

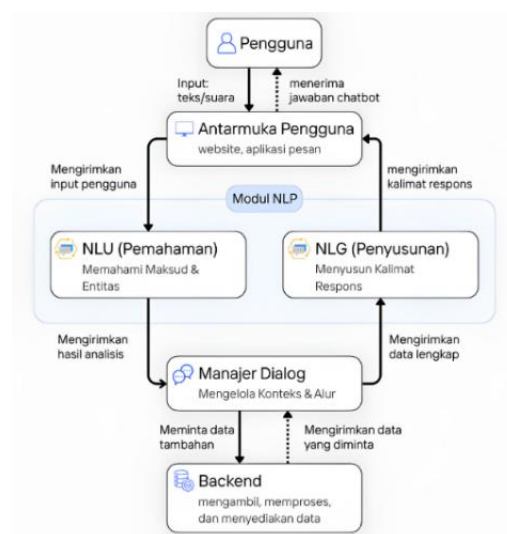
BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Chatbot

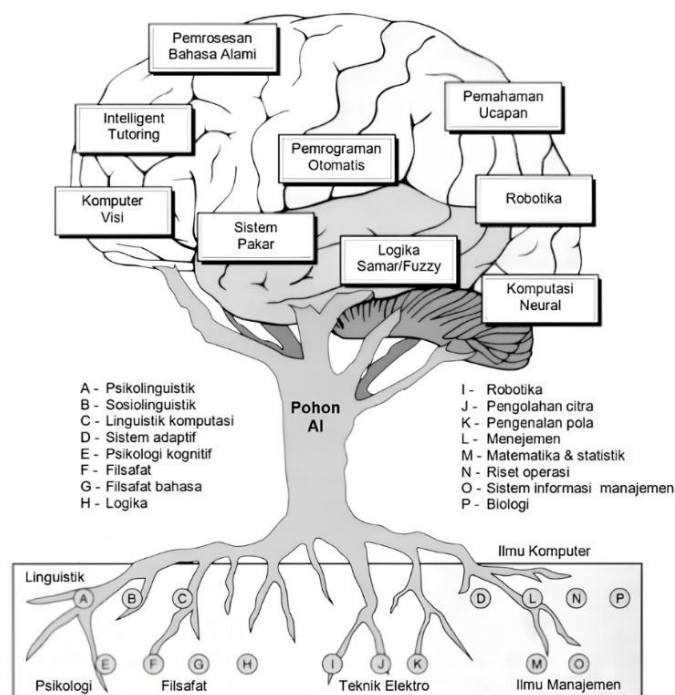
Chatbot merupakan perangkat lunak yang dikembangkan untuk mereplikasi komunikasi manusia melalui pemrosesan bahasa alami (Siahaan et al., 2023:1). Pengembangan chatbot dapat dilakukan melalui beberapa pendekatan, yaitu *Rule-based*, *Statistical-based*, *Machine Learning* dan *Hybrid* (Samperura et al., 2023:1). Menurut Suryanto et al. (2025:9) chatbot masa kini adalah ekosistem teknologi kompleks yang mengintegrasikan berbagai lapisan sistem dengan sruktur yang bekerja secara harmonis untuk menangkap pesan pengguna, membedah maknanya, dan mengelola alur percakapan demi memberikan respon yang tepat. Arsitekturnya mencakup 4 komponen inti pada gambar di bawah ini.



Gambar 2. 1 Arsitektur Chatbot
Sumber: Suryanto et al. (2025:9)

2. Artificial Intelligence

Artificial Intelligence (AI) atau Kecerdasan Buatan merupakan pada penciptaan sistem komputasi yang mampu menjalankan kemampuan kognitif setara manusia yang mengintegrasikan fungsi penalaran, pengambilan keputusan, dan pembelajaran berbasis pengalaman yang dirancang untuk menyelesaikan berbagai persoalan rumit (Parrangan et al., 2026:2). Menurut Mandowen et al. (2025:12) AI modern terbagi dalam lima cabang utama, yaitu *Machine Learning & Data Mining*, *Natural Language Processing* (NLP), *Computer Vision*, *Expert Systems* serta *Robotics & Autonomous Systems*. Pada kehidupan sehari-hari, AI dapat diimplementasikan dalam berbagai bidang seperti pada gambar berikut (Sudirwo et al., 2025:30).



Gambar 2. 2 Implementasi AI di berbagai bidang
Sumber: Sudirwo et al. (2025:30)

Berbeda dengan AI, *machine learning* yang merupakan sub-cabang AI menitikberatkan fungsinya pada pengembangan sistem yang mampu menganalisis dan mempelajari data (Juliani & Soleh, 2024). Melalui pemanfaatan data masa lalu, metode ini melatih sistem agar tingkat ketepatannya terus meningkat (Febriano & Rino, 2026). Machine learning bekerja dengan pembentukan model yang dirancang untuk memetakan *input* ke *output* (Lumbantoruan et al., 2026:3). Dalam proses ini dikenal sebagai *training*, model mempelajari parameter dari data historis untuk meminimalkan kesalahan prediksi. Hasilnya adalah model terlatih yang kemudian dapat digunakan pada fase inference untuk memproses data baru dan menghasilkan keluaran berupa prediksi atau keputusan. *Machine Learning* terbagi menjadi tiga kategori utama:

1. *Supervised Learning*

Metode pembelajaran yang menggunakan data berlabel untuk tugas seperti regresi dan klasifikasi. Contohnya adalah SVM (*Support Vector Machine*), *Decision Tree*, dan *Random Forest*.

