibel dengan Amazon S3 dan dapat digunakan untuk menyimpan file dalam skala besar. Chung dan Nakamura (2026:86) menjelaskan bahwa MinIO merupakan platform penyimpanan objek (*object storage*) yang bersifat *cloud-native* dan kompatibel dengan Amazon S3 sehingga mampu menyediakan penyimpanan data yang skalabel serta berkinerja tinggi untuk berbagai kebutuhan aplikasi modern. Dalam penelitian ini, MinIO digunakan untuk menyimpan dokumen, dataset, dan file pendukung yang dibutuhkan dalam pengembangan sistem.

Tanaman dan Obat Herbal

Tanaman obat herbal merupakan tanaman yang bagian tertentu, seperti daun, rimpang, batang, bunga, atau akar, digunakan untuk tujuan pemeliharaan kesehatan atau pengobatan tradisional. Kim et al. (2020:1077) menjelaskan bahwa obat tradisional memiliki peran penting dalam pelayanan kesehatan primer di berbagai negara, khususnya di negara berpendapatan rendah dan menengah, serta memerlukan dukungan penelitian ilmiah agar dapat diintegrasikan secara aman ke dalam sistem pelayanan kesehatan**..** Rahmat et al. (2021:1) menunjukkan bahwa temulawak (*Curcuma xanthorrhiza* Roxb.) mengandung berbagai senyawa bioaktif dan memiliki aktivitas farmakologis yang berpotensi dikembangkan lebih lanjut.

Obat herbal perlu dipahami secara hati-hati karena manfaat empiris belum selalu sama dengan bukti klinis. Balkrishna et al. (2024:36) menjelaskan bahwa pengembangan herbal modern membutuhkan kajian keamanan, efektivitas, bioaktivitas, serta standardisasi. Oleh karena itu, informasi herbal perlu memuat nama ilmiah, bagian tanaman, senyawa aktif, manfaat, risiko, dan rujukan ilmiah.

Tabel 2. 1 Sampel Data Tanaman Herbal dan Obat Herbal Indonesia

No.	Klasifikasi Obat Herbal (BPOM)	Bahan Baku Utama (Spesies)	Wilayah Distribusi Tumbuhan Asal	Senyawa Marka / Kandungan Standar	Indikasi Klinis / Khasiat Teruji
1	Fitofarmaka (Uji Klinis)	Meniran (Phyllanthus niruri)	Jawa, Sumatera (Dataran rendah tropis)	Filantin	Imunomodulator (Peningkat sistem kekebalan tubuh)

No.	Klasifikasi Obat Herbal (BPOM)	Bahan Baku Utama (Spesies)	Wilayah Distribusi Tumbuhan Asal	Senyawa Marka / Kandungan Standar	Indikasi Klinis / Khasiat Teruji
2	Fitofarmaka (Uji Klinis)	Kumis Kucing (Orthosiphon stamineus)	Sumatera, Jawa (Dataran tinggi)	Sinensetin	Antihipertensi dan Diuretik
3	Fitofarmaka (Uji Klinis)	Kayu Manis (Cinnamomum burmannii)	Sumatera Barat, Jambi, Jawa	Sinamaldehid	Terapi ajuvan Antidiabetes (Sensitizer insulin)
4	Obat Herbal Terstandar / OHT (Uji Praklinis)	Kulit Manggis (Garcinia mangostana)	Sumatera, Kalimantan, Jawa	Alfa-mangostin	Antioksidan sistemik dan Anti-inflamasi
5	Obat Herbal Terstandar / OHT (Uji Praklinis)	Daun Jambu Biji (Psidium guajava)	Merata di seluruh kepulauan Indonesia	Kuersetin	Antidiare (Menghambat motilitas usus)

Sumber: Kemenkes RI, (2023)

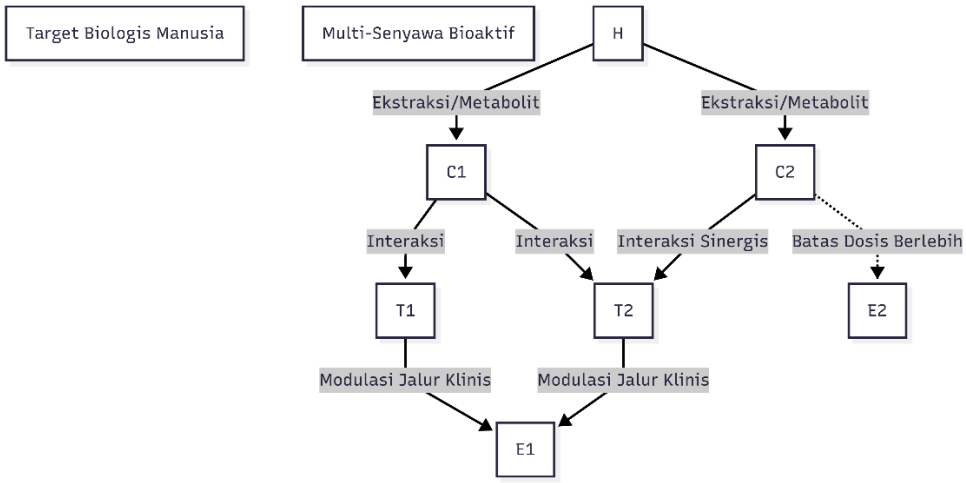
Berdasarkan tabel 2.1 tersebut, tanaman dan obat herbal dalam penelitian ini diposisikan sebagai objek pengetahuan yang perlu disajikan secara terstruktur. Sistem yang dikembangkan tidak hanya menampilkan manfaat, tetapi juga risiko, kontraindikasi, dan interaksi obat agar informasi yang diberikan lebih seimbang.

