

***CHATBOT IT SUPPORT ASSISTANT FOR BADAN SISTEM
INFORMASI UNIVERSITAS ISLAM INDONESIA***



Disusun Oleh:

N a m a : Moch Raffry Syalwa Rizqullah

NIM : 20523210

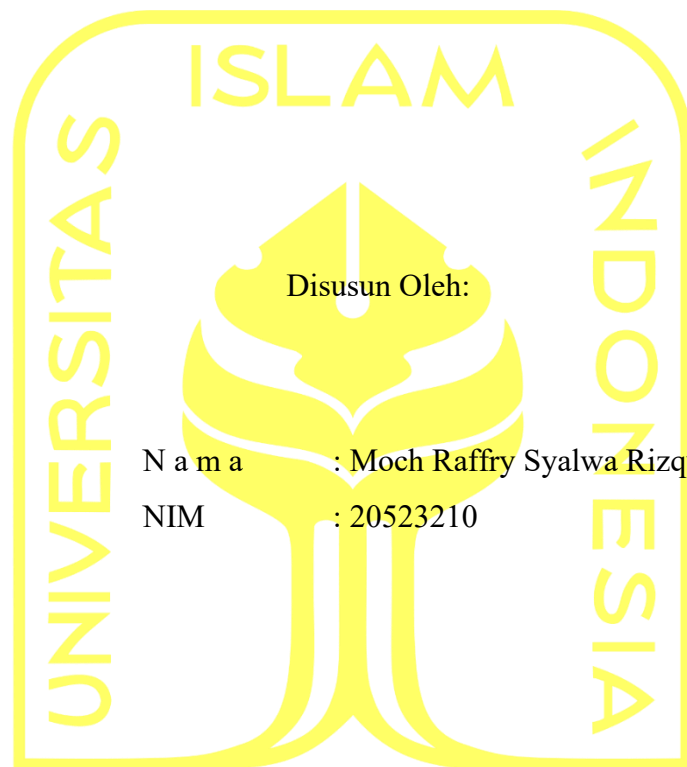
**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**CHATBOT IT SUPPORT ASSISTANT FOR BADAN SISTEM
INFORMASI UNIVERSITAS ISLAM INDONESIA**

TUGAS AKHIR



N a m a : Moch Raffry Syalwa Rizqullah
NIM : 20523210

Yogyakarta, 25 Desember 2025

Pembimbing,

(Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

**CHATBOT IT SUPPORT ASSISTANT FOR BADAN SISTEM
INFORMASI UNIVERSITAS ISLAM INDONESIA****TUGAS AKHIR**

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 15 Januari 2026

Tim Penguji

Mukhammad Andri Setiawan, S.T., M.Sc.,
Ph.D.

Anggota 1

Dr. Raden Teduh Dirgahayu, S.T., M.Sc.

Anggota 2

Sheila Nurul Huda, S.Kom., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia



(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Moch Raffry Syalwa Rizqullah

NIM : 20523210

Tugas akhir dengan judul:

**CHATBOT IT SUPPORT ASSISTANT FOR BADAN SISTEM
INFORMASI UNIVERSITAS ISLAM INDONESIA**

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 23 Desember 2025



(Moch Raffry Syalwa Rizqullah)

HALAMAN PERSEMBAHAN

Alhamdulillah Rabbil ‘Alamin. Segala puji hanya bagi Allah SWT atas limpahan rahmat dan hidayahnya, sehingga penulis dapat menjalani kehidupan serta menyelesaikan kewajiban dengan tetap berpegang pada ajaran Islam. Shalawat dan salam semoga senantiasa tercurah kepada Nabi Muhammad S.A.W, yang telah membawa umat manusia dari masa kegelapan menuju terang benderang yang penuh tuntunan.

Penulis mengucapkan rasa syukur kepada Allah Subhanahu wa Ta’ala atas limpahan nikmat, rahmat, dan ridhonya yang senantiasa mengalir. Shalawat serta salam semoga selalu tercurah kepada Nabi Muhammad SAW sebagai suri teladan bagi seluruh umat.

Penulis menyampaikan terima kasih yang tulus kepada kedua orang tua atas doa, dukungan, dan kasih sayang yang tak henti menguatkan. Apresiasi juga disampaikan kepada saudara-saudari yang senantiasa memberi semangat dan menemani setiap proses.

Penulis mengucapkan terima kasih kepada teman-teman yang telah memberikan dukungan dan semangat selama penyusunan skripsi ini. Penulis mohon maaf karena tidak dapat menyebutkan nama-nama tersebut satu per satu.

Melalui karya ini, penulis berupaya menyampaikan rasa terima kasih dan dedikasi kepada orang tua, sekaligus menunaikan amanah dalam menyelesaikan jenjang pendidikan Sarjana.

HALAMAN MOTO

Sometimes it is the people no one imagines anything of who do the things no one can imagine. When your time comes, no force can make you lose — for time and seasons belong to God alone.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala atas segala rahmat, hidayah, dan karunia-Nya sehingga tugas akhir yang berjudul "*Chatbot IT Support Assistant for Badan Sistem Informasi Universitas Islam Indonesia*" dapat diselesaikan dengan baik. Shalawat serta salam senantiasa tercurah kepada Nabi Muhammad Shallallahu 'alaihi wa sallam yang telah membawa umat manusia dari zaman kegelapan menuju zaman yang penuh dengan ilmu pengetahuan.

Penyusunan tugas akhir ini merupakan salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Informatika Universitas Islam Indonesia. Penulis menyadari bahwa penyelesaian tugas akhir ini tidak lepas dari bantuan, bimbingan, dan dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Mukhammad Andri Setiawan, S.T., M.Sc., Ph.D. selaku dosen pembimbing yang telah memberikan bimbingan, arahan, dan masukan yang sangat berharga dalam penyelesaian tugas akhir ini.
2. Dr. Ir. Raden Teduh Dirgahayu, S.T., M.Sc. selaku dosen pembimbing akademik yang telah memberikan arahan dan solusi atas berbagai kendala selama masa perkuliahan.
3. Kedua orang tua tercinta, yang senantiasa memberikan dukungan moral, material, doa, dan kasih sayang yang tiada henti.
4. Tim *IT Support* Badan Sistem Informasi UII yang telah bersedia menjadi narasumber dan memberikan validasi terhadap sistem yang dikembangkan.
5. Semua pihak yang telah membantu dalam penyelesaian tugas akhir ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan di masa mendatang. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu pengetahuan, khususnya dalam bidang teknologi informasi dan kecerdasan buatan.

Yogyakarta, 23 Desember 2025



(Moch Raffry Syalwa Rizqullah)

SARI

Peningkatan jumlah mahasiswa di Universitas Islam Indonesia mengakibatkan lonjakan permintaan *IT Support* pada Badan Sistem Informasi yang mencapai ribuan pertanyaan per bulan, menciptakan beban kerja berlebih bagi tim *IT Support* yang memiliki keterbatasan sumber daya manusia dan waktu operasional terbatas pada jam kerja. Penelitian ini mengembangkan sistem *Chatbot IT Support* BSI UII berbasis teknologi OpenAI dan *Retrieval-Augmented Generation* untuk mengurangi beban kerja tim *IT Support*, mempercepat waktu respon, dan menyediakan layanan dukungan 24 jam.

Metode penelitian prototipe diterapkan dalam tiga iterasi pengembangan dengan implementasi arsitektur berlapis yang mengintegrasikan *LangGraph* sebagai orkestrator alur kerja untuk klasifikasi maksud dan *routing* cerdas, OpenAI GPT-4o untuk pembangkitan respon kontekstual dengan *prompt engineering* ketat anti halusinasi, *Pinecone Vector Database* untuk pencarian semantik *hybrid* dengan metadata *filtering*, serta *FastAPI*, *PostgreSQL*, *Redis*, dan *Cloudflare R2* sebagai komponen pendukung. Sistem mengimplementasikan *Role-Based Access Control* dengan tiga tingkatan peran *Admin*, Dosen, Mahasiswa untuk membatasi akses dokumen berdasarkan tingkatan sensitivitas, serta manajemen riwayat percakapan dengan *Time-to-Live* untuk kepatuhan privasi.

Hasil pengujian fungsional menunjukkan bahwa seluruh 25 *endpoint API* lulus pengujian dengan implementasi fitur kunci seperti *hybrid search*, *RBAC*, dan alur kerja *LangGraph* berfungsi sesuai spesifikasi. Pengujian non-fungsional menunjukkan waktu respon rata-rata sistem 2,06 detik memenuhi persyaratan di bawah 10 detik, sementara pengujian usabilitas menghasilkan tingkat akurasi respon 4,56 dari skala 5 dengan 100% responden menyatakan *chatbot* mempercepat waktu pencarian informasi.

Konfirmasi kualitatif dari tim *IT Support* BSI UII pada tanggal 17 Oktober 2025 memvalidasi bahwa sistem berhasil mengurangi beban kerja operasional terutama untuk pertanyaan repetitif yang mencakup 60-70% dari total permintaan bantuan seperti panduan *reset password*, prosedur akses *Wifi* kampus, dan informasi layanan *IT* dasar. Sistem terbukti mampu beroperasi 24 jam tanpa batasan waktu operasional, mengeliminasi kendala jam kerja yang sebelumnya menjadi hambatan utama dalam layanan *IT Support* konvensional, serta memungkinkan pengguna memperoleh jawaban instan tanpa harus menunggu jam kerja atau mengajukan tiket bantuan melalui email.

Kata kunci: *Chatbot, IT Support, OpenAI, LangGraph, Retrieval-Augmented Generation, Role-Based Access Control, Pinecone*

GLOSARIUM

<i>API</i>	<i>Application Programming Interface</i> atau Antarmuka pemrograman aplikasi yang memungkinkan komunikasi antar perangkat lunak melalui sekumpulan aturan dan protokol yang telah ditentukan.
<i>Asynchronous</i>	langkah untuk menelusuri kesalahan kode program.
<i>BM25</i>	metode pengembangan perangkat lunak.
<i>Chatbot</i>	Perangkat lunak yang dirancang untuk meniru percakapan manusia secara otomatis menggunakan teknologi kecerdasan buatan, baik melalui teks maupun suara.
<i>Cloudflare R2</i>	Layanan <i>object storage</i> yang kompatibel dengan Amazon S3 untuk menyimpan dan mengakses file berukuran besar dengan biaya rendah tanpa biaya <i>egress</i> .
<i>Dense Vector</i>	Representasi numerik berkepadatan tinggi yang merepresentasikan makna semantik teks dalam ruang vektor berdimensi tinggi untuk pencarian berbasis kemiripan makna.
<i>Embedding</i>	Proses mengkonversi data teks menjadi representasi vektor numerik yang dapat diproses oleh algoritma pembelajaran mesin untuk menangkap hubungan semantik antar kata.
<i>Endpoint</i>	Titik akhir URL yang menerima permintaan HTTP untuk mengakses sumber daya atau layanan tertentu dalam <i>Application Programming Interface</i> .
<i>FastAPI</i>	<i>Framework web modern</i> untuk membangun <i>Application Programming Interface</i> dengan Python yang mendukung pemrograman asinkron dan dokumentasi otomatis menggunakan standar OpenAPI.
GPT	<i>Generative Pre-trained Transformer</i> adalah Model bahasa berbasis <i>transformer</i> yang dilatih pada data teks masif untuk menghasilkan teks natural dan memahami konteks kompleks.
Halusinasi	Kondisi di mana model bahasa besar menghasilkan informasi yang tidak akurat atau tidak berdasar pada data yang diberikan, seolah-olah "mengada-ada" fakta.

<i>Hybrid Search</i>	Metode pencarian yang menggabungkan pencarian semantik berbasis vektor dengan pencarian kata kunci tradisional untuk meningkatkan akurasi hasil pencarian.
<i>JSON Web Token</i>	Standar terbuka untuk membuat token akses yang aman dan ringkas, digunakan untuk autentikasi dan pertukaran informasi antar pihak secara terverifikasi.
<i>LangChain</i>	<i>Framework open-source</i> untuk mengembangkan aplikasi berbasis model bahasa besar yang menyediakan komponen modular untuk integrasi dengan berbagai sumber data dan layanan.
<i>LangGraph</i>	<i>Extension</i> dari <i>LangChain</i> untuk membuat aplikasi stateful multi-actor dengan model bahasa besar menggunakan pendekatan <i>graph-based workflow</i> untuk orkestrasi alur kerja kompleks.
<i>Latency</i>	Waktu tunda antara permintaan dikirim dan respons diterima, diukur dalam milidetik, yang menentukan kecepatan responsivitas sistem.
<i>Metadata</i>	Data deskriptif yang memberikan informasi tentang data lain, seperti penulis dokumen, tanggal pembuatan, atau tingkat sensitivitas akses.
<i>Metadata Filtering</i>	Teknik penyaringan hasil pencarian berdasarkan atribut metadata untuk membatasi akses berdasarkan kriteria tertentu seperti peran pengguna atau tingkat sensitivitas.
NLP	Cabang kecerdasan buatan yang memungkinkan komputer memahami, menafsirkan, dan menghasilkan bahasa manusia secara natural.
OpenAI	Perusahaan penelitian kecerdasan buatan yang mengembangkan model bahasa besar seperti GPT dan menyediakan layanan <i>Application Programming Interface</i> untuk mengakses model tersebut.
<i>Pinecone</i>	<i>Platform vector database</i> yang dioptimalkan untuk menyimpan dan melakukan pencarian kemiripan pada <i>embedding</i> vektor dengan kecepatan tinggi dan skalabilitas.
<i>PostgreSQL</i>	Sistem manajemen basis data relasional <i>open-source</i> yang mendukung transaksi ACID dan fitur lanjutan seperti JSON, <i>full-text search</i> , dan <i>concurrency control</i> .
<i>Prompt</i>	Teknik merancang instruksi atau pertanyaan yang efektif untuk model bahasa besar guna menghasilkan <i>output</i> yang akurat, relevan, dan sesuai kebutuhan.