2. *Unsupervised Learning*

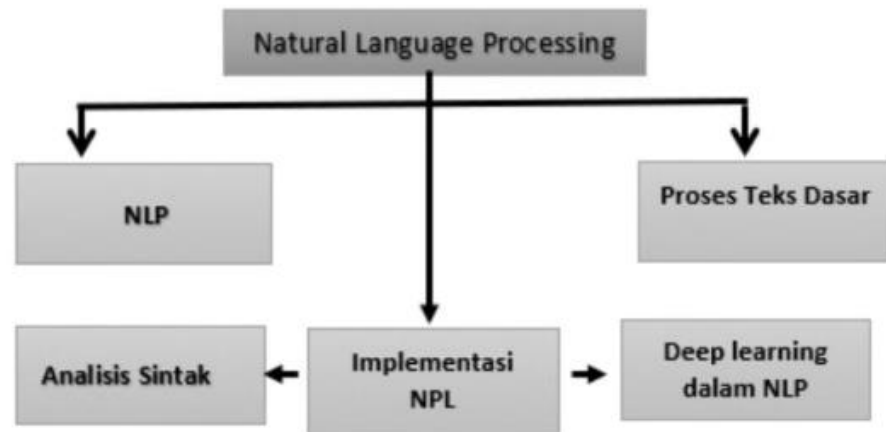
Metode pembelajaran yang menggunakan data tanpa label untuk menemukan pola, seperti *clustering* dan *dimensionality reduction*. Contohnya adalah *K-Means* dan PCA (*Principal Component Analysis*).

3. *Reinforcement Learning*

Metode pembelajaran yang berfokus melalui interaksi dengan lingkungan untuk memaksimalkan hadiah atau *reward*. Metode ini banyak diterapkan pada robotika dan industri game.

3. *Natural processing Language (NLP)*

Natural Language Processing (NLP) atau Pemrosesan Bahasa Alami merupakan bidang dalam kecerdasan buatan yang berfokus pada kemampuan komputer untuk memahami dan memproses bahasa manusia, dengan pengembangannya memadukan metode komputasi serta konsep linguistik (Wardhono et al., 2026:195). Proses NLP bekerja melalui dua tahapan utama, yakni prapemrosesan data dan pengembangan algoritma. Tahap prapemrosesan mencakup persiapan dan pembersihan data teks dengan tujuan mengubah data ke dalam format yang siap digunakan oleh algoritma, sehingga komputer dapat menganalisis data secara optimal. Tahap ini meliputi *part-of-speech tagging*, *stop word removal*, *lemmatization/stemming*, dan *tokenization*. Setelah itu, pengembangan algoritma dapat dilakukan melalui dua pendekatan utama: *rule-based system* yang menggunakan aturan linguistik, dan *machine learning-based system* yang memanfaatkan metode statistik serta data pelatihan untuk mempelajari pola (Ohyver et al., 2024:37).



Gambar 2. 3 Proses NLP bekerja

Sumber: Ohyver et al. (2024:37)

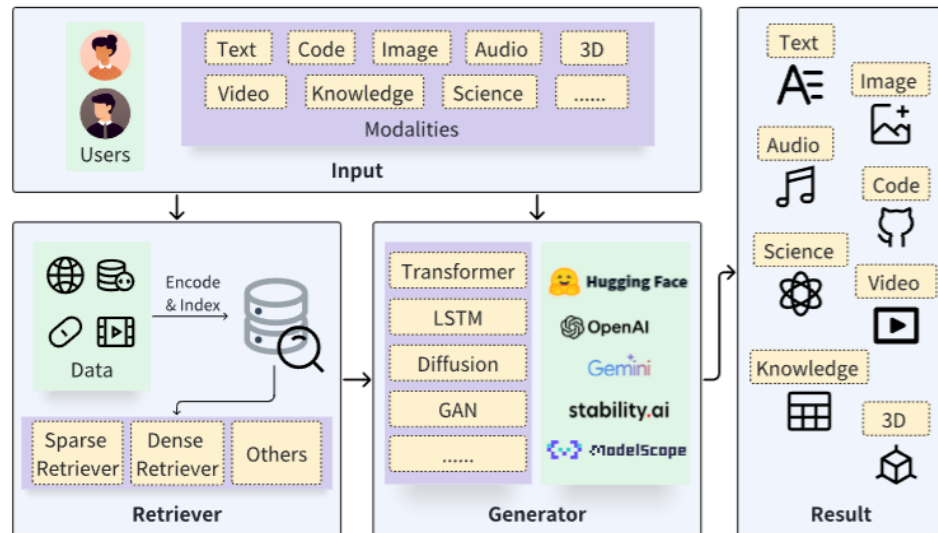
4. *Large Language Model (LLM)*

Large Language Model (LLM) merupakan kategori AI mutakhir yang dioptimalkan untuk menganalisis dan mengolah data bahasa manusia dalam volume yang masif (Wibowo, 2023:20). LLM memiliki kemampuan krusial dalam memproses dan memproduksi bahasa manusia dengan menganalisis dataset masif yang terdiri dari miliaran hingga triliunan kata (Asri et al., 2025:31). Proses pengembangan LLM mencakup dua fase krusial, yakni *pre-training* untuk menyerap pola bahasa dari data masif seperti literatur digital dan internet, serta *fine-tuning* melalui supervisi manusia guna meningkatkan akurasi dan relevansi jawaban (Gauthier et al., 2025:5).