Fitofarmakologi

Fitofarmakologi merupakan kajian mengenai senyawa aktif dalam tanaman dan pengaruhnya terhadap aktivitas biologis atau farmakologis. Nugraha et al. (2022:1) menjelaskan bahwa penelitian fitokimia dan farmakologi pada tanaman obat dapat mengidentifikasi senyawa aktif serta aktivitas seperti antioksidan, antimikroba, antiinflamasi, dan efek biologis lainnya. Suwartiny et al. (2022:540) menjelaskan bahwa kajian fitokimia dan

fitofarmakologi diperlukan untuk memahami hubungan antara kandungan senyawa bioaktif tanaman dengan aktivitas biologis maupun potensi terapinya.

Kajian fitofarmakologi tidak hanya membahas manfaat, tetapi juga perlu memperhatikan keamanan. Kongkaew et al. (2024:11) menunjukkan bahwa pelaporan efek samping obat herbal masih bervariasi dan sering kali tidak disertai informasi komposisi produk yang lengkap. Hal ini menunjukkan perlunya sistem informasi yang dapat menyajikan manfaat, risiko, dan bukti ilmiah secara seimbang.



Gambar 2. 9 Skema Fitofarmakologi (Atanasov et al., 2021:201)

Skema pada gambar 2.9 merepresentasikan bagaimana ilmu fitofarmakologi modern bekerja menggunakan pendekatan jejaring (*network pharmacology*). Berbeda dengan obat kimia sintetis yang umumnya berfokus pada mekanisme "satu molekul, satu target penyakit", sebuah Tanaman Herbal memiliki Multi-Senyawa Bioaktif (misalnya Senyawa A dan Senyawa B). Senyawa-senyawa ini dapat berinteraksi secara bersamaan maupun bersinergi mengikat berbagai Target Biologis (reseptor atau enzim) di dalam

tubuh manusia. Modulasi dari target-target biologis inilah yang kemudian bermuara pada Efek Terapeutik (penyembuhan). Di sisi lain, skema ini juga menggarisbawahi pentingnya dosis, di mana senyawa bioaktif yang sama dapat memicu Efek Samping (Toksisitas) jika diakumulasi secara berlebihan. Arsitektur relasional inilah yang persis diadopsi dan dipetakan ke dalam bentuk *Knowledge Graph* pada sistem kecerdasan buatan dalam penelitian ini.

Berdasarkan teori tersebut, fitofarmakologi menjadi dasar penting dalam penelitian ini karena sistem yang dikembangkan perlu menghubungkan tanaman, senyawa aktif, aktivitas farmakologis, mekanisme kerja, efek samping, dan sumber ilmiah. Dengan demikian, informasi herbal dapat dipahami tidak hanya dari sisi tradisional, tetapi juga dari sisi ilmiah.

Framework

Framework merupakan kerangka kerja perangkat lunak yang menyediakan struktur, komponen, dan aturan untuk mempermudah pengembangan aplikasi. Setyautami et al. (2024:293) menjelaskan bahwa kerangka kerja pengembangan web dapat membantu penggunaan ulang komponen dan mempercepat pembangunan aplikasi. Dalam penelitian ini, *framework* digunakan untuk membangun sistem secara lebih terstruktur, baik pada sisi antarmuka, layanan API, maupun integrasi kecerdasan buatan.

Berdasarkan teori tersebut, *framework* dipahami sebagai fondasi teknis yang membantu pengembang membangun aplikasi lebih cepat dan konsisten. Penggunaan *framework* juga membantu sistem lebih mudah dikembangkan, diuji, dan dipelihara.

a. Python

Python merupakan bahasa pemrograman tingkat tinggi yang banyak digunakan dalam pengembangan kecerdasan buatan, analisis data, dan komputasi ilmiah. Ziogas et al. (2021:1) menjelaskan bahwa Python banyak digunakan karena produktif, mudah dibaca, dan didukung ekosistem pustaka ilmiah yang luas. Dalam penelitian ini, Python digunakan pada sisi *backend*, pemrosesan data, dan integrasi model kecerdasan buatan.

b. Next.Js

Next.js merupakan *framework* berbasis React yang mendukung pengembangan aplikasi web modern dengan kemampuan *server-side rendering* dan *static site generation.* Zaidan dan Suyatno (2025:2) menjelaskan bahwa Next.js melalui pendekatan Server Side Rendering (SSR) mampu meningkatkan performa aplikasi web, optimasi mesin pencari (SEO), dan pengalaman pengguna.. Dalam penelitian ini, Next.js digunakan untuk membangun antarmuka web yang responsif dan mudah digunakan.

c. FastAPI

FastAPI merupakan *framework* Python untuk membangun API modern yang mendukung validasi data, dokumentasi otomatis, dan pemrosesan asinkron. Bednarz dan Miłosz (2025:337) menjelaskan bahwa FastAPI merupakan framework Python berperforma tinggi yang mendukung pengembangan REST API secara efisien melalui pemrosesan asynchronous serta validasi data berbasis type hints. Dalam penelitian ini,

FastAPI digunakan sebagai layanan *backend* untuk menghubungkan antarmuka, database, model bahasa, dan *GraphRAG.*

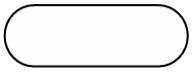
d. Llama.cpp

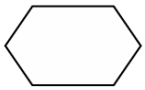
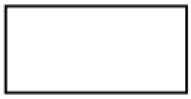
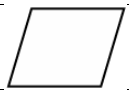
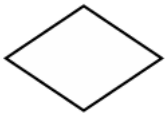
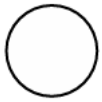
Llama.cpp merupakan pustaka sumber terbuka untuk menjalankan inferensi model bahasa besar secara lokal dengan dukungan kuantisasi. Kurt (2026:1) menjelaskan bahwa kuantisasi pada llama.cpp memungkinkan model bahasa dijalankan pada perangkat komoditas dengan kebutuhan memori yang lebih rendah. Dalam penelitian ini, llama.cpp digunakan untuk menjalankan model bahasa lokal agar sistem dapat diuji tanpa selalu bergantung pada layanan AI berbasis cloud.