<i>RAG</i>	<i>Retrieval-Augmented Generation</i> adalah Pendekatan yang menggabungkan pencarian informasi dari basis pengetahuan eksternal dengan kemampuan generatif model bahasa untuk menghasilkan respons yang akurat dan faktual.
<i>RBAC</i>	<i>Role-Based Access Control</i> yaitu Mekanisme kontrol akses yang membatasi akses sistem berdasarkan peran pengguna, memastikan setiap peran hanya dapat mengakses sumber daya yang sesuai dengan kewenangannya.
<i>Redis</i>	Sistem penyimpanan data <i>in-memory</i> berbasis struktur data <i>key-value</i> yang mendukung berbagai tipe data dan digunakan sebagai <i>cache</i> , <i>message broker</i> , atau penyimpanan sesi.
<i>Reranking</i>	Proses pengurutan ulang hasil pencarian menggunakan model yang lebih kompleks untuk meningkatkan relevansi dokumen yang dikembalikan kepada pengguna.
<i>Retrieval</i>	Proses pencarian dan pengambilan informasi relevan dari basis pengetahuan berdasarkan <i>query</i> pengguna untuk digunakan dalam pembangkitan respons.
<i>Routing</i>	Proses mengarahkan permintaan atau <i>query</i> ke jalur pemrosesan yang tepat berdasarkan klasifikasi atau kondisi tertentu dalam alur kerja sistem.
<i>Semantic Search</i>	Metode pencarian yang memahami maksud dan konteks <i>query</i> untuk menemukan hasil berdasarkan makna semantik, bukan hanya kecocokan kata kunci literal.
<i>Sparse Vector</i>	Representasi vektor yang sebagian besar elemennya bernilai nol, digunakan dalam pencarian berbasis kata kunci untuk efisiensi komputasi dan penyimpanan.
<i>System Prompt</i>	Instruksi awal yang diberikan kepada model bahasa besar untuk mendefinisikan peran, batasan, dan perilaku yang diharapkan dalam menghasilkan respons.
<i>Time-to-Live</i>	Durasi waktu tertentu di mana data disimpan dalam <i>cache</i> atau penyimpanan sementara sebelum dihapus secara otomatis untuk kepatuhan privasi dan efisiensi memori.

<i>Vector Database</i>	Basis data yang dirancang khusus untuk menyimpan dan melakukan pencarian efisien pada data berbentuk vektor, mendukung operasi kemiripan dan jarak antar vektor.
<i>Vectorization</i>	Proses mengubah data teks atau media lainnya menjadi representasi vektor numerik yang dapat diproses oleh algoritma pembelajaran mesin.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING	ii
HALAMAN PENGESAHAN DOSEN PENGUJI	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR	iv
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	viii
GLOSARIUM	ix
DAFTAR ISI	xiii
DAFTAR TABEL	xv
DAFTAR GAMBAR	xvi
BAB I LATAR BELAKANG	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	3
1.5 Metodologi Penelitian	3
1.6 Sistematika Penulisan	4
BAB II LANDASAN TEORI DAN KAJIAN PUSTAKA	6
2.1 <i>Chatbot</i>	6
2.2 Natural Language Processing (NLP)	8
2.3 Kajian Pustaka	9
2.4 <i>Vector Embeddings</i>	12
2.5 <i>Vector Database</i>	13
2.6 Model <i>Generative Pre-trained Transformer</i>	15
2.6.1 Pelatihan Dan Adaptasi	16
2.6.2 Pengaplikasian dalam <i>Chatbot</i>	17
2.7 OpenAI	18
2.8 LangChain	20
2.9 AI Prompting	22
2.10 <i>Prompting Engineering</i>	26
BAB III METODELOGI PENELITIAN	29
3.1 Metodologi Penelitian Prototipe	29
3.2 Analisis Kebutuhan Sistem	29
3.2.1 Kebutuhan Fungsional	29
3.2.2 Kebutuhan Non-Fungsional	30
3.2.3 Analisis Kebutuhan <i>Input</i>	31
3.2.4 Analisis Kebutuhan <i>Output</i>	32
3.2.5 Analisis Kebutuhan Tampilan Antarmuka	33
3.3 Perancangan Desain Sistem	34
3.3.1 Gambaran Umum Sistem	35
3.3.2 <i>Use Case Diagram</i>	36
3.3.3 <i>Sequence Diagram</i>	40
3.3.4 Perancangan <i>Entity Relationship Diagram</i>	46
3.3.5 Perancangan Desain Antarmuka Pengguna	47
3.4 Implementasi Sistem	52

3.4.1	Implementasi Autentikasi dan Otorisasi	52
3.4.2	Implementasi <i>Pipeline</i> Vektorisasi Dokumen.....	54
3.4.3	Implementasi <i>Proses Retrieval-Augmented Generation</i>	56
3.4.4	<i>Chat</i> Interaktif.....	58
3.4.5	<i>Details</i> Proses <i>Prompting</i>	59
3.5	Pengembangan Sistem	61
3.5.1	Iterasi I	61
3.5.2	Iterasi II.....	66
3.5.3	Iterasi III.....	71
BAB IV	HASIL DAN PEMBAHASAN.....	77
4.1	Diagram Proses Bisnis	77
4.1.1	Diagram Proses Bisnis	77
4.1.2	Arsitektur Sistem Final	78
4.2	Implementasi	81
4.2.1	Implementasi Autentikasi	81
4.2.2	Implementasi Pipeline Vektorisasi Dokumen.....	86
4.2.3	Implementasi Proses <i>RAG</i>	91
4.2.4	Implementasi <i>Prompt Engineering</i>	97
4.3	Pengujian.....	100
4.3.1	Pengujian Fungsional.....	100
4.3.2	Pengujian Kinerja Non-Fungsional.....	107
4.4	Hasil Implementasi	110
4.4.1	Hasil Antarmuka Pengguna	110
4.4.2	Hasil Keluaran <i>RAG</i>	117
4.5	Pembahasan Kinerja Sistem.....	123
4.5.1	Analisis Reduksi Beban Kerja Tim IT <i>Support</i>	123
4.5.2	Analisis Peningkatan Kecepatan Respon.....	124
4.5.3	Validasi Pemilihan Teknologi.....	125
BAB V	KESIMPULAN	127
5.1	Kesimpulan	127
5.2	Saran.....	128
	DAFTAR PUSTAKA.....	129
	LAMPIRAN	131

DAFTAR TABEL

Tabel 2.1 Tabel perbandingan teknologi <i>artificial intelligence</i>	10
Tabel 3.1 Penjelasan <i>Use case</i> Diagram Admin.....	37
Tabel 3.2 Penjelasan <i>Use case</i> Diagram Dosen.....	38
Tabel 3.3 Penjelasan <i>Use case</i> Diagram Mahasiswa.....	39
Tabel 3.4 Desain Struktur Tabel User Iterasi I	62
Tabel 3.5 Daftar <i>Endpoint</i> Iterasi I	64
Tabel 3.6 Desain Struktur Tabel Iterasi II	66
Tabel 3.7 Daftar <i>Endpoint</i> Iterasi II.....	68
Tabel 3.8 Desain Struktur Tabel Iterasi III	71
Tabel 3.9 Daftar <i>Endpoint</i> Iterasi III.....	73
Tabel 4.1 Komponen Teknologi Arsitektur Sistem	78
Tabel 4.2 Matriks <i>RBAC</i> Sistem Chatbot <i>IT Support</i>	84
Tabel 4.3 Implementasi Metadata <i>Filtering</i> Berdasarkan <i>Role</i>	94
Tabel 4.4 Komponen System Prompt dan Fungsinya.....	98
Tabel 4.5 Hasil Pengujian Autentikasi dan Manajemen <i>User</i>	101
Tabel 4.6 Hasil Pengujian <i>RBAC</i>	102
Tabel 4.7 Hasil Pengujian Manajemen Dokumen dan <i>RAG</i>	103
Tabel 4.8 Hasil Pengujian <i>API Key</i> dan <i>Health Monitoring</i>	105
Tabel 4.9 Hasil Pengujian Waktu Respon Sistem <i>RAG Pipeline</i>	108
Tabel 4.10 Hasil Pengujian Usability Sistem Chatbot <i>IT Support</i>	109

DAFTAR GAMBAR

Gambar 2.1 Arsitektur Umum Sistem <i>Chatbot</i> (Simon Prananta Barus & Evalien, 2022).....	6
Gambar 2.2 Arsitektur Komponen <i>Natural Language Processing</i>	8
Gambar 2.3 Diagram <i>Sequens</i> Proses Penanganan Dokumen	13
Gambar 2.4 Diagram Alir Proses Pencarian Semantik (Arian Masoudi, 2024)	14
Gambar 2.5 Arsitektur Model Transformer yang Menjadi Dasar GPT	16
Gambar 2.6 Implementasi Fungsi OpenAI <i>API</i> dalam <i>Chatbot</i>	19
Gambar 2.7 Struktur Dasar Aplikasi LangChain	21
Gambar 2.8 Taksonomi Teknik Prompt Engineering	23
Gambar 2.9 Standard Prompting.....	24
Gambar 2.10 Perbandingan Standard Prompt dan Zero-Shot Prompting.....	24
Gambar 2.11 Contoh Implementasi Few-shot Prompting	25
Gambar 2.12 Komponen <i>Prompting Engineering</i>	26
Gambar 3.1 <i>Business Process Model and Notation</i>	35
Gambar 3.2 <i>Business Process Model and Notation</i>	36
Gambar 3.3 <i>Use Case Diagram</i>	37
Gambar 3.4 Proses <i>Register</i>	40
Gambar 3.5 Proses Login.....	41
Gambar 3.6 Proses Upload dan Indexing Dokumen PDF.	42
Gambar 3.7 Proses <i>Query RAG</i>	43
Gambar 3.8 Proses <i>Continue Conversation</i>	44
Gambar 3.9 Proses Create <i>API Key</i>	45
Gambar 3.10 <i>Entity Relationship Diagram</i>	46
Gambar 3.11 Tampilan <i>Wireframe</i> Halaman <i>Login</i>	48
Gambar 3.12 Tampilan <i>Wireframe</i> Halaman <i>Dashboard</i>	48
Gambar 3.13 Tampilan <i>Wireframe</i> Halaman <i>Health Monitoring</i>	49
Gambar 3.14 Tampilan <i>Wireframe</i> Halaman <i>Upload Document</i>	50
Gambar 3.15 Halaman Pengelolaan <i>API Key</i>	51
Gambar 3.16 Tampilan <i>Wireframe</i> Halaman <i>Chatbot Interface</i>	52
Gambar 3.17 Implementasi Proses <i>Login</i> dan <i>JWT Generation</i>	53
Gambar 3.18 Implementasi <i>RBAC</i>	54
Gambar 3.19 Implementasi <i>Upload</i> dan Ekstraksi Konten	54
Gambar 3.20 Implementasi Pembentukan <i>Embedding</i>	56

Gambar 3.21 Implementasi Pencarian Semantik	57
Gambar 3.22 Implementasi Metadata <i>Filtering</i>	57
Gambar 3.23 Implementasi <i>Generation</i>	58
Gambar 3.24 <i>System prompt</i>	59
Gambar 3.25 <i>Intent classification</i>	60
Gambar 3.26 <i>Respon generation prompt</i>	60
Gambar 3.27 <i>ERD</i> Iterasi I	62
Gambar 3.28 Desain Arsitektur Sistem Iterasi I	63
Gambar 3.29 <i>ERD</i> Iterasi I	67
Gambar 3.30 Desain Arsitektur Sistem Iterasi II	68
Gambar 3.31 <i>ERD</i> Iterasi III	72
Gambar 3.32 Desain Arsitektur Sistem Iterasi III	73
Gambar 4.1 LangGraph Alur kerja untuk Orkestrasi RAG Pipeline	80
Gambar 4.2 Diagram Alur Proses Login	82
Gambar 4.3 Implementasi Fungsi <i>Login User</i>	83
Gambar 4.4 Implementasi <i>Dependency Function require role</i>	85
Gambar 4.5 Alur Proses Pembuatan <i>API Key</i>	86
Gambar 4.6 Alur Upload Dokumen Asinkron	87
Gambar 4.7 Diagram Ekstraksi Konten Multi-Modal	88
Gambar 4.8 Implementasi Fungsi Ekstraksi <i>PDF</i>	89
Gambar 4.9 Diagram Pembentukan Embedding Hybrid Search	90
Gambar 4.10 Implementasi Fungsi <i>Hybrid Search</i>	91
Gambar 4.11 LangGraph Workflow untuk Orkestrasi RAG Pipeline	93
Gambar 4.12 Implementasi Fungsi <i>Build Metadata Filter</i>	95
Gambar 4.13 Implementasi <i>System Prompt</i>	96
Gambar 4.14 Arsitektur Prompt Engineering Tiga Lapis	97
Gambar 4.15 Implementasi <i>Intent Classification Prompt</i>	99
Gambar 4.16 Halaman Login	111
Gambar 4.17 Halaman Dashboard Administrator	112
Gambar 4.18 Halaman Health Monitoring Layanan Eksternal	113
Gambar 4.19 Halaman Upload Dokumen PDF	114
Gambar 4.20 Halaman Manajemen API Key untuk Akses Programmatic	115
Gambar 4.21 Halaman Chat Interface	116
Gambar 4.22 Halaman Daftar Session dan Detail Chat History	117

Gambar 4.23 Contoh Query Technical dengan Retrieval dan RBAC Filtering	120
Gambar 4.24 Contoh <i>Query Greeting</i> dengan <i>Short-Circuit Routing</i>	121
Gambar 4.25 Contoh <i>Query Out-of-Scope</i> dengan Penolakan dan <i>Redirect</i>	122

BAB I

LATAR BELAKANG

1.1 Latar Belakang

Teknologi informasi kini menjadi elemen penting dalam menunjang berbagai aktivitas, terutama dalam dunia pendidikan. Di lingkungan universitas, sistem informasi yang handal merupakan penunjang utama dalam kegiatan akademik dan administratif. Universitas Islam Indonesia (UII) melalui Badan Sistem Informasi (BSI) menghadapi tantangan serius dalam mengelola permintaan *Information Technology (IT) Support* yang semakin meningkat. Permintaan tersebut tidak hanya terkait masalah teknis, tetapi juga mencakup konsultasi mengenai layanan yang disediakan. Akibat keterbatasan sumber daya manusia, tim *IT Support* sering kali mengalami keterlambatan respon, yang berimbas pada penurunan tingkat kepuasan pengguna (Gabriel, 2019).