5. *Retrieval-Augmented Generation (RAG)*

Retrieval-Augmented Generation (RAG) memberikan solusi dengan mengintegrasikan model bahasa *pre-trained* dengan modul pengambilan nonparametrik dengan menarik data eksternal selama proses inferensi (Sharma, 2025:1). Sebagaimana dijelaskan oleh Gupta et al. (2024)

model RAG telah berkembang dalam berbagai modalitas data meliputi *Text-Based*, *Audio-Based*, *Video-Based* dan *Multimodal*.



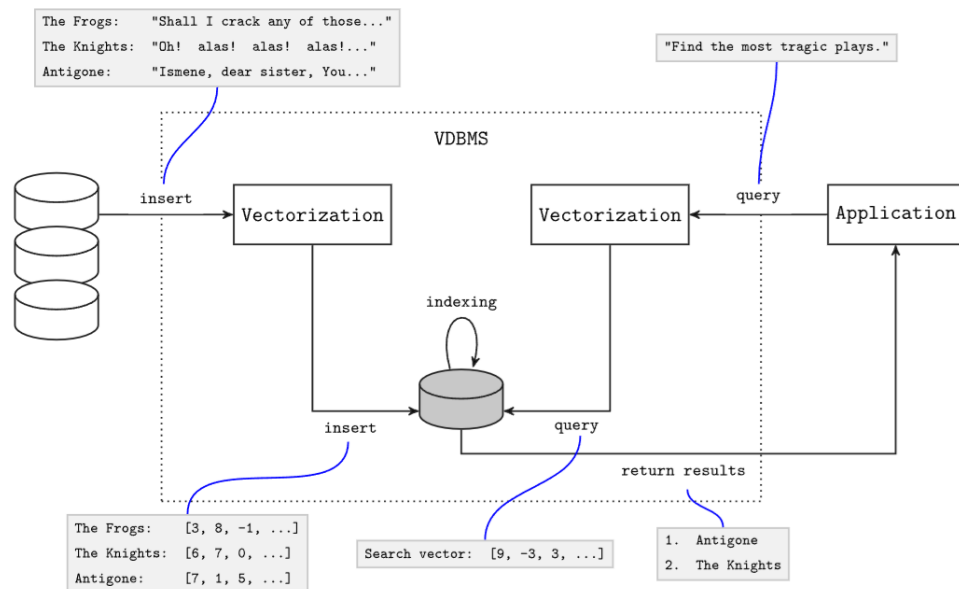
Gambar 2. 4 Arsitektur RAG
Sumber: P. Zhao et al. (2024)

Arsitektur umum RAG di atas bekerja dengan memproses kueri pengguna dari berbagai modalitas sebagai input bagi *retriever* yang mengekstraksi informasi relevan dari sumber data, serta bagi *generator* yang mengolah hasil pencarian tersebut untuk menghasilkan luaran akhir dalam berbagai bentuk modalitas yang sesuai (P. Zhao et al., 2024).

6. *Vector Database*

Vector Database terbagi menjadi prosesor kueri yang menangani spesifikasi, operator logis, implementasi fisik, dan optimasi kueri, serta manajer penyimpanan yang bertanggung jawab mengelola indeks pencarian dan penyimpanan fisik vektor (Pan et al., 2023). *Vector Database* memiliki dua fungsi inti, yaitu *vetor storage* yang dioptimalkan melalui kuantisasi, kompresi, dan sistem terdistribusi untuk efisiensi skalabilitas, serta *vector*

retrieval yang mengandalkan teknik pengindeksan khusus seperti metode *tree-based*, *hashing*, model berbasis graf, dan kuantisasi (Ma et al., 2025).



Gambar 2. 5 Komponen *Vector Database*
Sumber: Taipalus (2024)

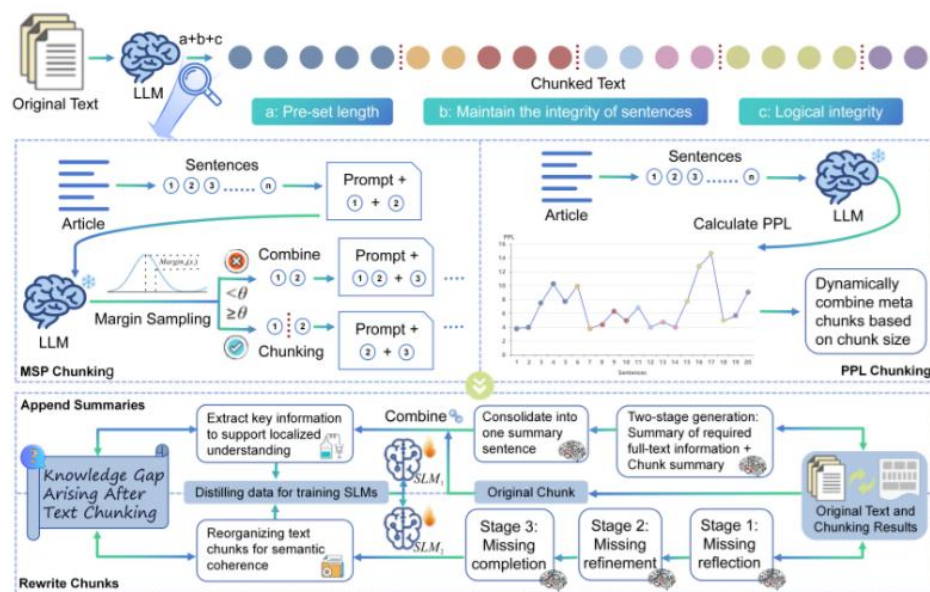
Komponen *Vector Database* di atas menunjukkan menunjukkan alur informasi ke dan dari basis data vektor. Proses vektorisasi mengubah data menjadi vektor sehingga dapat dibandingkan dengan cepat, sedangkan kueri bahasa alami memerlukan informasi tambahan di luar data utama. (Taipalus, 2024).