Flowchart

Flowchart merupakan diagram yang digunakan untuk menggambarkan urutan proses, alur keputusan, dan langkah kerja dalam suatu sistem. Putra et al. (2022:239) menjelaskan bahwa flowchart digunakan untuk menggambarkan alur suatu proses atau program secara visual sehingga memudahkan dokumentasi, analisis, dan penyampaian informasi kepada pengguna maupun pengembang sistem.. Dalam pengembangan sistem, *flowchart* digunakan untuk memperjelas alur kerja mulai dari input pengguna, pemrosesan pertanyaan, pengambilan data, sampai keluaran jawaban.

Tabel 2. 2 *Flowchart*

Simbol	Nama	Fungsi
	Terminator	Permulaan/akhir program
	Garis Alir (*Flow Line*)	Arah aliran program

Simbol	Nama	Fungsi
	Preparation	Proses inisialisasi/pemberian harga awal
	Proses	Proses perhitungan/proses pengolahan data
	Input/Output data	Proses input/output data, parameter, informasi
	Predefined Process (Sub Program)	Permulaan sub program/proses menjalankan sub program
	Decision	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk Langkah selanjutnya
	On Page Connector	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	Off Page Connector	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda.

Sumber: Zalukhu et al., (2023:63)

Berdasarkan tabel 2.2, *flowchart* dalam penelitian ini digunakan sebagai alat bantu perancangan agar proses sistem mudah dipahami sebelum diimplementasikan. Dengan adanya *flowchart*, alur aplikasi dapat dijelaskan secara sederhana kepada pengembang, pembimbing, dan pengguna.

Unified Modeling Language (UML)

Unified Modeling Language atau UML merupakan bahasa pemodelan visual yang digunakan untuk menggambarkan struktur dan perilaku sistem perangkat lunak. Wayahdi dan Fahmi (2023:1514) menjelaskan bahwa UML dapat digunakan melalui *use case diagram*, *activity diagram*, *sequence diagram*, dan *class diagram* untuk memperjelas

kebutuhan serta rancangan sistem. Dalam penelitian ini, UML digunakan untuk memodelkan aktor, proses, interaksi, dan struktur data sistem.

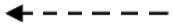
Berdasarkan teori tersebut, UML membantu perancangan sistem menjadi lebih terarah karena kebutuhan sistem dapat divisualisasikan sebelum tahap implementasi. UML juga memudahkan komunikasi antara peneliti, pengembang, dan pembimbing dalam memahami rancangan aplikasi.

a. Use Case Diagram

Use case diagram digunakan untuk menggambarkan hubungan antara aktor dan fungsi yang tersedia dalam sistem. Wayahdi dan Fahmi (2023:1515) menjelaskan bahwa *use case diagram* mendeskripsikan kebutuhan sistem berdasarkan interaksi pengguna. Dalam penelitian ini, diagram digunakan untuk menggambarkan peran admin dan user dengan persona umum, mahasiswa, peneliti, tenaga farmasi, dan admin, pada tabel 2.3.

Tabel 2. 3 *Use Case* Diagram

Simbol	Keterangan
	Aktor : mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan use case.
	Use Case : abstraksi atau interaksi antara system dengan aktor.
	Association : abstraksi dari penghubung antara actor dengan use case.
	Generalisasi : menunjukkan spesialisasi actor untuk dapat berpartisipasi dengan use cae.

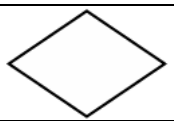
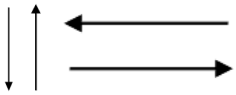
Simbol	Keterangan
<<include>> 	Includes : menunjukkan bahwa suatu use case seluruhnya merupakan fungsionalitas dari use case lainnya.
<<extend>> 	Extends : menunjukkan bahwa suatu use case merupakan tambahan fungsional dari use case lainnya jika suatu kondisi terpenuhi.

Sumber: Gunawan et al., (2023:56)

b. Activity Diagram

Activity diagram digunakan untuk menggambarkan alur aktivitas atau proses kerja dalam sistem. Wayahdi dan Fahmi (2023:1515-1516) menjelaskan bahwa diagram aktivitas membantu memperlihatkan urutan proses dari awal sampai akhir. Dalam penelitian ini, *activity diagram* digunakan untuk memodelkan alur pencarian informasi, percakapan, rekomendasi, dan pengelolaan data pada tabel 2.4.

Tabel 2. 4 *Activity* Diagram

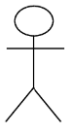
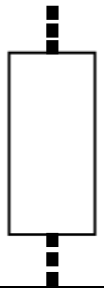
Simbol	Elemen	Keterangan
	Activity	Memperlihatkan bagaimana masing- masing kelas antarmuka saling berinteraksi satu sama lain.
	Action	State dari sistem yang mencerminkan eksekusi dari suatu aksi.
	Initial Node	Menunjukkan titik awal dari diagram aktivitas.
	Activity Final Node	Menunjukkan titik akhir dari diagram aktivitas.
	Decision	Digunakan untun menggambarkan suatu keputusan/tindakan yang harus diambil pada kondisi tertentu.
	Line Connector	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya.

Sumber: Gunawan et al., (2023:56)

c. *Sequence* Diagram

Sequence diagram digunakan untuk menggambarkan urutan interaksi antarobjek berdasarkan waktu. Wayahdi dan Fahmi (2023:1516) menjelaskan bahwa sequence diagram digunakan untuk memvisualisasikan interaksi dan pertukaran pesan antarobjek sehingga mempermudah pemahaman alur proses dalam sistem perangkat lunak. Dalam penelitian ini, diagram ini digunakan untuk menjelaskan interaksi antara pengguna, antarmuka web, API, database, *Knowledge Graph*, dan model bahasa pada tabel 2.5.