Di UII, pertumbuhan jumlah mahasiswa semakin menambah beban kerja tim *IT Support*. Keterbatasan waktu respon yang hanya tersedia selama jam kerja menjadi kendala, terutama ketika permintaan bantuan datang di luar jam operasional seperti pada malam hari atau akhir pekan (Keeheon dkk., 2019). Hal ini menyebabkan penundaan penyelesaian masalah dan mengganggu aktivitas pengguna, sehingga menciptakan ketidaknyamanan bagi mereka yang membutuhkan dukungan segera.

Hasil wawancara dengan tim *IT Support* BSI UII pada hari Jumat, 14 Februari 2025, mengungkapkan bahwa tim merasa kewalahan akibat peningkatan jumlah permintaan bantuan. Banyak permintaan tersebut bersifat berulang dan dapat diotomatisasi, misalnya atur ulang kata sandi, permintaan informasi layanan, dan panduan teknis dasar (Shiva Prasad dkk., 2021). Oleh karena itu, diperlukan solusi yang tidak hanya mengurangi beban kerja tim *IT*, tetapi juga meningkatkan kecepatan dan ketersediaan layanan.

Salah satu solusi inovatif yang diusulkan adalah implementasi *chatbot IT Support* yang terintegrasi dengan kecerdasan buatan (*AI*) dan Basis pengetahuan sendiri. *Chatbot* ini mampu memberikan respon otomatis dan instan terhadap permintaan rutin, sehingga dapat beroperasi selama 24 jam tanpa terpengaruh oleh batasan jam kerja (Anjali dkk., 2021). Dengan demikian, tim *IT Support* dapat fokus menangani masalah yang lebih kompleks, sementara *chatbot* mengelola pertanyaan yang bersifat repetitif.

Berdasarkan permasalahan dan solusi yang telah diidentifikasi, penelitian ini bertujuan untuk mengeksplorasi kinerja implementasi *chatbot IT Support* di BSI UII serta mengkaji dampak jangka panjangnya terhadap operasional *IT* di UII. Diharapkan, penerapan *chatbot* ini dapat mengoptimalkan layanan *IT Support*, meningkatkan kepuasan pengguna, dan secara signifikan mengurangi beban kerja tim *IT Support* (Johanna dkk., 2021).

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah yang muncul adalah bagaimana pengembang dapat memanfaatkan teknologi OpenAI untuk mengembangkan *chatbot IT Support* yang mampu mengurangi beban kerja tim *IT Support* di BSI UII, mempercepat waktu respon, serta menyediakan dukungan yang mudah dipahami, dan dapat diakses sepanjang 24 jam.

1.3 Batasan Masalah

Untuk memastikan fokus penelitian ini tetap terarah dan sesuai dengan tujuan yang ingin dicapai, beberapa batasan permasalahan yang disesuaikan dengan solusi yang telah dikembangkan adalah sebagai berikut:

a. Ruang Lingkup Penelitian

Penelitian ini akan dibatasi pada pengembangan dan implementasi *chatbot IT Support* yang diintegrasikan dengan OpenAI dan basis pengetahuan BSI di UII. Fokus utama adalah pada peningkatan kecepatan respon terhadap permintaan bantuan *IT Support* serta pengurangan beban kerja tim *IT Support* di BSI UII.

b. Keterbatasan Teknologi OpenAI

Teknologi OpenAI yang digunakan dalam penelitian ini dibatasi pada kemampuan pemrosesan bahasa alami (NLP) yang disediakan oleh *API* OpenAI. Penelitian ini tidak akan mencakup pengembangan algoritma AI baru atau penyesuaian mendalam terhadap teknologi OpenAI di luar penggunaan *API* yang tersedia.

c. Konteks Pengguna dan Lingkungan

penelitian ini akan terbatas pada pengguna internal UII, yakni mahasiswa, dosen, dan staf. Implementasi dan pengujian *chatbot* akan dilakukan di lingkungan kampus UII, sehingga hasil penelitian mungkin tidak sepenuhnya berlaku atau dapat diaplikasikan pada institusi atau organisasi lain yang memiliki karakteristik berbeda.

d. Metodologi Pengukuran Kinerja

Kinerja *chatbot* akan diukur berdasarkan beberapa parameter, termasuk kecepatan waktu respon, tingkat kepuasan pengguna, dan pengurangan beban kerja tim *IT Support*. Pengukuran ini akan dilakukan melalui survei pengguna. Penelitian ini tidak akan mencakup analisis dampak finansial secara mendetail dari implementasi *chatbot* ini.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengkaji penerapan teknologi OpenAI dalam mengembangkan dan meningkatkan kinerja *chatbot IT Support* di BSI UIL. Penelitian ini berfokus pada bagaimana *chatbot* dapat diintegrasikan secara efektif untuk mengatasi berbagai tantangan yang dihadapi oleh tim *IT Support*, dengan tujuan utama sebagai berikut.

a. Memanfaatkan Teknologi OpenAI

Memanfaatkan teknologi OpenAI untuk merancang dan mengimplementasikan *chatbot IT Support* yang dapat mengurangi beban kerja tim *IT Support* di BSI UIL.

b. Meningkatkan Kecepatan Respon

Mengoptimalkan *chatbot* untuk mempercepat waktu respon terhadap permintaan pengguna, sehingga pengguna mendapatkan bantuan IT dengan lebih cepat dan efisien.

c. Menyediakan Dukungan yang Mudah Dipahami

Merancang *chatbot* dengan kemampuan pemrosesan bahasa alami yang memastikan bahwa penjelasan dan solusi yang diberikan dapat dengan mudah dipahami oleh pengguna, sehingga meminimalkan kebingungan dan kesalahan pengguna.

d. Menyediakan Layanan Dukungan 24 jam

Menerapkan *chatbot* yang dapat diakses 24 jam, sehingga dukungan IT tersedia bagi pengguna kapan saja, tanpa terbatas oleh jam operasional.

1.5 Metodologi Penelitian

Dalam penelitian ini, pengembang memilih metode prototipe sebagai pendekatan utama untuk pengembangan sistem. Metode ini dipilih karena memungkinkan proses pengulangan yang cepat dan fleksibilitas dalam menyesuaikan kebutuhan pengguna yang mungkin berkembang selama proses pengembangan. Berikut adalah tahapan-tahapan yang dilakukan dalam penerapan metode ini:

a. Komunikasi

Pada tahap ini penulis akan melakukan interaksi dengan pengguna guna menentukan kebutuhan dari pengembangan yang akan diberikan.

b. Perencanaan

Pada tahap ini penulis akan menentukan kebutuhan dari sistem yang akan dikembangkan dengan berdasarkan hasil dari tahap-tahap sebelumnya.

c. Pemodelan Desain

Pada tahap ini penulis membuat UML dan tampilan antarmuka untuk aplikasi hasil pengembangan.

d. Kontruksi Prototipe

Pada tahap ini, penulis mengimplementasikan perancangan sistem ke dalam kode untuk membuat sistem yang dapat digunakan. Ini mencakup pengkodean fitur-fitur yang telah direncanakan sebelumnya serta memberikan contoh alur penggunaan sistem.

e. Pelaksanaan Penyampaian dan Umpan Balik

Pada tahap ini, penulis melakukan pengujian sistem untuk mendapatkan *feedback* dari pengguna. Pengujian dilakukan dengan metode *usability testing*.

1.6 Sistematika Penulisan

Pola penyajian penulisan laporan skripsi ini terbagi ke dalam sejumlah bab dan sub bab dengan susunan sebagai berikut:

a. Bab I Pendahuluan

Bab ini berisi gambaran umum mengenai topik yang dibahas. Mencakup pembahasan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penulisan.

b. Bab II Landasan Teori

Bab ini menguraikan teori-teori dasar yang relevan dengan penelitian ini, termasuk konsep dasar teknologi *chatbot*, penggunaan OpenAI dalam pengembangan *chatbot*, serta studi-studi terkait yang menjadi dasar dalam pengembangan *chatbot* IT Support di BSI UIL.

c. Bab III Metodologi Penelitian

Bab ini menjelaskan persyaratan dan perencanaan pengembangan *chatbot*, bersama dengan metodologi penelitian dan teknik pengembangan sistem. Ini mencakup analisis kebutuhan *Input*, *output*, proses, serta desain pengembangan *chatbot*. Tahapan-tahapan yang digunakan dalam metodologi prototipe juga dijelaskan di sini, termasuk langkah-langkah dari komunikasi awal dengan pengguna hingga pengujian dan penyempurnaan prototipe.

d. Bab IV Hasil dan Pembahasan

Bab ini menguraikan implementasi *chatbot* IT *Support* yang telah dirancang, serta hasil dari penilaian sistem. Bab ini mencakup hasil pengujian fungsionalitas dan kinerja *chatbot*, serta analisis dan diskusi mengenai kinerja sistem dalam mengurangi beban kerja tim IT *Support* dan meningkatkan layanan IT *Support* di UII.

e. Bab V Kesimpulan dan Saran

Bab ini menyimpulkan temuan utama dari penelitian ini dan menjawab rumusan masalah yang telah diidentifikasi. Bagian ini juga memberikan rekomendasi untuk pengembangan *chatbot* IT *Support* di masa depan agar sistem dapat terus ditingkatkan dan dioptimalkan sesuai kebutuhan pengguna dan perkembangan teknologi.

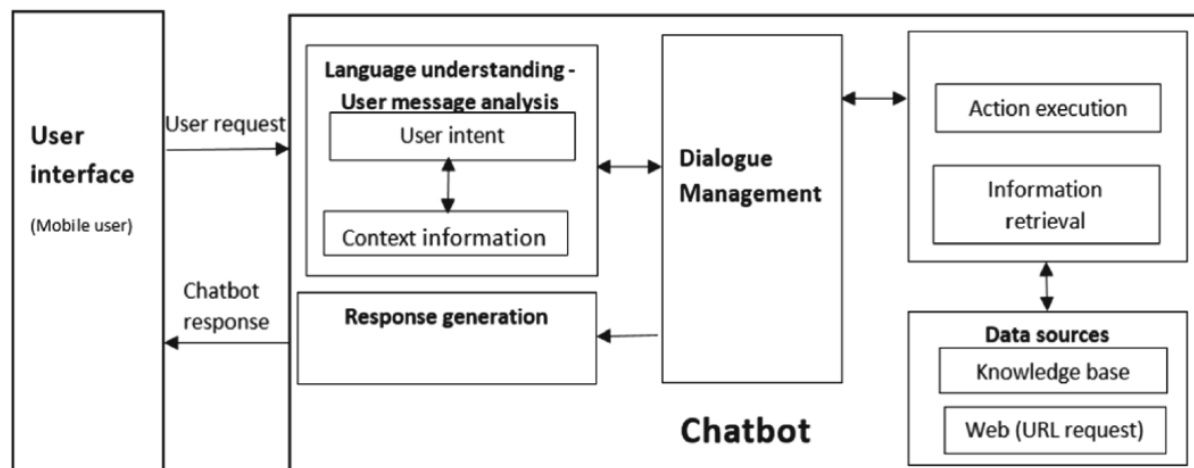
BAB II

LANDASAN TEORI DAN KAJIAN PUSTAKA

2.1 *Chatbot*

Chatbot adalah perangkat lunak yang dirancang untuk meniru percakapan manusia secara alami, baik melalui teks maupun suara, dengan menggunakan teknologi kecerdasan buatan (AI), khususnya *Natural Language Processing* (NLP). Dalam konteks penelitian ini, *chatbot* berfungsi sebagai solusi utama untuk mengatasi tantangan yang diidentifikasi di BSI UII, yaitu volume permintaan bantuan yang tinggi, keterbatasan waktu operasional tim *IT Support*, dan kebutuhan akan respons yang cepat dan konsisten.

Untuk memahami cara kerja *chatbot* secara teknis, dapat dilihat pada arsitektur umum berikut ini:



Gambar 2.1 Arsitektur Umum Sistem *Chatbot*
(Simon Prananta Barus & Evalien, 2022).

Berdasarkan arsitektur tersebut, *chatbot* berfungsi sebagai antarmuka interaktif yang dapat diintegrasikan ke berbagai *platform*, seperti situs *web*, aplikasi pesan instan, atau aplikasi seluler, untuk memberikan solusi otomatis yang cepat dan efisien. Dalam konteks *IT Support* di BSI UII, solusi ini mencakup menjawab pertanyaan teknis, menyelesaikan masalah dasar, atau menjalankan tugas tertentu berdasarkan kebutuhan pengguna seperti pengaturan ulang kata sandi, panduan instalasi perangkat lunak, dan informasi status layanan.