7. Text Chunking

Text Chunking banyak diadopsi karena sederhana dan hemat komputasi, sehingga efektif untuk memproses kumpulan dokumen dalam jumlah besar (Chen et al., 2026). Bernardo et al. (2025) menyebutkan pendekatan memanfaatkan *embedding* dan LLM (*Large Language Model*) untuk menentukan titik pemisahan secara lebih semantik. Strateginya

meliputi *Fixed Token Chunking*, *Recursive Chunking*, *Kamradt Semantic Chunking* dan *Cluster Semantic Chunking*, serta *Proposition Language Model Chunking* dan *Lumber Language Model Chunking*.

Penggunaan *text chunking* terlebih dahulu dipecah menjadi beberapa sub-urutan berdasarkan token. Proses *Meta-Chunking* membagi teks menjadi *meta-chunk* berdasarkan kalimat, kemudian menggabungkannya secara dinamis agar panjang *chunk* sesuai kebutuhan. Untuk menangani keterbatasan memori dan konteks LLM, digunakan mekanisme *KV (key value) caching* yang menghapus data sementara secara bertahap sehingga pemrosesan tetap efisien tanpa banyak mengurangi konteks. (J. Zhao et al., 2025).



Gambar 2. 6 Contoh penggunaan *text chunking*
Sumber: J. Zhao et al. (2025)

8. Cosine Similarity

Cosine similarity merupakan metode yang umum digunakan untuk mengukur tingkat kemiripan antara dua dokumen teks (Philip & Srivastava,

2023). *Cosine similarity* menilai tingkat kemiripannya berdasarkan kesesuaian arah antar vektor tersebut dan setiap dokumen yang akan dibandingkan direpresentasikan sebagai vektor dalam suatu ruang (Januzaj & Luma, 2022).

Dalam penelitian AlShammari (2023) terdapat dua vektor A dan B. Hasil kali titik (*dot product*) keduanya dinyatakan sebagai:

$$A \cdot B = |A| |B| \cos(\theta) \quad (1)$$

dengan $|A|$ dan $|B|$ adalah panjang (norma) masing-masing vektor, serta θ merupakan sudut di antara keduanya.

Nilai *cosinus* sudut tersebut dihitung menggunakan:

$$\cos(\theta) = \frac{A \cdot B}{|A||B|} \quad (2)$$

Secara umum, jika:

$$A = (a_1, a_2, \dots, a_n) \quad (3)$$

$$B = (b_1, b_2, \dots, b_n) \quad (4)$$

maka rumusnya menjadi:

$$\cos(\theta) = \frac{\sum(a_i b_i)}{\sqrt{\sum a_i^2} \sqrt{\sum b_i^2}} \quad (5)$$

Nilai *cosine similarity* menunjukkan persentase kemiripan antara dua vektor. Jika nilainya mendekati 1, maka kedua vektor sangat mirip. Sedangkan jika mendekati 0, maka keduanya semakin tidak mirip.

9. *Embedding*

Embedding merupakan representasi kata dalam bentuk vektor berdimensi tetap yang mampu merepresentasikan makna semantik umum serta pola linguistik dalam bahasa alami (Asudani et al., 2023). Kinerja model sangat dipengaruhi oleh pemilihan teknik *embedding* yang mencakup pendekatan tradisional, statis, atau kontekstual (Muchori & Maina, 2025).

10. *ChromaDB*

ChromaDB merupakan basis data vektor dalam sistem RAG yang berfungsi menampung bentuk vektor dari fragmen teks yang diperoleh dari tahap *embedding* (Albert & Voutama, 2025). ChromaDB mempermudah proses pencarian vektor berdasarkan kesamaan semantik (Qoniah & Ramadhani, 2025). ChromaDB dapat dengan mudah diintegrasikan dengan alat AI seperti LangChain (Jeong, 2023).

11. Google Gemini

Gemini merupakan model AI paling ambisius dari Google DeepMind yang dirancang dengan kemampuan memproses serta mengintegrasikan berbagai jenis informasi, mulai dari teks dan kode hingga audio, gambar, dan video (Pande et al., 2024). Meskipun memerlukan sumber daya komputasi tambahan, Gemini secara konsisten menunjukkan performa tangguh di berbagai jenis data, yang menegaskan kemampuannya dalam mengelola tugas-tugas kompleks dan terintegrasi (Rahman et al., 2025). Cara kerja Gemini dalam NLP dimulai dengan memproses input

pengguna dengan memecahnya ke dalam komponen-komponen utamanya (Alghozali & Mukminatun, 2024).

12. RAGAS

RAGAS (*Retrieval Augmented Generation Assessment*) merupakan evaluasi yang memungkinkan analisis performa *pipeline* RAG dilakukan secara mendetail dan terperinci (Antal & Buza, 2025). *Framework* RAGAS menyediakan integrasi dengan LlamaIndex dan LangChain kerangka kerja yang paling umum digunakan untuk membangun solusi RAG sehingga memudahkan pengembang untuk menerapkan RAGAS ke dalam alur kerja standar mereka (Es et al., 2025). Evaluasi kualitas sistem RAG dilakukan secara komprehensif melalui metrik kualitas jawaban mencakup *faithfulness*, *answer relevancy*, *semantic similarity*, dan *factual correctness* serta *context recall* (Husain et al., 2025).