Tabel 2. 5 *Sequence* Diagram

Simbol	Elemen	Keterangan
	Actor	Menggambar orang yang sedang berinteraksi dengan sistem.
	Entity Class	Menggambarkan hubungan yang akan dilakukan.
	Boundary Class	Menangani komunikasi antar lingkungan sistem.
	Control Class	Menggambarkan penghubung antar boundary dengan tabel.
	A Focus of Control & A Life Line	Menggambarkan tempat mulai dan berakhirnya message.
	A Message	Menggambarkan pengiriman pesan.

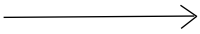
Sumber: Gunawan et al., (2023:56)

d. *Class* Diagram

Class diagram digunakan untuk menggambarkan struktur kelas, atribut, metode, dan hubungan antarobjek dalam sistem. Wayahdi dan

Fahmi (2023:1516) menjelaskan bahwa class diagram digunakan untuk memodelkan struktur statis sistem melalui representasi kelas, atribut, operasi, serta hubungan antarkelas sebagai dasar implementasi perangkat lunak.. Dalam penelitian ini, *class diagram* digunakan untuk menggambarkan struktur data seperti pengguna, tanaman, senyawa, penyakit, relasi, referensi, dan riwayat percakapan pada tabel 2.6.

Tabel 2. 6 *Class* Diagram

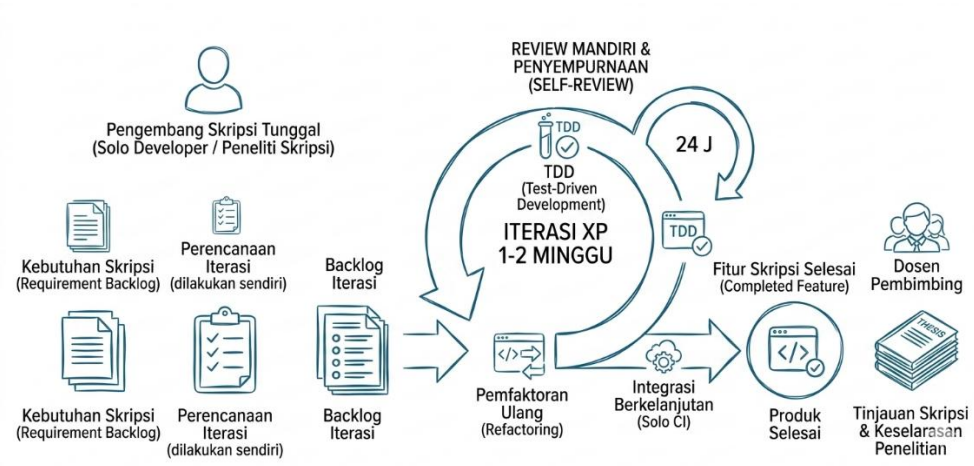
Simbol	Elemen	Keterangan
	Garis Lurus	Menunjukan hubungan objek anak (descendent) dan induk (ancestor) dalam hal berbagai perilaku dan struktur datanya.
	Nary Association	Suatu upaya untuk menghindari asosiasi yang melebihi 2 objek.
	Class	Suatu himpunan dari objek-objek dalam sistem, yang kemudian berbagi atribut dan operasi yang persis sama.
	Collaboration	Berupa urutan aksi-aksi dalam sistem agar menghasilkan sebuah hasil yang terukur. Sebuah.
	Realization	Sebuah operasi yang benar-benar dilakukan oleh objek dalam sistem.
	Dependency	Suatu hubungan pada perubahan yang terjadi dalam independent yang mempengaruhi elemen yang tidak mandiri.
	Association	Bagian yang menghubungkan objek yang satu dengan yang lainnya.

Sumber: Gunawan et al., (2023:56)

Extreme Programming

Extreme Programming atau XP merupakan metode pengembangan perangkat lunak berbasis *agile* yang menekankan komunikasi, umpan balik cepat, pengujian, dan perubahan bertahap. Sankhe et al. (2022:4) menjelaskan

bahwa *Extreme Programming* (XP) merupakan metodologi *Agile* yang berfokus pada keterlibatan pelanggan, iterasi singkat, serta praktik rekayasa seperti *continuous testing* dan *refactoring* untuk menghasilkan perangkat lunak berkualitas.. Selain itu, praktik XP mendukung pengembangan sistem yang adaptif terhadap perubahan kebutuhan pengguna.



Gambar 2. 10 *Extreme Programming* (Sommervile, 2020:86)

Gambar 2.10, secara teknikal dan manajerial dalam konteks penelitian skripsi oleh pengembang tunggal (*solo developer*), pendekatan XP diadaptasi dengan menitikberatkan pada kedisiplinan mandiri serta penerapan standar kualitas kode yang ketat. Meskipun pada dasarnya metode ini mengimplementasikan praktik kolaboratif seperti *Pair Programming* menurut penjabaran Sommerville (2020:77), untuk keperluan penelitian tunggal ini, praktik tersebut disesuaikan menjadi *Self-Review* (tinjauan mandiri dan penyempurnaan) guna mengidentifikasi dan menekan tingkat kesalahan secara independen. Praktik teknis ekstrem lainnya tetap diaplikasikan secara relevan, meliputi integrasi kode berkelanjutan yang dikelola secara mandiri (*Solo CI*) serta pemfaktoran ulang (*refactoring*).

Selain itu, kerangka kerja ini tetap berpegang teguh pada konsep *Test-Driven Development* (TDD), di mana skenario pengujian unit harus dirumuskan terlebih dahulu sebelum baris kode fungsional diimplementasikan. Pendekatan sistematis ini secara drastis meminimalisasi kemunculan *bug* dan memastikan bahwa setiap modul perangkat lunak yang dibangun memiliki stabilitas performa yang tinggi sejak awal, sehingga siap untuk dievaluasi dan ditinjau keselarasan penelitiannya oleh dosen pembimbing.