Untuk mencapai kemampuan tersebut, kemajuan teknologi telah memungkinkan integrasi kemampuan pembelajaran mesin (*ML*) dalam pengembangan *chatbot*. Teknologi ini memungkinkan *chatbot* untuk menangkap konteks percakapan yang lebih kompleks,

mengenali maksud pengguna, serta mengekstraksi informasi penting dari percakapan. Sebagai contoh, *chatbot modern* yang menggunakan *API OpenAI* mampu memberikan rekomendasi solusi *IT*, menganalisis kesalahan, hingga menawarkan tanggapan yang disesuaikan berdasarkan preferensi pengguna dan riwayat masalah sebelumnya (Gerald dkk., 2023). Hal ini tidak hanya meningkatkan personalisasi tetapi juga memastikan bahwa kebutuhan pengguna terpenuhi dengan cara yang efisien. Selain itu, *chatbot modern* dirancang untuk menjaga keberlanjutan percakapan dengan memahami konteks percakapan sebelumnya, yang menjadikannya semakin mirip dengan komunikasi manusia.

Mengingat kemampuan-kemampuan canggih tersebut, tidak mengherankan bahwa dalam dunia yang semakin berkembang, *chatbot* telah menjadi komponen penting dalam transformasi *digital*, karena kemampuannya untuk meningkatkan pengalaman pengguna, mengurangi beban kerja manusia, dan mempercepat layanan. Selain itu, *chatbot* memungkinkan penghematan biaya operasional secara signifikan dengan menggantikan sebagian besar tugas manual, khususnya dalam menangani permintaan atau pertanyaan berulang. Penelitian oleh (Neha Atul dkk., 2018). mengungkapkan bahwa *chatbot* dapat diterapkan di berbagai sektor, termasuk layanan teknologi informasi, pendidikan, dan *e-commerce*, dengan fokus pada efisiensi operasional dan kecepatan respons yang lebih tinggi.

Untuk mencapai optimalisasi tersebut, kemajuan teknologi telah memungkinkan Berdasarkan potensi transformasi digital yang telah diuraikan di atas, pemanfaatan *chatbot IT Support* untuk BSI UII merupakan solusi terintegrasi yang menangani pertanyaan repetitif dan sederhana seperti atur ulang kata sandi serta panduan teknis dasar untuk mengurangi beban kerja tim dan memungkinkan fokus pada masalah kompleks, beroperasi 24 jam penuh tanpa batasan jam kerja sehingga mengatasi keterbatasan waktu respon, memastikan konsistensi jawaban melalui teknologi *OpenAI API* yang memberikan respon otomatis dan instan, serta memiliki skalabilitas untuk menangani beberapa permintaan mahasiswa, dosen, dan staf secara bersamaan guna mengoptimalkan layanan *IT Support* dan meningkatkan kepuasan pengguna di lingkungan kampus UII.

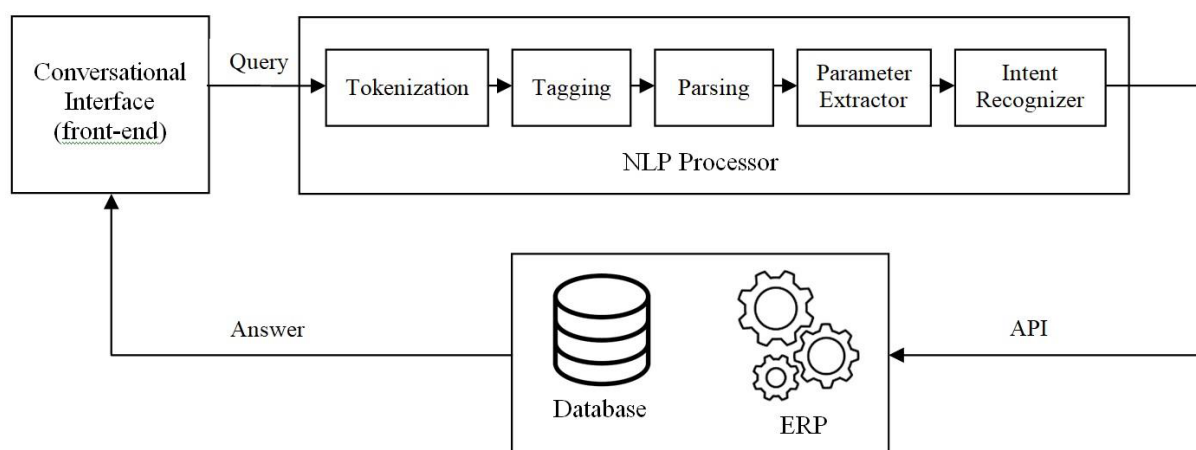
Dengan dukungan teknologi canggih dan implementasi yang tepat, *chatbot* tidak hanya berfungsi sebagai alat layanan pelanggan tetapi juga dapat digunakan untuk mendukung pengambilan keputusan dalam konteks *IT Support*. Sebagai contoh, dalam sektor pendidikan, *chatbot* dapat digunakan untuk memberikan panduan *troubleshooting* kepada pengguna, sementara dalam konteks BSI UII, *chatbot* dapat digunakan untuk menangani reset *password*, *multi factor authentication*, dan *login gateway*. Penelitian lebih lanjut menunjukkan bahwa

chatbot yang dirancang dengan baik dapat meningkatkan tingkat kepuasan pengguna dan mendorong loyalitas dengan memberikan layanan yang responsif dan akurat. Kemampuan *chatbot* untuk beroperasi 24 jam menjadikannya alat yang ideal untuk memberikan layanan tanpa batasan waktu.

2.2 Natural Language Processing (NLP)

Untuk mencapai kemampuan *chatbot* yang telah dijelaskan di atas, terutama dalam memahami variasi pertanyaan pengguna dan memberikan respon yang kontekstual dalam lingkungan *IT Support* BSI UII, diperlukan teknologi pendukung yang mumpuni. Fondasi teknologi utama yang memungkinkan *chatbot* berinteraksi secara natural dengan pengguna adalah *Natural Language Processing* (NLP). Teknologi ini berfungsi untuk menjembatani gap antara bahasa manusia dan bahasa yang dapat dipahami oleh komputer, yang dalam konteks penelitian ini sangat krusial untuk memahami variasi cara pengguna mengekspresikan masalah *IT* mereka.

Natural Language Processing memiliki cakupan pemanfaatan yang luas dalam berbagai aplikasi seperti *chatbot*, *machine translation*, *sentiment analysis*, *information extraction*, dan *document retrieval*. Teknologi ini memungkinkan mesin untuk berkomunikasi secara efisien dengan manusia, mendukung otomatisasi tugas, dan memberikan solusi berbasis data. Dalam konteks *IT Support* BSI UII, kemampuan NLP menjadi fundamental untuk memahami dan merespon berbagai jenis pertanyaan teknis yang diajukan pengguna.



Gambar 2.2 Arsitektur Komponen *Natural Language Processing*
(P. dkk., 2023).

Proses kerja NLP dalam sistem *chatbot IT Support* melibatkan beberapa komponen penting yang saling terintegrasi. Komponen *tokenization* berperan dalam memecah pertanyaan pengguna menjadi unit-unit yang dapat dianalisis secara sistematis. Selanjutnya, *konteks understanding* memungkinkan sistem untuk memahami konteks percakapan, terutama untuk menangani pertanyaan lanjutan yang sering muncul dalam interaksi *IT Support*. Terakhir, *response generation* berfungsi menghasilkan jawaban yang sesuai dan dapat ditindaklanjuti bagi pengguna.

Implementasi teknologi NLP *modern* memanfaatkan teknik ML berbasis pembelajaran mendalam untuk memahami konteks yang lebih kompleks. Salah satu perkembangan signifikan adalah penerapan *transformer-based models* seperti BERT dan GPT, yang memungkinkan analisis konteks secara lebih mendalam dan menghasilkan respons yang lebih akurat. Dalam konteks *IT Support*, kemampuan ini memungkinkan sistem untuk memahami pertanyaan teknis yang kompleks dan memberikan solusi yang tepat sasaran.

Pemanfaatan NLP memberikan manfaat strategis di berbagai sektor. Dalam bidang *customer service*, NLP memungkinkan pengembangan *chatbot* yang dapat merespon pertanyaan pelanggan dengan cepat dan akurat. Di sektor pendidikan, khususnya *IT Support* di universitas seperti BSI UII, NLP digunakan untuk membangun *IT Support assistant* yang mampu menyesuaikan respon berdasarkan Basis Pengetahuan.

Dengan kemampuannya untuk mengotomatisasi proses yang kompleks dan meningkatkan pengalaman pengguna, NLP menjadi komponen kunci dalam ekosistem *AI modern*. Teknologi ini mendukung pencapaian tujuan penelitian ini melalui pengembangan *chatbot IT Support* yang dapat memahami variasi pertanyaan pengguna, memberikan respon kontekstual yang tepat, dan pada akhirnya mengurangi beban kerja tim *IT Support* BSI UII sambil meningkatkan kualitas layanan kepada pengguna.

2.3 Kajian Pustaka

Setelah memahami konsep fundamental *chatbot* dan peran strategis NLP dalam mengoptimalkan interaksi natural antara pengguna dan sistem, langkah selanjutnya adalah menganalisis berbagai penelitian terdahulu yang telah mengimplementasikan teknologi serupa. Untuk memperdalam kajian dalam penelitian ini yang berfokus pada pemanfaatan teknologi OpenAI untuk mengembangkan *chatbot IT Support* di BSI UII, penulis telah melakukan eksplorasi mendalam mengenai berbagai karya ilmiah yang berfokus pada topik pengembangan yang serupa. Berdasarkan analisis yang teliti dan kritis terhadap rumusan

masalah utama, bagaimana teknologi OpenAI dapat mengurangi beban kerja tim *IT Support*, mempercepat waktu respon, dan menyediakan dukungan 24 jam, penulis telah berhasil mengkompilasi data dan temuan penting dari berbagai sumber tersebut ke dalam sebuah tabel yang sistematis, yang akan diuraikan sebagai berikut.

Tabel 2.1 Tabel perbandingan teknologi *artificial intelligence*

Judul Penelitian	Teknologi	Deskripsi	Kelebihan untuk IT Support	Relevansi untuk BSI UII
<i>Enhancing RASA NLU Model for Vietnamese Chatbot</i>	<i>RASA</i>	Digunakan untuk membangun <i>chatbot</i> berbasis dialog kompleks dengan kemampuan <i>intent recognition</i> dan <i>entity extraction</i> yang kuat.	Kontrol penuh atas model, kemampuan kustomisasi tinggi untuk <i>IT Support</i> .	Tinggi, cocok untuk pengembangan <i>chatbot</i> dengan <i>requirement</i> khusus.
<i>Chatbot with Dialogflow for FAQ Services in Matana University Library</i>	<i>Dialogflow</i>	Membantu membangun <i>chatbot</i> yang sederhana hingga menengah, terutama untuk <i>FAQ</i> atau kebutuhan layanan pelanggan.	Setup cepat, <i>interface user-friendly</i> , integrasi mudah untuk <i>FAQ IT Support</i> .	Sedang, terbatas untuk kasus kompleks <i>IT troubleshooting</i> .
<i>Development of OpenAI API-Based Chatbot to Improve User Interaction on the JBMS Website</i>	OpenAI	Digunakan untuk <i>chatbot</i> yang membutuhkan respon kontekstual kompleks, seperti asisten virtual atau layanan pelanggan.	Kemampuan NLP, respons natural, dapat memahami konteks kompleks masalah <i>IT</i> , skalabilitas tinggi	Sangat Tinggi, ideal untuk menangani variasi pertanyaan <i>IT Support</i> yang kompleks
<i>Implementation of Chatbot for ITSM Application Using IBM Watson</i>	<i>IBM Watson</i>	Cocok untuk <i>chatbot</i> yang memerlukan analisis percakapan tingkat lanjut dan integrasi dengan sistem bisnis yang ada	Kemampuan analisis mendalam, integrasi <i>enterprise</i> yang baik	Sedang, <i>over-engineered</i> untuk skala BSI UII.
<i>Wrap Your Mind Around Education: Applying Hugging</i>	<i>Hugging Face</i>	Ideal untuk <i>chatbot</i> berbasis model <i>transformer</i> dengan	Akses ke <i>state-of-the-art</i> model, komunitas	Tinggi, cocok untuk <i>research-oriented implementation</i>

<i>Face to a Chatbot with AI</i>		kemampuan yang sangat spesifik dan mendalam, seperti analisis dokumen atau asisten khusus.	<i>research</i> yang aktif, fleksibilitas tinggi	
----------------------------------	--	--	--	--

Berdasarkan kajian pustaka yang telah disusun, berbagai teknologi *chatbot* menawarkan kelebihan masing-masing dalam mendukung pengembangan sistem berbasis *AI* untuk mengatasi tantangan *IT Support* di BSI UII. Teknologi seperti *RASA*, *Dialogflow*, *IBM Watson*, dan *Hugging Face* memiliki karakteristik yang unik, mulai dari kemampuan *open source* seperti pada *RASA*, kemudahan integrasi pada *Dialogflow*, hingga analisis data mendalam yang ditawarkan oleh *IBM Watson*.

Namun, dari keseluruhan teknologi yang ada, teknologi OpenAI muncul sebagai pilihan yang paling relevan untuk kebutuhan pengembangan *chatbot IT Support* di BSI UII berdasarkan beberapa kriteria evaluasi yang spesifik untuk konteks penelitian ini. Keunggulan utama OpenAI terletak pada kemampuan memberikan jawaban yang presisi, sebagaimana dinyatakan bahwa "*The use of OpenAI API allows chatbot to provide more precise answers and improve user experience*" (Gerald dkk., 2023). yang memungkinkan sistem memberikan respon yang akurat dan tepat sasaran untuk berbagai pertanyaan *IT Support*. Selain itu, OpenAI juga memiliki kemampuan multi fungsi *ChatGPT* yang komprehensif yang memungkinkan *chatbot* "*have a function like ChatGPT that can answer all the difficulties of users when looking for information*" dengan kemampuan "*summarize the article*" dan "*explain the details of the article,*" memberikan pengalaman "*one-stop shopping*" bagi pengguna (Gerald dkk., 2023). Kombinasi kedua keunggulan ini membuat OpenAI menjadi teknologi yang paling sesuai untuk implementasi *chatbot IT Support* di BSI UII karena dapat menangani kompleksitas masalah IT yang beragam dengan tingkat presisi tinggi dalam satu platform terintegrasi.