13. Skala Likert

Menurut Abdullah et al. (2021:45), Skala Likert adalah skala yang digunakan untuk menilai sikap, opini, persepsi seseorang atau sekelompok orang tentang suatu isu tertentu di masyarakat. Setiap variabel penelitian dalam skala Likert dijabarkan ke dalam sejumlah indikator pengukuran yang terperinci.

Skala Likert memungkinkan responden menyatakan pendapat dengan memilih salah satu opsi yang tersedia, umumnya dalam rentang "sangat setuju" hingga "sangat tidak setuju". Setiap jawaban diberi nilai numerik (misalnya 1–5), lalu hasilnya diakumulasikan atau dirata-ratakan

guna memperoleh gambaran umum tentang persepsi responden terhadap topik penelitian.

Rumus untuk mendapatkan nilai dari skala likert ditunjukkan pada Persamaan.

$$\text{Nilai Rata - Rata} = \frac{\text{Jumlah Bobot Nilai responden}}{\text{Jumlah Responden}} \quad (1)$$

Setelah nilai rata-rata diperoleh, langkah selanjutnya adalah menghitung persentase menggunakan Persamaan.

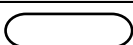
$$\text{Persentase} = \frac{\text{Nilai Rata - Rata}}{\text{Bobot Maksimum}} \quad (2)$$

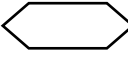
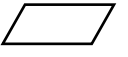
$$\text{Persentase} = \frac{\text{Jumlah Presentase Pernyataan}}{\text{Jumlah Pernyataan}} \quad (3)$$

14. *Flowchart*

Flowchart menggambarkan rangkaian proses visual dari algoritma komputasi menggunakan simbol-simbol khusus yang saling dikaitkan oleh panah penunjuk arah aliran instruksi (Putri et al., 2022:13). *Flowchart* berfungsi untuk menggambarkan prosedur operasional bisnis serta sirkulasi dokumen yang terjadi di dalam lingkungan organisasi (Purwanto & Alfarisi, 2025). *Program flowchart* menggambarkan kronologi dan logika berpikir dari sebuah instruksi atau prosedur dalam menuntaskan suatu persoalan (Samodra et al., 2024:41), notasi flowchart yang digunakan sebagai berikut.

Tabel 2. 1 Notasi *Flowchart*

No	Simbol	Nama	Fungsi
1.		Terminator	Mulai/akhiri program

No	Simbol	Nama	Fungsi
2.		Flow Direction	Sebagai koneksi antar simbol dan sebagai penunjuk garis alir suatu proses.
3.		<i>Preparation</i>	Memulai penyimpanan yang akan dipakai untuk wadah pengolahan di dalam storage.
4.		Proses	Proses pengolahan data oleh komputer.
5.		<i>Input/output data</i>	Proses <i>input/output</i> data tergantung jenis peralatannya
6.		<i>Predefined Process</i>	Proses menjalankan sub program yang belum detail dan dideskripsikan pada bagian lain.
7.		<i>Decision</i>	Perbandingan pilihan berdasarkan kondisi yang ada.
8.		<i>On page connector</i>	Penghubung bagian-bagian <i>flowchart</i> yang berada pada satu halaman
9.		<i>Off page connector</i>	Penghubung bagian-bagian <i>flowchart</i> yang berada pada halaman berbeda
10.		<i>Manual Input</i>	Input data secara <i>on-line keyboard</i> .
11.		<i>Document</i>	Input dokumen dalam bentuk kertas atau output dicetak ke kertas

Sumber: Samodra et al. (2024:41)

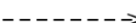
15. UML

UML (*Unified Modelling Language*) merupakan representasi perancangan objek untuk merencanakan sistem yang diterapkan, ataupun yang telah berjalan saat ini (Nisa' et al., 2024:1). Arsitektur UML mencakup komponen seperti *use case*, *activity*, *sequence* serta *class diagram*. Dalam penerapannya, *activity diagram* berfungsi memetakan rangkaian tindakan pengguna maupun admin, sementara *sequence diagram* berfokus memperlihatkan kronologi interaksi yang terjadi antarkomponen di dalam sistem (Hidayatullah et al., 2026). Seluruh struktur maupun fungsi komponen sistem dan dapat dilihat pada penelitian Mallu et al. (2023:94) dengan simbol-simbol di bawah ini.

1. Use Case Diagram

Arsitektur sistem secara visual dan terstruktur dari *Use Case Diagram* dapat dijabarkan melalui beberapa diagram berikut:

Tabel 2. 2 Simbol *Use Case Diagram*

No	Simbol	Nama	Deskripsi
1		<i>Actor</i>	Individu yang berinteraksi dengan sistem yang ditandai dengan peletakan kata benda pada awal penamaan aktor tersebut.
2		<i>Use Case</i>	Setiap fungsi sistem yang berkomunikasi dengan <i>actor</i> , di mana standarisasi penamaannya menggunakan format kata kerja yang mendahului kata benda atau frasanya.
3		<i>Association</i>	Koneksi antarkomponen sistem, baik berupa pertukaran informasi antara <i>actor</i> dengan <i>use case</i> , maupun bentuk korelasi timbal balik di antara beberapa <i>use case</i> yang berbeda.
4		<i>Include</i>	Hubungan ketergantungan antar- <i>use case</i> yang menunjukkan bahwa satu fungsi tidak dapat berjalan tanpa mengeksekusi fungsi lainnya.
5		<i>Extend</i>	Hubungan beberapa <i>use case</i> di mana satu fungsi berperan sebagai fitur tambahan opsional bagi pengguna.
6		<i>Generalization</i>	Koneksi terstruktur yang menghubungkan antara <i>use case</i> yang bersifat umum (<i>general</i>) dengan <i>use case</i> turunan yang lebih spesifik (<i>specific</i>).
7		<i>Dependency</i>	Hubungan ini mencerminkan ketergantungan antar-elemen <i>use case</i> , di mana simbol garis <i>dependency</i> mengarah dari komponen yang membutuhkan