Dalam pelaksanaannya, siklus hidup metode XP dikelola melalui tahapan yang berulang (*iterative*) dan berkelanjutan. Pressman (2020:3) menguraikan bahwa alur kerja XP terdiri dari empat fase utama: *Planning* (perencanaan), *Design* (perancangan), *Coding* (pengodean), dan *Testing* (pengujian). Siklus ini diawali dengan proses *planning* berupa pengumpulan *user stories* (kebutuhan klien yang ditulis dalam bahasa bisnis), dilanjutkan dengan perancangan arsitektur sederhana menggunakan teknik *Class Responsibilities Collaborator* (CRC). Setelah *coding* diselesaikan dengan metode berpasangan, sistem langsung diuji secara ketat, dan fitur yang berhasil melewati tahap *testing* akan segera diintegrasikan untuk dirilis menjadi sebuah *software increment* (potongan perangkat lunak yang siap pakai).

Evaluasi Sistem

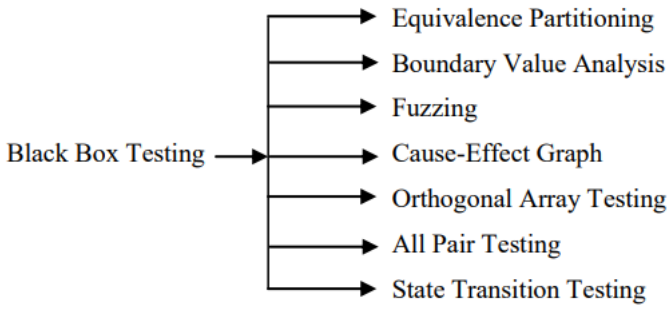
Evaluasi sistem merupakan proses untuk menilai apakah sistem telah berjalan sesuai kebutuhan pengguna dan tujuan pengembangan. Mahendra et al. (2022:2294) menjelaskan bahwa evaluasi aplikasi dapat dilakukan melalui pengujian fungsional dan pengukuran pengalaman pengguna. Dalam

penelitian ini, evaluasi dilakukan terhadap fungsi aplikasi, ketepatan *retrieval*, relevansi jawaban, keterlacakan sumber, konsistensi informasi, dan waktu respons.

Berdasarkan teori tersebut, evaluasi sistem diperlukan agar aplikasi tidak hanya berhasil dibuat, tetapi juga layak digunakan. Evaluasi membantu peneliti menemukan kelemahan sistem dan memperbaikinya sebelum digunakan oleh pengguna akhir.

a. *Black-box Testing*

Black-box testing adalah metode pengujian perangkat lunak yang berfokus pada fungsi sistem tanpa melihat struktur kode program. Mahendra et al. (2022:2293) menjelaskan bahwa metode ini cocok untuk memeriksa apakah masukan dan keluaran sistem telah sesuai dengan kebutuhan. Dalam penelitian ini, *black-box testing* digunakan untuk menguji fitur pencarian, percakapan, rekomendasi, login, dan pengelolaan data.



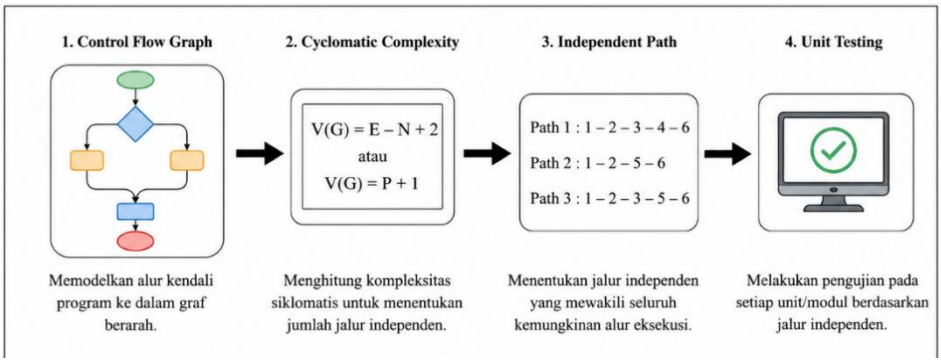
Gambar 2. 11 *Blackbox Testing* (Khan et al., 2021:33)

Gambar 2.11 menjelaskan bahwa *black-box testing* merupakan metode pengujian perangkat lunak yang berfokus pada pengujian fungsi sistem berdasarkan input dan output tanpa melihat struktur kode program

di dalamnya. Pada metode ini, penguji menilai apakah sistem sudah berjalan sesuai kebutuhan pengguna dan spesifikasi yang telah ditentukan. Beberapa teknik yang termasuk dalam *black-box testing* antara lain *equivalence partitioning* untuk membagi data uji ke dalam kelompok yang setara, *boundary value analysis* untuk menguji nilai batas, *fuzzing* untuk menguji sistem dengan input acak, *cause-effect graph* untuk menganalisis hubungan sebab-akibat, *orthogonal array testing* dan *all pair testing* untuk menguji kombinasi input secara efisien, serta *state transition testing* untuk menguji perubahan kondisi sistem. Dengan demikian, *black box testing* digunakan untuk memastikan bahwa fitur sistem berfungsi dengan benar dari sudut pandang pengguna.

b. *White-box Testing*

White-box testing adalah metode pengujian yang memperhatikan struktur internal kode, alur logika, dan jalur eksekusi program. Pamungkas et al. (2024:19) menjelaskan bahwa *white-box testing* membantu menguji logika program secara lebih mendalam, sedangkan *black-box testing* lebih berfokus pada keluaran sistem. Dalam penelitian ini, *white-box testing* digunakan untuk memeriksa alur fungsi penting seperti proses *retrieval*, pemanggilan model, dan pemeriksaan keamanan.