Selain itu, teknologi OpenAI memungkinkan personalisasi dan pengambilan informasi dengan cepat, yang dapat dioptimalkan untuk layanan-layanan khusus seperti panduan instalasi perangkat lunak, pemecahan masalah teknis, atau bahkan pelaporan otomatis. Keunggulan lainnya adalah kemampuannya dalam mengolah data besar, memungkinkan *chatbot* untuk belajar dari interaksi dan meningkatkan kualitas tanggapannya seiring waktu (Rojas González dkk., 2024). Dalam konteks penelitian ini, penggunaan OpenAI selaras dengan tujuan meningkatkan kecepatan respon dan kepuasan pengguna.

Dengan mempertimbangkan kemampuan NLP yang canggih, fleksibilitas integrasi, dan potensi skalabilitas yang tinggi, OpenAI dinilai sebagai solusi yang paling sesuai untuk kebutuhan pengembangan *chatbot* di BSI UII. Penggunaan teknologi ini juga memungkinkan universitas untuk memanfaatkan perkembangan terkini dalam kecerdasan buatan guna memberikan layanan *IT Support* yang lebih baik dan efisien, yang sejalan dengan semua tujuan penelitian yang telah ditetapkan.

2.4 *Vector Embeddings*

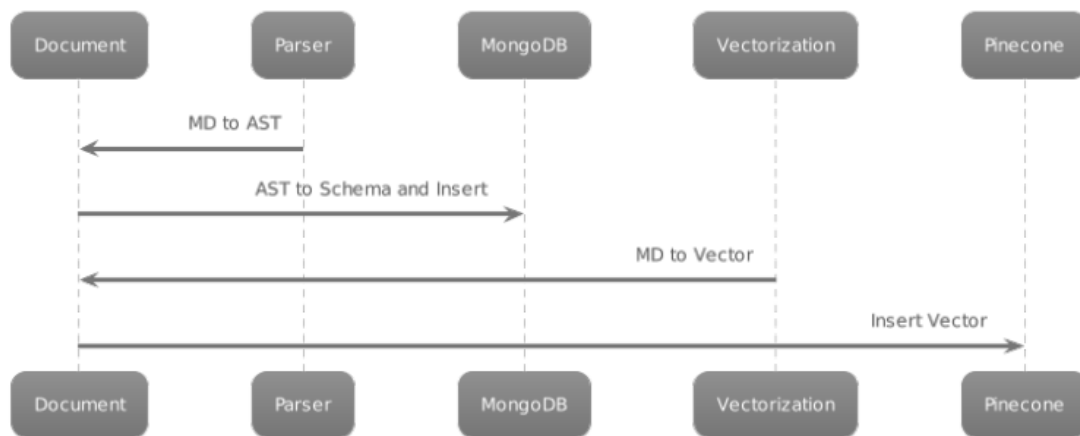
Berdasarkan analisis kajian pustaka yang menunjukkan keunggulan teknologi OpenAI dalam konteks *IT Support*, implementasi yang efektif memerlukan pemahaman mendalam tentang cara kerja teknologi ini. Salah satu komponen teknis fundamental yang memungkinkan *chatbot* berbasis OpenAI memahami dan memproses bahasa natural secara efektif adalah *Vector Embeddings*.

Vector Embeddings merupakan teknologi fundamental dalam pengembangan *chatbot modern* yang memungkinkan sistem untuk memahami dan memproses bahasa natural dengan tingkat akurasi tinggi. Teknologi ini mentransformasi informasi tekstual menjadi representasi numerik yang dapat diproses oleh algoritma ML, memungkinkan *chatbot* untuk mengenali kemiripan semantik dan kontekstual dalam pertanyaan pengguna.

Teknologi *Embedding* memiliki empat keunggulan utama dalam konteks *IT Support*. Pertama, kesamaan semantik yang memungkinkan sistem mengenali pertanyaan serupa meskipun diekspresikan dengan kata-kata berbeda. Kedua, *knowledge retrieval* yang memfasilitasi pencarian cepat dalam Basis pengetahuan *IT Support* BSI UII dengan akurasi tinggi. Ketiga, pelestarian konteks yang mempertahankan makna kontekstual dari dokumentasi teknis kompleks. Keempat, mendukung banyak bahasa yang memungkinkan sistem menangani pertanyaan dalam bahasa Indonesia dan Inggris secara efektif.

Penerapan teknologi OpenAI *API* dalam berbagai domain menunjukkan konsistensi performa yang tinggi. Studi terdahulu membuktikan bahwa teknologi ini telah berhasil diterapkan di perpustakaan dan pusat informasi, dimana sistem menggantikan *chatbot* tradisional berbasis Basis pengetahuan dengan pendekatan adaptif yang menghasilkan respon natural (Subhajit & Navkiran, 2023). *Chatbot* berbasis GPT menunjukkan kemampuan menjawab pertanyaan dengan tingkat akurasi dan fleksibilitas superior, sehingga membantu pengguna memperoleh informasi secara cepat dan efisien.

Teknologi *Embedding* ini tidak hanya meningkatkan akurasi dalam memahami maksud pengguna, tetapi juga mempercepat proses pemrosesan data dan pengambilan keputusan. Hasil penelitian menunjukkan bahwa integrasi *Embedding* ke dalam sistem pencarian semantik dan *chatbot* berbasis *ML* menghasilkan peningkatan signifikan dalam performa system (Nguyen Thi & Maxim, 2021). Dalam konteks *IT Support*, teknologi *Embedding* memungkinkan *chatbot* memahami berbagai jenis pertanyaan pengguna dan memberikan solusi secara cepat, yang berkontribusi langsung pada pengurangan beban kerja tim IT, peningkatan efisiensi operasional, dan penyediaan layanan berkualitas tinggi bagi pengguna akhir (Gerald dkk., 2023).



Gambar 2.3 Diagram *Sequens* Proses Penanganan Dokumen dalam Sistem Vector *Embedding*(Masoudi, 2024).

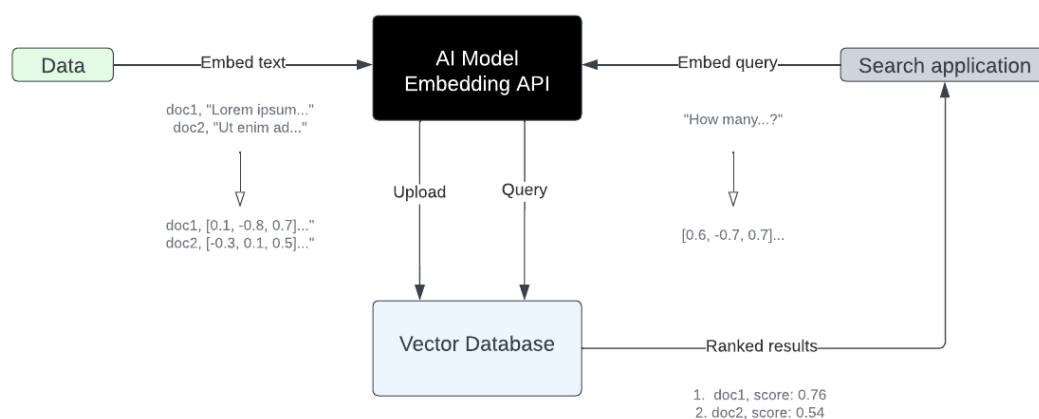
Dengan demikian, *Embedding* menjadi elemen mendasar dalam pengembangan *chatbot modern* yang tidak hanya responsif, tetapi juga cerdas dan mampu memenuhi kebutuhan pengguna dalam berbagai konteks pemanfaatan teknologi informasi.

2.5 Vector Database

Sebagai infrastruktur pendukung penyimpanan dan pengelolaan *vector Embedding* yang telah dibahas sebelumnya, implementasi *chatbot IT Support* di BSI UII memerlukan sistem basis data khusus yang dapat menangani representasi vektor secara efisien. *Vector Database* merupakan jenis basis data yang dirancang khusus untuk menyimpan dan mengelola data dalam bentuk representasi *vector multidimensional* yang dihasilkan dari proses *Embedding*.

Proses *Embedding* melibatkan transformasi data mentah menjadi representasi numerik yang lebih terstruktur, di mana setiap dimensi dalam vektor merepresentasikan karakteristik spesifik atau fitur utama dari data tersebut. Dalam konteks dokumentasi *IT*, *Embedding* dimanfaatkan untuk menangkap hubungan semantik antar kata sehingga memungkinkan sistem untuk mengenali sinonim atau konsep yang terkait meskipun kata-katanya berbeda. Teknologi ini menjadi fondasi dalam pencarian semantik yang lebih canggih dibandingkan dengan metode pencarian tradisional berbasis kata kunci (Shiva Prasad dkk., 2021).

Vector Database memberikan berbagai keunggulan yang signifikan. Pertama, sistem ini mendukung pencarian kemiripan yang cepat dalam Basis pengetahuan yang besar, memungkinkan *retrieval* informasi yang efisien meskipun volume data terus bertambah. Kedua, kemampuan pemahaman semantik memungkinkan sistem menemukan dokumen relevan berdasarkan makna kontekstual, bukan hanya pencocokan kata kunci literal. Ketiga, skalabilitas yang tinggi memastikan sistem dapat menangani pertumbuhan Basis pengetahuan seiring dengan berkembangnya kebutuhan organisasi. Keempat, performa yang optimal mendukung waktu respon yang cepat untuk *chatbot*, memenuhi ekspektasi pengguna akan layanan yang responsif.



Gambar 2.4 Diagram Alir Proses Pencarian Semantik (Masoudi, 2024)

Keunggulan utama *Vector Database* terletak pada kemampuannya untuk memungkinkan pencarian semantik yang lebih canggih dibandingkan dengan sistem basis data tradisional. Pencarian dilakukan berdasarkan relevansi semantik dan konteks, bukan hanya pencocokan kata kunci. Sebagai ilustrasi, ketika pengguna mencari informasi tentang "panduan instalasi," sistem dapat merekomendasikan dokumen yang relevan meskipun istilah tersebut tidak secara

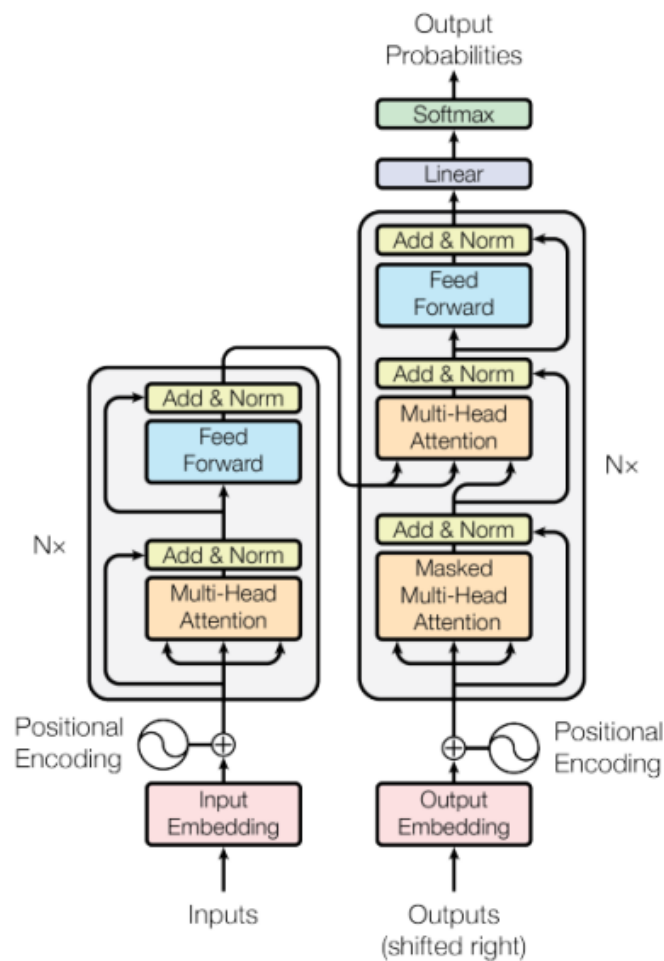
eksplisit disebutkan dalam teks, karena sistem memahami hubungan semantik dengan konsep-konsep terkait seperti "konfigurasi" atau "implementasi" (Arian Masoudi, 2024).

Integrasi teknologi NLP dan model bahasa besar (LLM) memungkinkan *Vector Database* memberikan hasil yang lebih kontekstual dan relevan dengan kebutuhan pengguna. Kombinasi teknologi ini meningkatkan efisiensi pencarian secara signifikan serta mendukung berbagai pemanfaatan seperti layanan pelanggan otomatis, sistem pendidikan adaptif, dan manajemen proyek yang lebih efektif. Dalam konteks UII, implementasi *Vector Database* dapat dimanfaatkan untuk mengoptimalkan sistem pengelolaan pengetahuan, seperti *chatbot* berbasis OpenAI untuk dukungan IT, yang mampu menjawab pertanyaan pengguna secara akurat dan mempercepat pencarian informasi (Gerald dkk., 2023).

Dengan kemampuan-kemampuan tersebut, *Vector Database* menawarkan solusi yang andal untuk meningkatkan efisiensi pengelolaan informasi dan mendukung transformasi digital dalam berbagai institusi pendidikan maupun sektor lainnya. Implementasi teknologi ini di BSI UII diharapkan dapat mengoptimalkan layanan *IT Support*, mengurangi beban kerja tim IT, dan meningkatkan kepuasan pengguna melalui layanan yang lebih responsif dan tersedia sepanjang waktu.