No	Simbol	Nama	Deskripsi
			bantuan (<i>dependent</i>) ke komponen utama yang menyediakan fungsi tersebut (<i>independent</i>).

Sumber: Mallu et al. (2023:96)

2. Activity Diagram

Rancangan *Activity Diagram* dimodelkan menggunakan notasi sebagai berikut:

Tabel 2.3 Notassi *Activity Diagram*

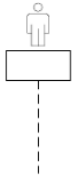
No	Simbol	Nama	Deskripsi
1		<i>Initial Node</i>	Komponen ini menandai dimulainya alur aktivitas pada diagram dan disimbolkan dengan bentuk lingkaran yang memiliki tanda panah keluar.
2		<i>Action</i>	Komponen ini menggambarkan tindakan nyata dalam alur kerja dan divisualisasikan melalui simbol persegi panjang yang memuat label keterangan aktivitasnya.
3		<i>Flow Line</i>	Elemen yang merepresentasikan jalur perpindahan kontrol atau data, yang berfungsi mengintegrasikan satu tahapan aktivitas dengan aktivitas berikutnya dalam diagram.
4		<i>Decision</i>	<i>Decision</i> merepresentasikan pos pemilihan rute berdasarkan kondisi tertentu di dalam alur aktivitas.
5		<i>Fork Node & Join Node</i>	<i>Fork node</i> merepresentasikan pembagian alur menjadi beberapa tugas paralel yang dikerjakan bersamaan, sedangkan <i>join node</i> bertindak sebagai penggabung akhir yang melebur jalur-jalur paralel tersebut kembali ke satu alur utama.
6		<i>Final Node</i>	Digambarkan dengan notasi lingkaran dengan tanda panah keluar dan berfungsi menjadi titik akhir aktivitas.

Sumber: Mallu et al. (2023:99)

3. Sequence Diagram

Spesifikasi dan visualisasi *Sequence Diagram* yang dimodelkan menggunakan notasi di bawah ini.

Tabel 2.4 Simbol *Sequence Diagram*

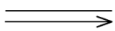
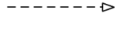
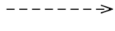
No	Simbol	Nama	Deskripsi
1		<i>Actor Lifeline</i>	Komponen ini melambangkan pengguna yang berkomunikasi dengan sistem, di mana fokusnya hanya pada peran luar mereka tanpa memperlihatkan rincian bagian dalam sistem.
2		<i>Object Lifeline</i>	Bagian ini menunjukkan masa hidup (<i>lifeline</i>) dan keterlibatan aktif suatu objek dalam model sistem yang sedang dikembangkan.
3		<i>Activation</i>	Bagian ini menggambarkan masa keaktifan atau fase eksekusi operasi dari suatu objek selama interaksi berlangsung.
4		<i>Message</i>	Jenis pesan antraobjek yang menuntut adanya respons balik dari penerima, sehingga alur eksekusi pada objek pengirim akan tertahan sementara waktu hingga balasan tersebut diterima.
5		<i>Asynchronous Message</i>	Jenis pesan yang dikirim ke objek tujuan tanpa memerlukan respons langsung, sehingga objek pengirim tidak mengalami penundaan (<i>blocking</i>) dalam melanjutkan aktivitasnya.
6		<i>Return Message</i>	Pesan konfirmasi berupa nilai balik (<i>return value</i>) yang diserahkan oleh objek penerima kepada objek pengirim setelah stimulus atau instruksi awal sukses dieksekusi.
7		<i>Self Message</i>	Pemesanan metode (<i>method</i>) yang digunakan terhadap objek itu sendiri.

Sumber: Mallu et al. (2023:101)

4. Class Diagram

Rancangan *Class Diagram* dimodelkan menggunakan notasi sebagai berikut:

Tabel 2.5 Notasi *Class Diagram*

No	Simbol	Nama	Deskripsi
1		<i>Class</i>	Struktur sebuah kelas secara umum terbagi menjadi tiga bagian penting: nama kelas, atribut (karakteristik data), dan metode/fungsi (<i>method</i>).
2		<i>Association</i>	Relasi penghubung antar-dua kelas, di mana hubungan asosiasi ini dapat bersifat berarah (<i>directed</i>) ataupun tidak berarah, menyesuaikan dengan kebutuhan interaksi dari kelas-kelas terkait.
3		<i>Aggregation</i>	Jenis relasi ini mengaitkan kelas kontainer (utama) dengan kelas bagian, di mana hubungan yang terjalin bersifat longgar sehingga kedua kelas tersebut tetap dapat berdiri masing-masing tanpa bergantung satu sama lain.
4		<i>Composition</i>	Relasi antara kelas dan bagian dari kelas tersebut, akan tetapi kelas bagian cukup diakui dari satu kelas utama.
5		<i>Inheritance</i>	Relasi penurunan yang terdiri dari dua kelas dan diwakili dari garis panah yang menuju pada kelas anak ke kelas utama.
6		<i>Realization</i>	Relasi yang terdiri dari kelas abstrak dengan kelas konkret yang menerapkan penurunan kelas abstrak.
7		<i>Dependency</i>	Relasi yang terdiri dari dua kelas dimana satu kelas memerlukan kelas yang lain untuk mengimplementasikan operasi terkhusus.