Gambar 2. 12 Control Flow Graph Whitebox Testing Method (Prasetya et al., 2025:1596)

Gambar 2.12, *White-box testing* merupakan metode pengujian perangkat lunak yang dilakukan dengan memperhatikan struktur internal, alur logika, dan kode program. Pada penelitian ini, metode *white-box testing* yang digunakan meliputi *control flow graph*, *cyclomatic complexity*, *independent path*, dan *unit testing*. *Control flow graph* digunakan untuk memodelkan alur kendali program ke dalam bentuk graf berarah, sehingga setiap proses, percabangan, dan alur eksekusi dapat dianalisis secara sistematis. Selanjutnya, *cyclomatic complexity* digunakan untuk menghitung tingkat kompleksitas program dan menentukan jumlah jalur independen yang perlu diuji. *Independent path* merupakan jalur-jalur dasar yang mewakili kemungkinan alur eksekusi dalam program. Setelah jalur independen ditentukan, pengujian dilakukan melalui *unit testing* untuk memastikan setiap unit atau modul program berjalan sesuai dengan fungsi dan spesifikasi yang telah ditentukan.

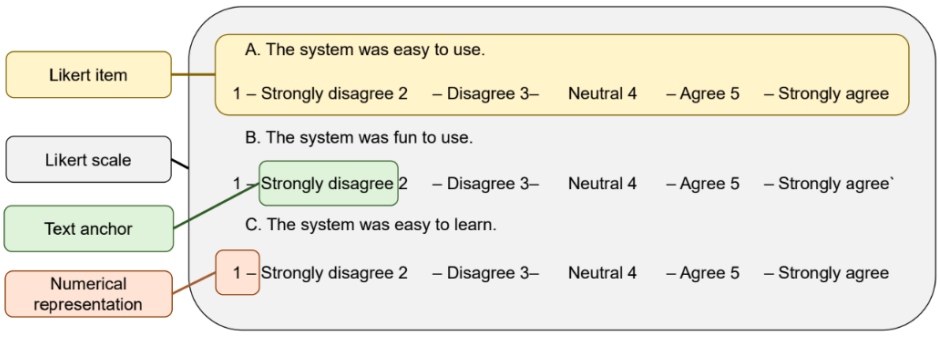
$$V(G) = P + 1 \tag{2.1}$$

Persamaan 2.1 merupakan salah satu cara menghitung *cyclomatic complexity* dalam metode *white-box testing*. *Cyclomatic complexit*y digunakan untuk mengukur tingkat kompleksitas logika program

berdasarkan jumlah jalur independen yang terdapat pada struktur kontrol program. Pada rumus tersebut, V(G) menunjukkan nilai kompleksitas siklomatis, sedangkan P adalah jumlah titik keputusan atau percabangan dalam program, seperti kondisi *if, else if, while,* atau *for*. Semakin besar nilai V(G), maka semakin banyak jalur pengujian yang perlu dilakukan untuk memastikan seluruh logika program telah diuji dengan baik. Dengan demikian, rumus ini membantu penguji menentukan jumlah minimal *test case* yang dibutuhkan dalam *white-box testing*.

c. Skala Likert

Skala Likert merupakan skala pengukuran yang digunakan untuk mengetahui tingkat persetujuan atau penilaian responden terhadap suatu pernyataan. Mahendra et al. (2022:2297) menggunakan instrumen penilaian pengguna untuk mengevaluasi penerimaan dan kegunaan aplikasi. Dalam penelitian ini, skala likert digunakan untuk menilai persepsi pengguna terhadap kemudahan penggunaan, kejelasan informasi, relevansi jawaban, dan manfaat sistem.



Gambar 2. 13 Komponen Utama Skala Likert (Salleh et al., 2026:552)

Gambar 2.13 menjelaskan komponen utama dalam skala Likert, yaitu metode pengukuran sikap, persepsi, atau pendapat responden

terhadap suatu pernyataan. Setiap item likert berisi pernyataan yang dinilai oleh responden, Dalam penelitian skripsi, skala likert sering digunakan karena memudahkan peneliti mengukur tingkat persetujuan responden terhadap indikator tertentu secara sistematis dan terukur pada tabel 2.7.

Tabel 2. 7 Bobot Nilai Skala Likert

No	Kategori Jawaban	Bobot Nilai
1	Sangat Setuju (SS)	5
2	Setuju (S)	4
3	Netral (N)	3
4	Tidak Setuju (TS)	2
5	Sangat Tidak Setuju (STS)	1

Sumber: Salleh et al., (2026:553)

Bobot nilai skala Likert 1–5 digunakan untuk mengukur tingkat persetujuan responden terhadap suatu pernyataan secara kuantitatif. Pada skala ini, setiap pilihan jawaban diberi nilai angka, yaitu 1 = Sangat Tidak Setuju (STS), 2 = Tidak Setuju (TS), 3 = Netral (N), 4 = Setuju (S), dan 5 = Sangat Setuju (SS). Dalam penelitian, skala likert 1–5 sering digunakan karena sederhana, mudah dipahami oleh responden, serta mampu menggambarkan sikap, persepsi, atau penilaian pengguna terhadap suatu objek penelitian secara sistematis pada tabel 2.8.