2.6 Model Generative Pre-trained Transformer

Model *Generative Pre-trained Transformer* (GPT) merupakan implementasi *AI* generatif yang berfungsi sebagai komponen inti dalam menghubungkan seluruh teknologi yang telah dibahas sebelumnya. Setelah memahami infrastruktur penyimpanan data melalui *Vector Database*, model GPT mampu memproses dan memanfaatkan representasi vektor dari *Vector Database* untuk memahami konteks serta menghasilkan respon yang relevan dan natural dalam konteks *IT Support* di BSI UII.



Gambar 2.5 Arsitektur Model *Transformer* yang Menjadi Dasar GPT

(Vaswani dkk., t.t.).

2.6.1 Pelatihan Dan Adaptasi

Proses pengembangan model GPT terdiri dari dua tahap fundamental yang saling melengkapi, yaitu pelatihan awal dan penyesuaian tugas khusus. Tahap pertama melibatkan pelatihan awal di mana model GPT dilatih menggunakan pembelajaran tanpa pengawasan pada kumpulan data teks yang sangat besar, seperti Common Crawl dan Wikipedia, yang mencakup miliaran kata dari berbagai domain termasuk dokumentasi teknis dan Basis pengetahuan.

Proses pelatihan awal ini bertujuan untuk memahami pola, struktur, dan hubungan dalam bahasa alami melalui mekanisme perhatian. Mekanisme ini memungkinkan model membangun representasi yang kaya terhadap teks dan menangkap konteks dalam data secara mendalam. Model dilatih untuk memprediksi kata berikutnya dalam teks, sehingga menghasilkan tujuan umum model yang fleksibel untuk berbagai pemanfaatan (Brown dkk., 2020).

Tahap selanjutnya adalah penyesuaian tugas khusus, di mana model GPT dapat disesuaikan untuk tugas tertentu melalui proses pelatihan tambahan dengan *dataset* yang lebih kecil dan spesifik. Pada tahap ini, *dataset* yang relevan dengan tugas tertentu, seperti layanan pelanggan atau pencarian informasi akademik, digunakan untuk menyesuaikan parameter model guna meningkatkan performanya dalam konteks tertentu. penyesuaian tugas khusus memungkinkan GPT untuk menghasilkan respon yang relevan dengan kebutuhan spesifik pengguna, seperti mendukung *chatbot* berbasis layanan pelanggan yang memberikan jawaban natural dan akurat (Simon Prananta Barus & Evalien, 2022).

Keunggulan implementasi untuk BSI UII mencakup efisiensi karena tidak membutuhkan pelatihan dari awal, adaptabilitas yang memungkinkan penyesuaian dengan Basis pengetahuan BSI, efektifitas biaya yang lebih ekonomis dibanding pengembangan model khusus.

2.6.2 Pengaplikasian dalam *Chatbot*

Model GPT telah digunakan secara luas untuk mendukung pengembangan *chatbot* yang tidak hanya interaktif tetapi juga mampu menyesuaikan respon sesuai dengan konteks percakapan dalam lingkungan *IT Support*. Implementasi GPT dalam *chatbot* menawarkan sejumlah keunggulan yang membuatnya unggul dalam memberikan pengalaman pengguna yang lebih baik untuk mencapai tujuan penelitian.

Dengan kemampuan untuk memahami konteks dan makna teks secara mendalam, GPT memungkinkan *chatbot* untuk merespon pertanyaan pengguna dengan relevan dan akurat. Model ini dapat menganalisis maksud pengguna dan menghasilkan jawaban yang sesuai dengan kebutuhan spesifik pengguna, termasuk dalam skenario percakapan yang kompleks seperti prosedur pemecahan masalah bertahap. Keunggulan ini menjadikan GPT ideal untuk pemanfaatan yang memerlukan tingkat pemahaman yang tinggi, seperti layanan pelanggan atau panduan teknis (Subhajit & Navkiran, 2023),(Simon Prananta Barus & Evalien, 2022).

Manfaat spesifik untuk *IT Support* meliputi pemecahan masalah bertahap yang mampu melakukan penyelesaian masalah secara berurutan, profil pengguna yang menyesuaikan kompleksitas jawaban berdasarkan keahlian teknis pengguna, serta percakapan berkelanjutan yang mendukung dialog berkesinambungan untuk penyelesaian masalah. GPT mampu menghasilkan respon yang terasa alami, menyerupai percakapan manusia, sehingga menciptakan pengalaman pengguna yang lebih menyenangkan dalam konteks *IT Support*. *Chatbot* berbasis GPT dapat dengan mudah menangani pertanyaan kompleks, seperti permintaan informasi permasalahan atau rekomendasi solusi teknis, dalam waktu singkat. Hal

ini menjadikannya alat yang sangat efektif untuk meningkatkan efisiensi dalam interaksi berbasis bahasa dan mendukung tujuan mempercepat waktu respon (Gerald dkk., 2023).

Melalui proses penyesuaian tugas khusus, GPT dapat disesuaikan dengan kebutuhan domain tertentu, seperti kesehatan, layanan IT, atau pendidikan. Kemampuan adaptasi ini telah berhasil diimplementasikan secara konkret pada layanan perpustakaan di Matana *University*, di mana GPT digunakan untuk mengembangkan *chatbot* yang mendukung layanan *Frequently Asked Questions (FAQ)* perpustakaan. *Chatbot* ini dirancang khusus untuk menjawab pertanyaan terkait ketersediaan buku, prosedur peminjaman, dan pendaftaran anggota baru, serta menyediakan informasi umum layanan perpustakaan lainnya. Implementasi ini memungkinkan pengguna mendapatkan informasi dengan mudah dan efisien, sehingga meningkatkan kualitas layanan yang diberikan kepada mahasiswa dan staf universitas.

Dengan kemampuan GPT untuk memahami konteks dan menghasilkan respon yang relevan secara otomatis, *chatbot* ini berhasil membantu mengurangi beban kerja staf perpustakaan sekaligus meningkatkan efisiensi operasional dan kepuasan pengguna. Selain itu, *chatbot* dapat beroperasi 24 jam tanpa terbatas waktu, memberikan dukungan berkelanjutan bagi pengguna yang membutuhkan informasi di luar jam kerja perpustakaan. Keberhasilan implementasi ini menunjukkan kemampuan GPT untuk memberikan jawaban yang cepat, akurat, dan sesuai dengan kebutuhan spesifik domain, sehingga menciptakan pengalaman pengguna yang lebih baik dalam pencarian informasi akademik dan mendemonstrasikan potensi besar untuk pemanfaatan serupa di berbagai domain lainnya (Simon Prananta Barus & Evalien, 2022).

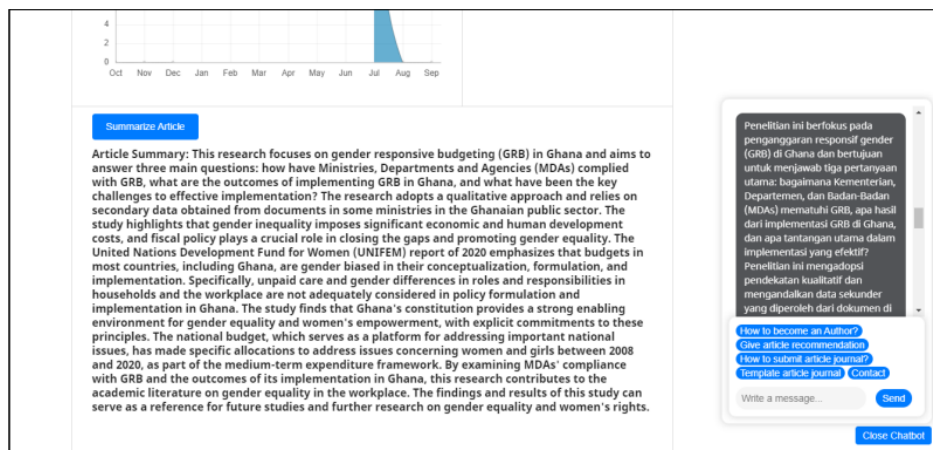
2.7 OpenAI

Model GPT yang telah dibahas sebelumnya merupakan produk unggulan dari OpenAI, sebuah perusahaan yang menyediakan infrastruktur dan layanan kecerdasan buatan generatif. OpenAI berperan sebagai penyedia teknologi dan *Application Programming Interface (API)* yang memungkinkan implementasi praktis dari model GPT, sehingga menjadi komponen krusial dalam arsitektur *chatbot IT Support* di BSI UII.

OpenAI *API* adalah layanan yang memungkinkan pengembang mengakses model *ML* tingkat lanjut yang dikembangkan oleh OpenAI, termasuk GPT-4. *API* ini dirancang untuk mendukung berbagai aplikasi dalam pemrosesan bahasa alami, analisis teks, pembuatan *Embedding* data, dan interaksi berbasis percakapan. Fitur penting yang disediakan OpenAI *API*

adalah kemampuan mengubah teks menjadi representasi numerik berkepadatan tinggi yang mencerminkan hubungan semantik antar kata atau kalimat.

Representasi *Embedding* ini memungkinkan *chatbot* untuk memahami dan memproses teks secara mendalam, seperti mengidentifikasi maksud pengguna, mengklasifikasikan pertanyaan, atau mencari respon yang relevan dalam basis pengetahuan berukuran besar. Selain itu, *Embedding* dapat digunakan untuk pencarian semantik data berdasarkan kesamaan kontekstual. Model GPT yang tersedia melalui OpenAI *API* memiliki kemampuan memahami konteks kompleks dan menghasilkan respon yang koheren serta relevan berdasarkan *Input* yang diberikan.



Gambar 2.6 Implementasi Fungsi OpenAI *API* dalam *Chatbot*
(Gerald dkk., 2023).

Dalam konteks *IT Support* BSI UII, OpenAI *API* menawarkan beberapa kemampuan inti yang relevan. Pertama, pemahaman bahasa alami yang dapat memahami variasi cara pengguna mengungkapkan permasalahan *IT*. Kedua, pembangkitan respon kontekstual yang menghasilkan solusi relevan dengan konteks permasalahan. Ketiga, percakapan berputar jamak yang mendukung percakapan berkelanjutan untuk pemecahan masalah yang kompleks. Keempat, integrasi basis pengetahuan yang dapat diintegrasikan dengan pengetahuan BSI UII. Kelima, dukungan multibahasa yang mendukung komunikasi dalam Bahasa Indonesia dan bahasa Inggris.

Kemampuan ini sangat berguna dalam pengembangan *chatbot* yang memungkinkan pengguna mendapatkan informasi dengan mudah dan efisien, sehingga meningkatkan kualitas layanan yang diberikan (Simon Prananta Barus & Evalien, 2022). Studi sebelumnya menunjukkan bahwa OpenAI telah digunakan untuk mengintegrasikan model GPT dalam

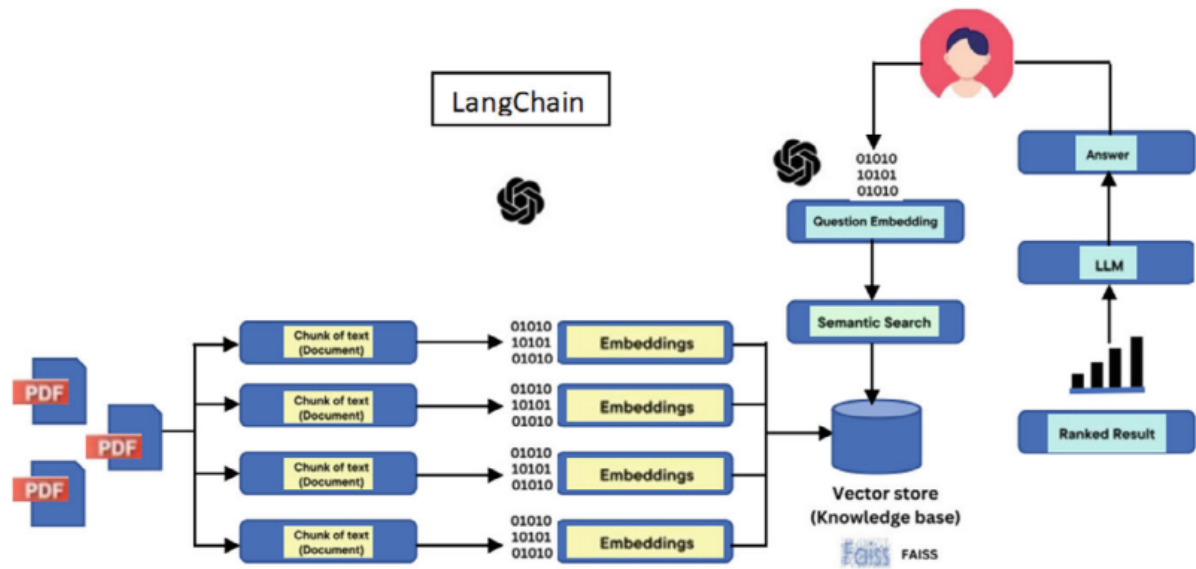
chatbot layanan pelanggan dan pendidikan, yang memungkinkan respons lebih cepat, personalisasi yang lebih baik, dan pengalaman pengguna yang meningkat (Gerald dkk., 2023),(Subhajit & Navkiran, 2023).

Dengan aksesibilitas dan kemampuan canggihnya, termasuk pembuatan *Embedding* data, OpenAI memperkuat peran model GPT dalam mendukung pengembangan *chatbot* di berbagai domain. Hal ini menjadikannya solusi yang efektif untuk meningkatkan efisiensi dan kualitas layanan berbasis kecerdasan buatan dalam mencapai tujuan penelitian yang telah ditetapkan.

2.8 LangChain

Meskipun OpenAI *API* menyediakan kemampuan *AI* generatif yang *powerful*, implementasi *chatbot* yang kokoh memerlukan kerangka kerja pengembangan yang dapat mengintegrasikan berbagai komponen teknologi secara efisien. Dalam konteks ini, LangChain menawarkan solusi sebagai lapisan orkestrasi dan integrasi yang memungkinkan pengembangan aplikasi berbasis *LLM* dengan pendekatan yang lebih terstruktur. Kerangka kerja ini memfasilitasi integrasi mudah antara model bahasa besar seperti GPT dengan data eksternal dan BSI UII, menciptakan ekosistem teknologi yang kohesif dan dapat diandalkan.