Sumber: Mallu et al. (2023:103)

B. Kajian Empiris

Kajian empiris dalam penelitian ini didasarkan pada penelusuran terhadap penelitian-penelitian terdahulu yang relevan dengan topik pengembangan sistem chatbot akademik berbasis *Large Language Model* (LLM) dan metode *Retrieval-Augmented Generation* (RAG). Penelitian pertama dilakukan oleh Talaohu et al. (2025) bertujuan membangun sistem chatbot berbasis website untuk melayani kebutuhan informasi Penerimaan Mahasiswa Baru (PMB) di Universitas Muhammadiyah Sorong. Metode yang digunakan adalah *Retrieval-Augmented Generation* (RAG) dengan pendekatan *Extreme Programming*. Sistem berhasil mencapai akurasi 93%, precision 90%, recall 95%, dan F1-score 92% berdasarkan evaluasi BERTScore, serta memperoleh skor *Usability Testing* sebesar 88,5%. Keterbatasan penelitian ini adalah cakupan yang hanya menyasar calon mahasiswa baru, serta tidak menjelaskan *framework* dan *vector database* yang digunakan secara spesifik.

Penelitian kedua dilakukan oleh Virgiansyah & Pribadi (2025) membangun chatbot helpdesk berbasis WhatsApp menggunakan Gemini 2.5 Flash, RAG, dan Qdrant dengan metode *waterfall*. Evaluasi penelitian ini menggunakan RAGAS menunjukkan peningkatan *Answer Correctness* dari 0,22 menjadi 0,89 setelah penerapan RAG, dengan *Faithfulness* mencapai 0,98. Keterbatasan penelitian ini adalah penggunaan Qdrant yang bersifat *cloud-based* serta tidak menggunakan LangChain sebagai *framework*.

Penelitian ketiga dilakukan oleh Muhajir et al. (2025) dengan mengimplementasikan chatbot panduan akademik Universitas Trunojoyo menggunakan LangChain berbasis GPT dan Pinecone sebagai *vector database*. Hasil penelitian ini menunjukkan akurasi 96,66% dengan kombinasi *query preprocessing*, serta UAT sebesar 82,79%. Keterbatasan penelitian ini terletak pada penggunaan GPT dan Pinecone yang keduanya bersifat *cloud-based* dan berbayar, serta belum mengkaji efisiensi *retrieval* pada skala program studi.

Penelitian keempat oleh Neupane et al. (2024) mengembangkan chatbot berbasis RAG dengan memanfaatkan OpenAI sebagai LLM dan ChromaDB sebagai *vector database* untuk menyediakan informasi sumber daya universitas. Sistem ini mampu menghasilkan jawaban yang kontekstual dengan memanfaatkan dokumen sebagai *knowledge base*. Namun demikian, penelitian ini masih menghadapi beberapa kendala seperti adanya *hallucination*, keterbatasan *context window*, serta belum optimalnya sistem *retrieval*.

Penelitian kelima oleh Kizi & Suh (2025) mengevaluasi performa *pipeline* RAG dalam layanan chatbot untuk penerimaan mahasiswa internasional dengan menggunakan berbagai model LLM, termasuk GPT-4o dan model *open-source* seperti LLaMA3 dan OpenChat. Sistem dibangun menggunakan LangChain dan FAISS sebagai *vector database*. Hasil penelitian menunjukkan bahwa model *open-source* yang dioptimalkan dengan teknik seperti *multi-query retrieval* dan *hybrid retriever* mampu menyamai bahkan melampaui performa model komersial dalam metrik seperti *answer relevancy*,

faithfulness, dan *context precision*. Meskipun demikian, penelitian ini lebih berfokus pada optimasi teknis *pipeline* RAG dan belum mengkaji implementasi pada domain akademik lokal yang spesifik seperti program studi.

Berdasarkan keseluruhan penelitian tersebut, dapat disimpulkan bahwa penggunaan RAG dalam chatbot akademik terbukti mampu meningkatkan akurasi dan relevansi jawaban, terutama ketika dikombinasikan dengan berbagai LLM dan teknik optimasi *retrieval*. Namun, masih terdapat beberapa *research gap* yang belum terjawab, antara lain kurangnya fokus pada pengembangan chatbot di tingkat program studi, terbatasnya eksplorasi penggunaan ChromaDB dibandingkan *vector database* lain, serta masih minimnya penelitian yang mengintegrasikan LLM Gemini secara optimal dalam arsitektur RAG. Selain itu, permasalahan terkait pengelolaan konteks, *hallucination*, dan optimalisasi *retrieval* juga masih menjadi tantangan. Oleh karena itu, penelitian ini menghadirkan *novelty* berupa pengembangan chatbot akademik berbasis RAG yang secara khusus diterapkan pada Program Studi Teknik Informatika UNIPMA dengan mengintegrasikan LLM Gemini dan ChromaDB sebagai *vector database*, serta memanfaatkan *knowledge base* lokal agar menghasilkan jawaban yang lebih kontekstual, akurat, dan sesuai dengan kebutuhan pengguna.