Tabel 2. 8 Keterangan Hasil Penilaian Skala Likert

No	Persentase	Kategori
1	0% - 20%	Sangat Tidak Layak
2	21% - 40%	Tidak Layak
3	41% - 60%	Cukup Layak
4	61% - 80%	Layak
5	81% - 100%	Sangat Layak

Sumber: Salleh et al., (2026:555)

Nilai akhir pengujian skala likert dapat diperoleh dengan menghitung persentase skor aktual dibandingkan dengan skor maksimal yang mungkin

diperoleh. Persentase tersebut kemudian digunakan untuk menentukan kategori kelayakan hasil penilaian. Secara umum, rentang persentase 0%–20% dikategorikan sangat tidak layak, 21%–40% tidak layak, 41%–60% cukup layak, 61%–80% layak, dan 81%–100% sangat layak. Dengan demikian, semakin tinggi persentase yang diperoleh, maka semakin baik tingkat kelayakan objek yang dinilai, sedangkan persentase yang rendah menunjukkan bahwa objek tersebut masih perlu diperbaiki atau dikembangkan lebih lanjut.

$$\text{Skor Aktual} = ((SS \times 5) + (S \times 4) + (N \times 3) + (TS \times 2) + (STS \times 1) \quad (2.2)$$

$$\text{Skor Ideal} = \text{Jumlah Responden} \times \text{Skor Tertinggi} \quad (2.3)$$

$$\text{Persentase Kelayakan} = (\text{Skor Aktual} / \text{Skor Ideal}) \times 100 \quad (2.4)$$

Perhitungan skala likert dilakukan dengan menentukan persamaan 2.2, persamaan 2.3, dan persamaan 2.4. Skor aktual diperoleh dari hasil penjumlahan setiap kategori jawaban responden yang dikalikan dengan bobot nilainya, yaitu Sangat Setuju (SS) bernilai 5, Setuju (S) bernilai 4, Netral (N) bernilai 3, Tidak Setuju (TS) bernilai 2, dan Sangat Tidak Setuju (STS) bernilai 1.

d. DeepEval

DeepEval merupakan *framework* evaluasi untuk aplikasi berbasis *Large Language Model* (LLM) yang dirancang untuk mengukur kualitas sistem secara otomatis melalui berbagai metrik evaluasi, seperti *Answer Relevancy*, *Faithfulness*, *Context Relevance*, *Context Precision*, *Context Recall*, dan *Hallucination*. Berbeda dengan pengujian konvensional yang hanya berfokus pada aspek fungsional, DeepEval mampu mengevaluasi apakah jawaban yang dihasilkan model benar-benar relevan terhadap

pertanyaan, didukung oleh konteks yang diambil, serta bebas dari informasi yang tidak didasarkan pada sumber (*hallucination*).

Tabel 2. 9 Grade DeepEval

Rentang Skor	Persentase	Grade
0,90–1,00	90 - 100%	A
0,80–0,89	80 – 89%	B+
0,70–0,79	70 – 79%	B
0,60–0,69	60 – 69%	B-
0,50-0,59	50 – 59%	C+
0,40–0,49	40 – 49%	C
0,30-0,39	30 – 39%	C-
0,20-0,29	20 – 29%	D
0,00-0,19	0 – 19%	E

Sumber Confident AI (2025)

Evaluasi kualitas sistem berbasis LLM dilakukan menggunakan DeepEval. Menurut Confident AI (2025), evaluasi aplikasi LLM dapat dilakukan menggunakan berbagai metrik untuk mengukur kualitas jawaban, relevansi konteks, serta kesesuaian respons terhadap input pengguna. Dalam penelitian ini, skor metrik DeepEval berada pada rentang 0–1, sedangkan interpretasi dalam bentuk grade A–E digunakan sebagai skema penilaian tambahan yang dirancang oleh peneliti untuk mempermudah interpretasi hasil evaluasi

B. Kajian Empiris

Penelitian mengenai integrasi model bahasa besar dengan basis pengetahuan terstruktur semakin banyak dikembangkan untuk mengurangi kelemahan *Large Language Model* (LLM), terutama pada domain yang membutuhkan ketepatan informasi. Edge et al. (2024:3) mengembangkan *GraphRAG* yang memanfaatkan *Knowledge Grap*h untuk memperkaya proses *retrieval* sehingga jawaban LLM menjadi lebih relevan dan mudah ditelusuri

sumbernya. Hasil penelitian tersebut menunjukkan bahwa penggunaan *Knowledge Graph* dapat membantu LLM menghasilkan jawaban yang lebih berbasis pengetahuan, memiliki *provenance*, dan mengurangi penggunaan token lebih dari 50% tanpa menurunkan akurasi. Temuan ini menunjukkan bahwa pendekatan berbasis graf relevan digunakan pada sistem kesehatan karena mampu menghubungkan entitas dan relasi secara lebih jelas.

Pada konteks Indonesia, Afifa et al. (2022:223-230) mengembangkan pemodelan ontologi tanaman obat Indonesia berbasis *Semantic Web*. Penelitian tersebut memanfaatkan data tanaman obat, penyakit, senyawa, efek samping, kontraindikasi, dan sumber lain untuk membangun representasi pengetahuan yang dapat ditelusuri melalui kueri SPARQL. Hasilnya menunjukkan bahwa informasi tanaman obat dapat direpresentasikan dalam bentuk relasi pengetahuan, tetapi aksesnya masih cenderung teknis sehingga belum sepenuhnya ramah bagi pengguna umum. Hal ini menjadi dasar bahwa sistem berbasis *Knowledge Graph* perlu dikembangkan dengan antarmuka yang lebih interaktif.