Kemampuan inti LangChain untuk mendukung *IT Support* mencakup berbagai aspek teknologi yang saling terintegrasi. Kerangka kerja ini memungkinkan pengolahan dan pengindeksan Basis Pengetahuan BSI UII melalui kemampuan pemrosesan dokumen yang canggih. Selain itu, LangChain mendukung penalaran bertahap untuk pemecahan masalah kompleks melalui pendekatan *Chain of Thought*, yang memungkinkan sistem untuk melakukan analisis bertahap dalam menyelesaikan permasalahan teknis yang kompleks. Sistem ini juga dilengkapi dengan pengelolaan memori yang mempertahankan konteks dalam percakapan panjang, memastikan konsistensi dan relevansi respon sepanjang interaksi pengguna.



Gambar 2.7 Struktur Dasar Aplikasi LangChain

(Devangi. Das dkk., 2024).

Integrasi berbagai *tools* dan *API* menjadi salah satu keunggulan utama LangChain dalam mendukung operasional *IT Support*. Kerangka Kerja ini memungkinkan konektivitas dengan berbagai sistem eksternal yang digunakan di lingkungan BSI UII, menciptakan ekosistem teknologi yang terintegrasi. Lebih lanjut, LangChain menyediakan pengelolaan *prompt* lanjutan untuk mengoptimalkan respon yang dihasilkan oleh sistem, memastikan bahwa setiap interaksi menghasilkan *output* yang relevan dan berkualitas tinggi. *Pipeline orchestration* yang tersedia memungkinkan pengelolaan alur kerja modular, memberikan fleksibilitas dalam pengelolaan proses bisnis yang beragam.

Dalam implementasi teknisnya, Lang Chain menggabungkan kemampuan generatif model AI dengan teknik pemrosesan dokumen seperti pencarian berbasis vektor untuk memastikan hasil yang lebih relevan dan terfokus. Teknologi ini juga memungkinkan integrasi berbagai sumber data melalui *pipeline* modular yang mendukung skalabilitas sistem, memastikan bahwa platform dapat berkembang seiring dengan pertumbuhan kebutuhan BSI UII.

Penelitian yang dilakukan oleh (Devangi. Das dkk., 2024). dalam jurnal "LLM-Based Custom *Chatbot* Using Lang Chain" menunjukkan bahwa Lang Chain memungkinkan pengembang untuk memanfaatkan *LLM* dengan pendekatan baru yang memungkinkan interaksi langsung dengan dokumen *Portable Document Format (PDF)*. Dalam penelitian tersebut, penulis menekankan bahwa "dengan menggunakan LangChain, *chatbot* dapat

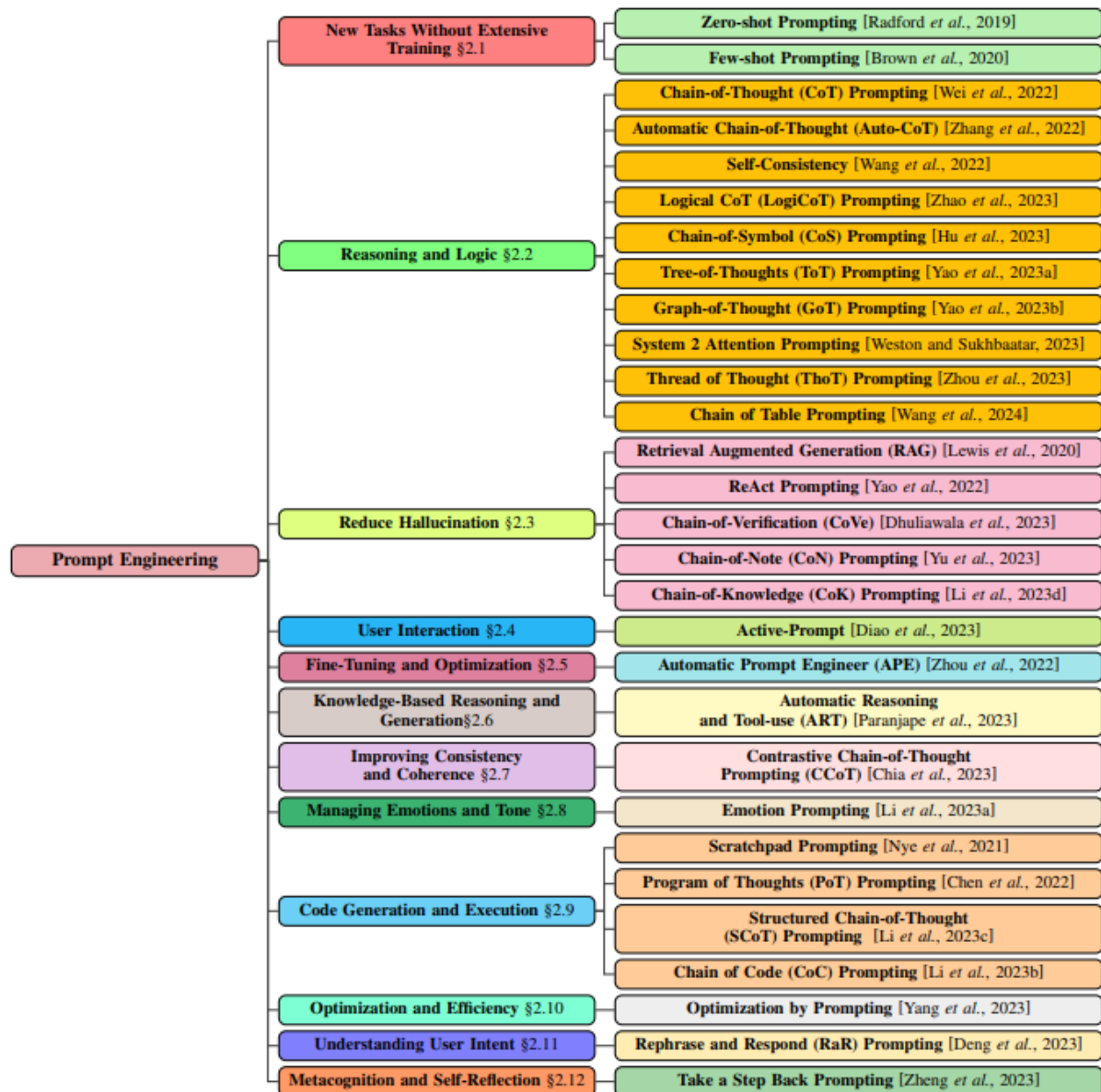
menghasilkan respon kontekstual dengan memanfaatkan kemampuan pencarian semantik yang kuat untuk memastikan relevansi jawaban". Temuan ini memperkuat argumen bahwa LangChain bukan hanya sekedar kerangka kerja integrasi, tetapi juga platform yang mampu meningkatkan kualitas interaksi antara pengguna dan sistem *AI* secara signifikan.

Selanjutnya, kemampuan LangChain dalam menangani berbagai format dokumen, khususnya *PDF*, menjadikannya solusi yang ideal untuk lingkungan akademik seperti BSI UII yang memiliki berbagai jenis dokumentasi teknis. Kerangka kerja ini dapat memproses Basis pengetahuan yang tersimpan dalam berbagai format, mengekstrak informasi yang relevan, dan menyajikannya dalam format yang mudah dipahami oleh pengguna akhir. Dengan demikian, implementasi LangChain dalam sistem *IT Support* tidak hanya meningkatkan efisiensi operasional, tetapi juga memperkaya pengalaman pengguna melalui penyediaan informasi yang lebih akurat dan kontekstual.

2.9 AI Prompting

Dalam kerangka kerja LangChain yang telah dibahas sebelumnya, salah satu aspek krusial yang menentukan kualitas *output chatbot* adalah teknik komunikasi dengan model *AI*. *AI Prompting* merupakan metodologi untuk merancang instruksi yang optimal kepada *LLM*, yang bertujuan memastikan respon yang dihasilkan sesuai dengan konteks dan kebutuhan spesifik *IT Support* di BSI UII.

Dalam pengembangan *chatbot*, *prompting* berfungsi untuk mengarahkan model seperti OpenAI dalam memahami dan menjawab pertanyaan pengguna secara efektif. *Prompt* yang dirancang dengan baik harus mencakup konteks, instruksi yang jelas, dan contoh yang relevan, agar dapat menghasilkan respons berkualitas tinggi yang mendukung tujuan menyediakan dukungan yang mudah dipahami (Banghao dkk., 2024).



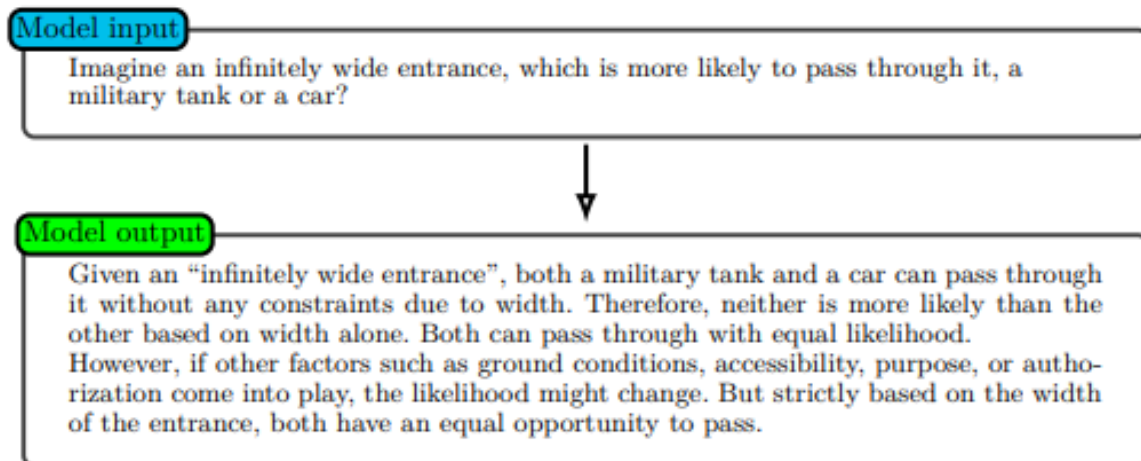
Gambar 2.8 Taksonomi Teknik Prompt Engineering

(Pranab Sahoo dkk., 2024)

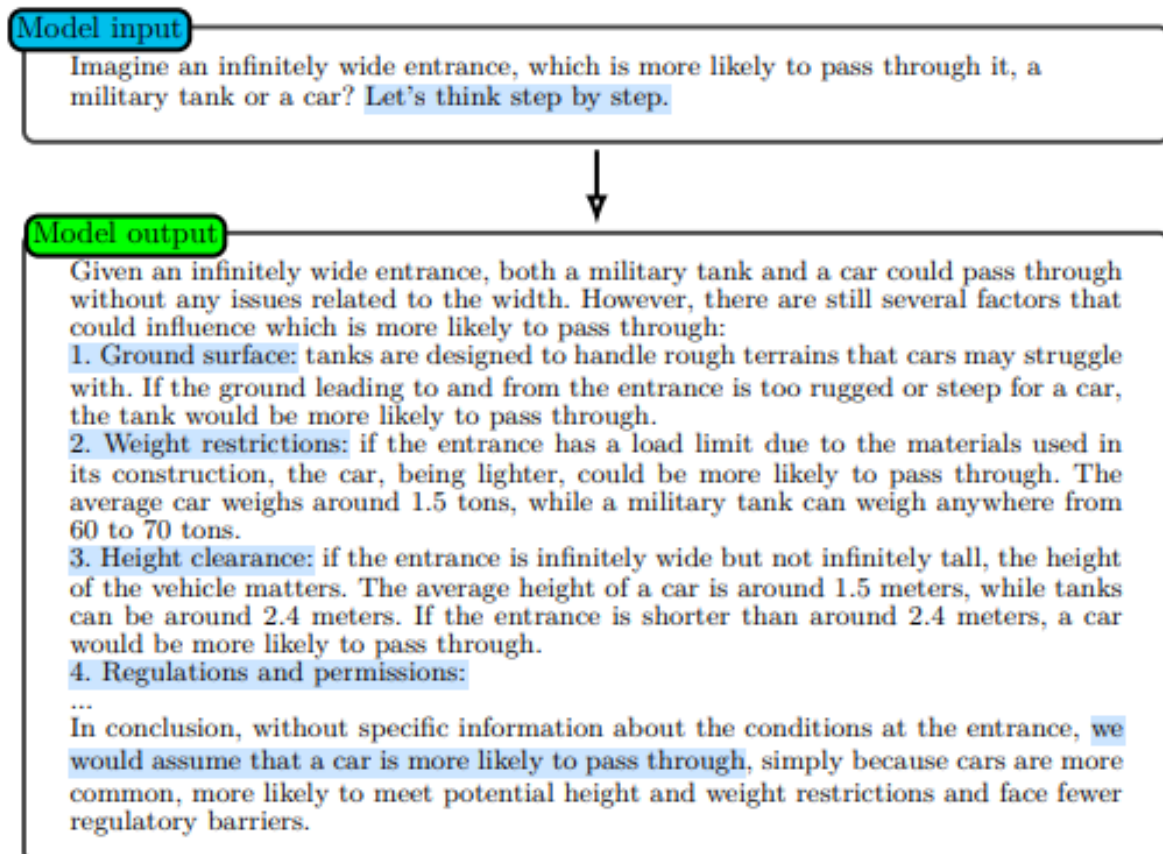
Berdasarkan taksonomi pada Gambar 2.8, terdapat berbagai teknik *prompting* yang dapat diterapkan dalam pengembangan chatbot *IT Support*. Teknik-teknik *prompting* utama yang relevan untuk konteks *IT Support* di BSI UII meliputi beberapa kategori sebagai berikut.

a. *Zero-Shot Prompting*

Model hanya menerima deskripsi tugas tanpa contoh. Teknik ini berguna dalam mengatasi tugas baru tanpa pelatihan tambahan yang ekstensif. *Zero-shot prompting* mengandalkan pengetahuan yang telah dimiliki model, seperti pemanfaatan informasi riwayat untuk menjawab pertanyaan pengguna yang belum pernah ditemui sebelumnya.