C. Kerangka Berpikir

Kerangka berpikir penelitian ini dibangun atas dasar permasalahan nyata yang dihadapi mahasiswa aktif Program Studi Teknik Informatika Universitas PGRI Madiun, yaitu belum tersedianya sistem layanan informasi

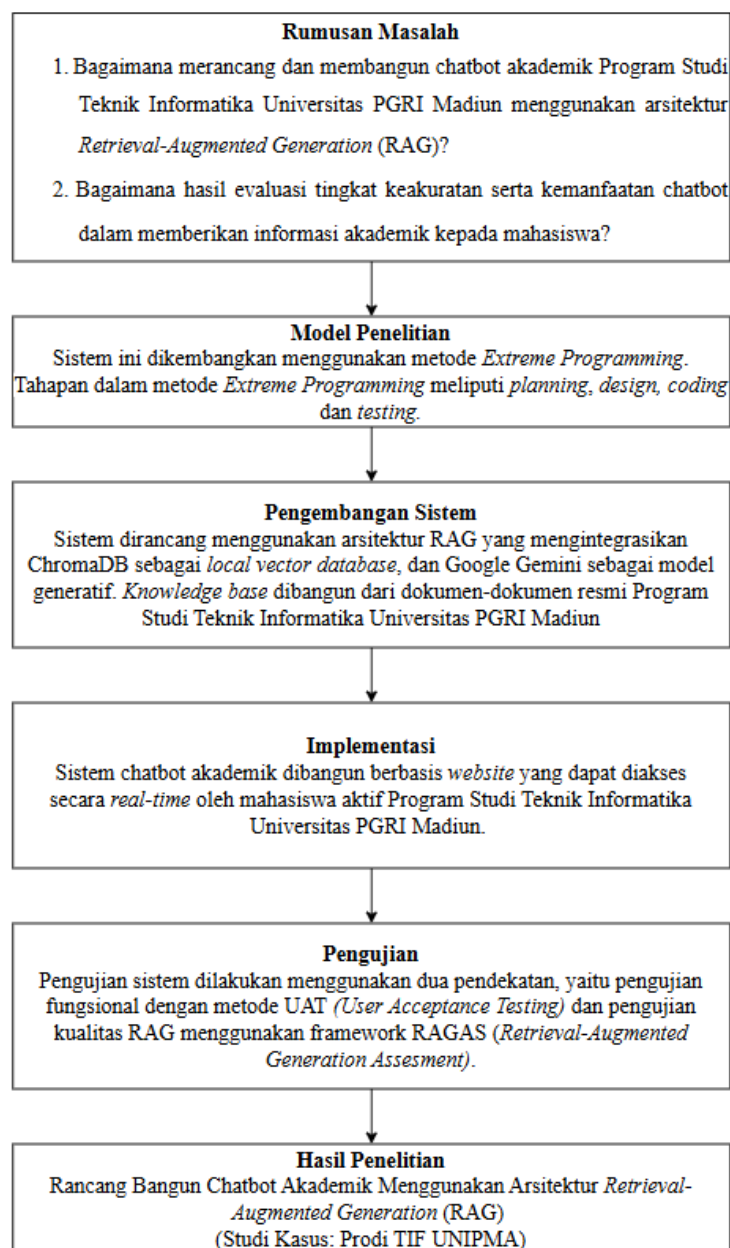
akademik yang dapat diakses secara cepat, akurat, dan *real-time* di luar jam kerja staf. Kondisi ini mendorong perlunya pengembangan sebuah sistem cerdas yang mampu menjawab kebutuhan informasi mahasiswa secara otomatis dan berbasis dokumen resmi institusi.

Untuk menjawab permasalahan tersebut, penelitian ini mengusulkan rancang bangun sistem chatbot akademik dengan memanfaatkan pendekatan *Retrieval-Augmented Generation* (RAG) yang mengintegrasikan ChromaDB sebagai *local vector database*, dan Google Gemini sebagai model generatif. Sistem dikembangkan menggunakan metode *Extreme Programming* (XP) agar proses penyempurnaan *pipeline* RAG dapat dilakukan secara iteratif hingga menghasilkan sistem yang benar-benar sesuai kebutuhan pengguna. Tahapan dalam metode *Extreme Programming* (XP) meliputi *planning*, *design*, *coding* dan *testing*.

Knowledge base dibangun dari dokumen-dokumen resmi program studi dan diimplementasikan dalam bentuk aplikasi berbasis website yang melibatkan dua aktor utama, yaitu mahasiswa sebagai pengguna yang mengajukan pertanyaan akademik, dan administrator sebagai pengelola *knowledge base* dan dokumen resmi program studi.

Kelayakan sistem kemudian diukur melalui dua pendekatan pengujian, yaitu *usability testing* untuk mengukur kemudahan dan kepuasan pengguna dalam berinteraksi dengan sistem, serta RAGAS untuk mengevaluasi kualitas *retrieval* dan generasi jawaban secara kuantitatif. Pengujian RAGAS yang mencakup metrik *faithfulness*, *answer relevancy*, *semantic similarity*, dan

factual correctness. Dengan alur berpikir yang sistematis tersebut, penelitian ini diharapkan menghasilkan sistem chatbot akademik yang akurat, efisien secara infrastruktur, dan mampu memberikan kontribusi nyata terhadap peningkatan kualitas layanan informasi akademik di Program Studi Teknik Informatika Universitas PGRI Madiun.



Gambar 2. 7 Kerangka Berpikir