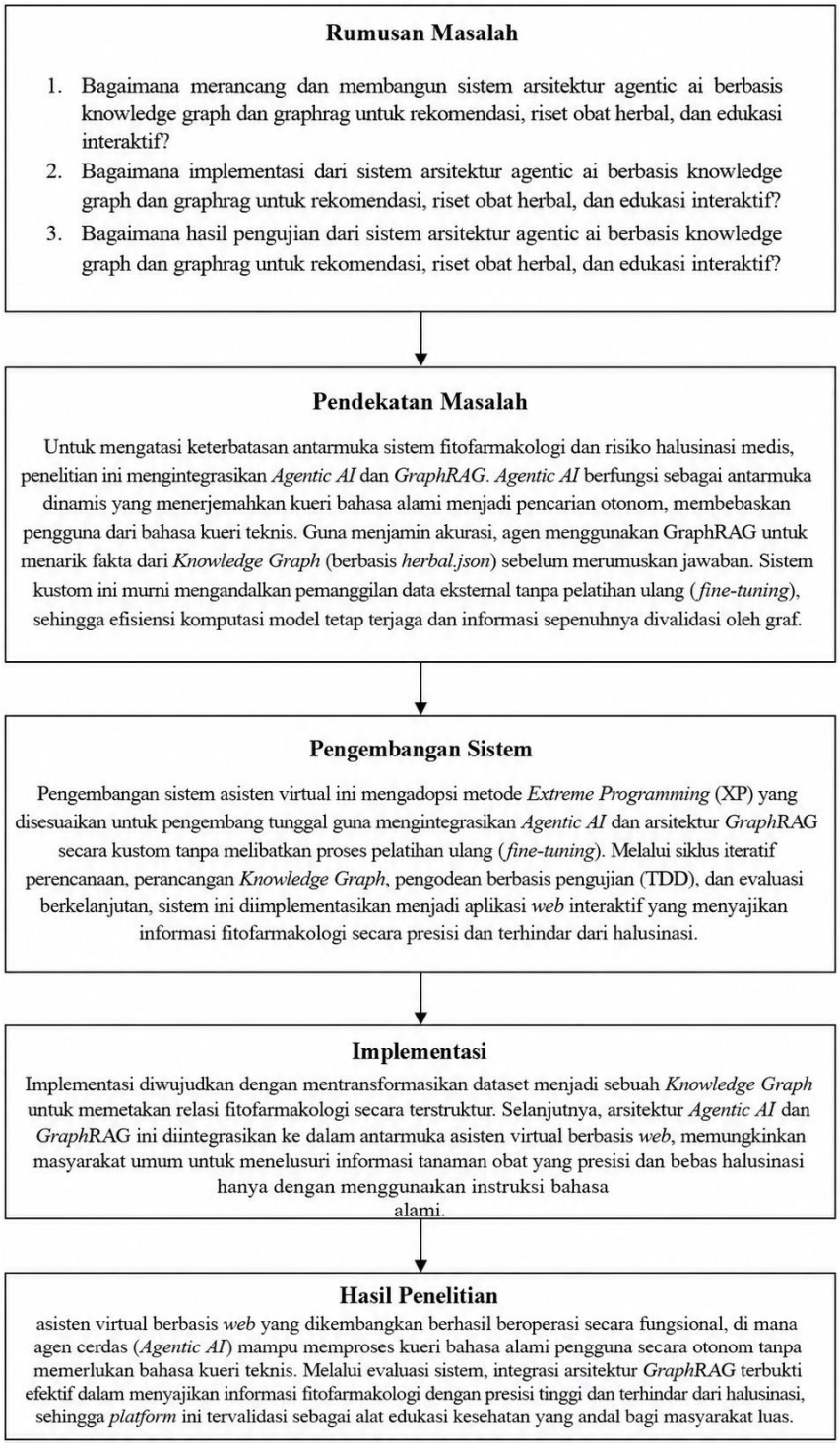
Penelitian lain juga menunjukkan bahwa *Retrieval-Augmented Generation* (RAG) mulai banyak digunakan dalam sistem tanya jawab. Chen et al. (2024:1) menerapkan *Retrieval-Augmented Generation* pada domain medis untuk meningkatkan akurasi jawaban dengan memanfaatkan dokumen eksternal yang relevan. Alfat et al. (2025:671) mengembangkan sistem *question answering* berbasis RAG pada domain pendidikan. Selain itu, Wu et al. (2025:44) menunjukkan bahwa pemanfaatan *GraphRAG* pada domain medis mampu meningkatkan ketepatan jawaban melalui integrasi *Knowledge Graph*

dan sumber informasi tepercaya.. Ketiga penelitian tersebut menunjukkan bahwa *chatbot* berbasis RAG dan NLP dapat membantu penyediaan informasi domain spesifik, tetapi belum berfokus pada pengetahuan obat herbal Indonesia yang menghubungkan tanaman, senyawa aktif, penyakit, efek samping, kontraindikasi, interaksi obat, dan referensi ilmiah.

Berdasarkan beberapa penelitian tersebut, dapat disimpulkan bahwa penggunaan LLM, RAG, dan *Knowledge Graph* telah terbukti bermanfaat untuk meningkatkan kualitas pencarian dan penyajian informasi. Namun, masih terdapat kesenjangan pada pengembangan sistem web interaktif yang secara khusus membahas obat herbal Indonesia dengan arsitektur *Agentic AI*, *Knowledge Graph*, dan *GraphRAG*. Oleh karena itu, penelitian ini diarahkan untuk membangun sistem yang tidak hanya menjawab pertanyaan pengguna, tetapi juga menelusuri relasi pengetahuan, memeriksa aspek keamanan, dan menyajikan informasi herbal secara edukatif, relevan, serta dapat ditelusuri sumbernya.

C. Kerangka Berpikir

Kerangka berpikir penelitian ini menggambarkan alur penyelesaian masalah dari perumusan masalah hingga hasil penelitian. Penelitian ini menerapkan *Agentic AI* yang dipadukan dengan *Knowledge Graph* dan *GraphRAG* guna memahami pertanyaan pengguna, menelusuri relasi pengetahuan, serta menghasilkan informasi yang relevan dan berbasis sumber. Pengembangan sistem dilakukan menggunakan metode *Extreme Programming* agar proses pembangunan aplikasi berjalan bertahap dan adaptif.



Gambar 2. 14 Kerangka Berpikir