Gambar 2.9 Standard *Prompting*

(Banghao dkk., 2024)

Gambar 2.10 Perbandingan *Standard Prompt* dan *Zero-Shot Prompting*

(Banghao dkk., 2024)

b. *Few-Shot Prompting*

Teknik ini menggunakan beberapa contoh *Input-output* untuk membantu model memahami tugas tertentu. Dalam konteks *chatbot*, hal ini berguna untuk tugas-tugas kompleks

yang memerlukan pemahaman mendalam. Misalnya, menyediakan beberapa cara menyatakan pertanyaan umum dalam berbagai formulasi bahasa.

Poor English input:	I eated the purple berries.
Good English output:	I ate the purple berries.
Poor English input:	Thank you for picking me as your designer. I'd appreciate it.
Good English output:	Thank you for choosing me as your designer. I appreciate it.
Poor English input:	The mentioned changes have done. or I did the alteration that you requested. or I changed things you wanted and did the modifications.
Good English output:	The requested changes have been made. or I made the alteration that you requested. or I changed things you wanted and made the modifications.
Poor English input:	I'd be more than happy to work with you in another project.
Good English output:	I'd be more than happy to work with you on another project.
Poor English input:	Please provide me with a short brief of the design you're looking for and that'd be nice if you could share some examples or project you did before.
Good English output:	Please provide me with a brief description of the design you're looking for and that would be nice if you could share some examples or projects you have done before.
Poor English input:	The patient was died.
Good English output:	The patient died.
Poor English input:	We think that Leslie likes ourselves.
Good English output:	We think that Leslie likes us.
Poor English input:	Janet broke Bill on the finger.
Good English output:	Janet broke Bill's finger.
Poor English input:	Mary arranged for, in St. Louis, John to rent a house cheap.
Good English output:	Mary arranged for John to rent a house in St. Louis.
Poor English input:	Today I have went to the store to to buys some many bottle of water.
Good English output:	Today I went to the store to buy some bottles of water.
Poor English input:	I have tried to hit ball with bat, but my swing is has miss.
Good English output:	I tried to hit the ball with the bat, but my swing missed.

Gambar 2.11 Contoh Implementasi *Few-shot Prompting*

(Brown dkk., 2020)

c. *Chain-of-Thought Prompting*

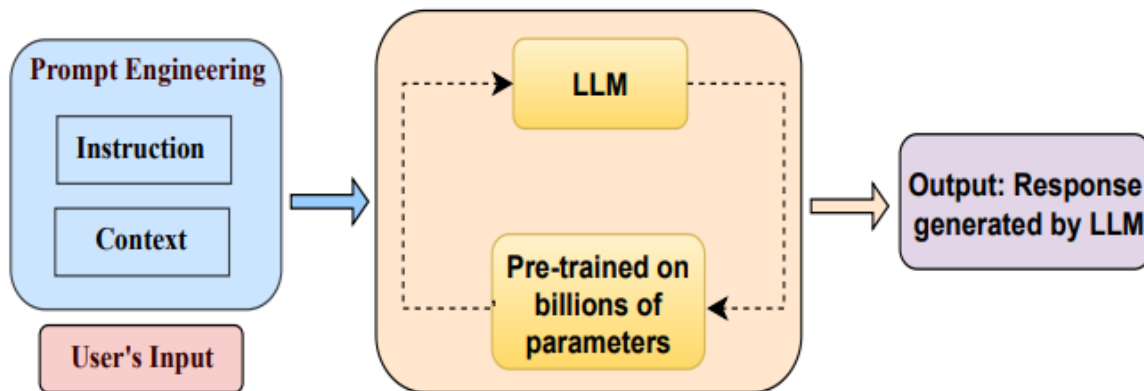
Teknik ini memandu model untuk memecah masalah kompleks menjadi langkah-langkah logis yang lebih kecil. Misalnya, menjawab pertanyaan multi-langkah dalam konteks pemecahan masalah *IT Support*. *Chain-of-Thought* terbukti efektif untuk meningkatkan akurasi dalam penalaran tingkat lanjut (Banghao dkk., 2024).

Pengaplikasian teknik-teknik ini dalam pengembangan chatbot *IT Support* di lingkungan UII memungkinkan sistem memberikan dukungan 24 jam, dengan respon yang relevan dan efektif, sehingga mengurangi beban kerja tim *IT Support* dan meningkatkan kepuasan

pengguna. Hal ini sejalan dengan semua tujuan penelitian yang telah ditetapkan dalam upaya optimalisasi layanan *IT Support* di BSI UII.

2.10 Prompting Engineering

Prompting Engineering merupakan disiplin teknis yang berkembang secara sistematis dan metodis dari konsep *AI Prompting*. Teknik ini berfokus pada optimalisasi interaksi dengan *LLM* seperti OpenAI GPT melalui perancangan *prompting* yang efektif. Pendekatan ini memungkinkan pengembang untuk memandu model dalam menghasilkan respon yang relevan, akurat, dan sesuai konteks untuk mencapai tujuan penelitian yang telah ditetapkan.



Gambar 2.12 Komponen *Prompting Engineering*
(Pranab Sahoo dkk., 2024)

Seperti yang ditunjukkan pada Gambar 2.1, *prompting engineering* melibatkan beberapa komponen kunci yang saling berinteraksi. Komponen-komponen tersebut meliputi instruksi, konteks, *Input* pengguna, dan model *LLM* yang telah dilatih dengan miliaran parameter untuk menghasilkan respon yang sesuai. Model bahasa seperti GPT mengelola *Input* melalui proses tokenisasi, dimana setiap kata atau frasa diubah menjadi token yang dapat diproses oleh sistem. Melalui mekanisme perhatian dalam lapisan *transformer*, model mengelola hubungan antar token berdasarkan konteks yang diberikan, sehingga menghasilkan respons yang sesuai dengan permintaan pengguna. Proses ini memungkinkan *chatbot* memahami pertanyaan pengguna dengan baik dan memberikan jawaban yang relevan serta dapat diandalkan (Banghao dkk., 2024).

- a. Pemberian Instruksi yang Jelas

Memberikan instruksi yang komprehensif kepada model merupakan aspek fundamental dalam *prompting engineering*. *Prompting* yang kurang spesifik dapat menyebabkan respon yang terlalu umum atau tidak relevan dengan kebutuhan pengguna. Dalam konteks pengembangan chatbot *IT Support*, instruksi yang jelas seperti "Berikan solusi untuk masalah koneksi jaringan pada Windows 10" memungkinkan model untuk memberikan jawaban yang lebih terarah dan akurat sesuai dengan permasalahan yang dihadapi (Banghao dkk., 2024).

b. Role-Prompting

Role-prompting melibatkan pemberian peran tertentu kepada model untuk meningkatkan relevansi respon yang dihasilkan. Teknik ini memungkinkan *chatbot* diarahkan untuk berperan sebagai "teknisi IT" yang membantu menyelesaikan masalah perangkat lunak atau perangkat keras. Pendekatan *role-prompting* memungkinkan model memberikan respon yang lebih spesifik dan sesuai konteks, seperti langkah-langkah pemecahan masalah jaringan yang detail dan mudah dipahami oleh pengguna (Banghao dkk., 2024).

c. Triple Quotes

Triple quotes digunakan untuk memisahkan atau mengorganisir bagian-bagian berbeda dalam *prompt* yang kompleks. Teknik ini sangat berguna untuk menangani permintaan multi-langkah dalam chatbot *IT Support*, seperti mencatat masalah pengguna, mengidentifikasi prioritas masalah, dan memberikan solusi teknis secara terstruktur dan sistematis. Penggunaan *triple quotes* membantu model memahami struktur informasi yang kompleks dan memberikan respon yang lebih terorganisir (Banghao dkk., 2024).

d. Resampling dan Iterasi

Karena sifat non-deterministik *LLM*, menjalankan *prompting* yang sama beberapa kali seringkali dapat memberikan hasil yang lebih baik dan bervariasi. Dalam pengembangan chatbot *IT Support*, teknik *resampling* dan iterasi dapat digunakan untuk memilih respon terbaik ketika model memberikan beberapa opsi solusi untuk masalah teknis tertentu. Pendekatan ini memungkinkan sistem untuk mengoptimalkan kualitas jawaban yang diberikan kepada pengguna (Banghao dkk., 2024).

e. One-Shot dan Few-Shot Prompting

One-shot prompting melibatkan pemberian satu contoh kepada model untuk memahami tugas tertentu, sedangkan *few-shot prompting* memberikan beberapa contoh untuk memberikan konteks yang lebih kaya. Dalam chatbot *IT Support*, *few-shot prompting* dapat digunakan untuk memberikan panduan tambahan yang membantu model memahami pola respon yang

diinginkan. Teknik ini terbukti efektif dalam meningkatkan akurasi dan relevansi respon *chatbot* (Brown dkk., 2020).

Dengan pendekatan *prompting engineering* yang tepat, *chatbot* dapat memberikan respon yang spesifik dan relevan, sehingga membantu pengguna menyelesaikan masalah teknis mereka dengan cepat dan efektif. Penelitian menunjukkan bahwa pendekatan *prompt engineering* yang tepat dapat meningkatkan efisiensi dan akurasi model dalam tugas-tugas kompleks (Banghao dkk., 2024). Implementasi teknik-teknik ini secara langsung mendukung pencapaian semua tujuan penelitian yang telah ditetapkan dalam konteks *IT Support* di BSI UII, khususnya dalam hal peningkatan kecepatan respon, pengurangan beban kerja tim IT, dan penyediaan dukungan yang mudah dipahami serta dapat diakses 24 jam.

BAB III

METODELOGI PENELITIAN

3.1 Metodologi Penelitian Prototipe

Metode prototipe merupakan pendekatan pengembangan perangkat lunak yang memungkinkan pengulangan yang cepat dan fleksibilitas dalam menyesuaikan kebutuhan pengguna yang mungkin berkembang selama proses pengembangan. Prototipe berfokus pada pembuatan model awal dari sistem yang akan digunakan untuk mendemonstrasikan, menguji, dan menyempurnakan fungsi-fungsi yang diperlukan sebelum sistem secara keseluruhan dibangun (Apriani dkk., 2024). Pendekatan ini memungkinkan tim pengembang dan pengguna untuk mengamati dan berinteraksi dengan sistem secara visual, mengidentifikasi masalah desain dan kekurangan fungsionalitas sebelum menginvestasikan sumber daya yang signifikan dalam pengembangan. Keunggulan utama metode ini adalah meminimalkan resiko dan biaya yang terkait dengan peluncuran produk yang mungkin tidak sesuai dengan kebutuhan pengguna (Apriani dkk., 2024). Dalam konteks pengembangan *chatbot IT Support* di BSI UII, metode ini memungkinkan validasi konsep dan fungsionalitas sistem melalui interaksi langsung dengan pengguna akhir.

3.2 Analisis Kebutuhan Sistem

Sebelum sistem dikembangkan, diperlukan proses analisis kebutuhan untuk memastikan bahwa seluruh komponen yang dirancang mampu memenuhi tujuan dan fungsionalitas yang diharapkan. Analisis kebutuhan sistem bertujuan untuk mengidentifikasi fitur, batasan, serta interaksi pengguna dengan sistem agar pengembangan dapat dilakukan secara efektif dan efisien.

3.2.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan berbagai fungsi utama yang harus dimiliki oleh sistem agar dapat berjalan sesuai dengan tujuan pengembangannya. Kebutuhan ini mencakup proses, layanan, serta interaksi yang dilakukan antara pengguna dan sistem untuk memastikan seluruh fitur bekerja secara optimal.

- a. Manajemen Dokumen Basis pengetahuan: Sistem harus dapat menerima *upload* dokumen *PDF* berisi basis pengetahuan *IT Support*, memproses dokumen secara otomatis, menyimpan metadata dokumen, serta mendukung *batch upload* untuk beberapa dokumen sekaligus.

- b. Sistem Autentikasi dan Otorisasi: Sistem harus dapat melakukan registrasi pengguna baru, *login* dengan *username* dan *password*, menghasilkan *JavaScript Object Notation (JSON) Web Token (JWT)* untuk autentikasi sesi, mendukung autentikasi menggunakan *API Key* untuk akses programatik, serta memverifikasi dan memvalidasi token atau *API key* pada setiap permintaan yang masuk.
- c. *Role-Based Access Control (RBAC)*: Sistem harus mendukung empat tingkatan *role* yaitu Admin, Dosen, Mahasiswa, dan anonim, dimana Admin memiliki akses penuh ke semua dokumen dan fitur manajemen, Dosen dapat mengakses dokumen publik dan internal, Mahasiswa hanya mengakses dokumen publik, dan pengguna anonim dapat mengakses dokumen publik tanpa autentikasi.
- d. Manajemen *API Key*: Sistem harus memungkinkan Admin untuk membuat *API key* untuk pengguna tertentu, melihat daftar semua *API key* yang aktif, menonaktifkan *API key* yang tidak lagi digunakan, serta mencatat waktu pembuatan dan penggunaan terakhir dari setiap *API key*.
- e. *Retrieval Augmented Generation (RAG)*: Sistem harus dapat menerima pertanyaan dalam bahasa natural Indonesia, melakukan pencarian semantik pada *Vector Database*, mengambil dokumen paling relevan berdasarkan skor kemiripan, menghasilkan jawaban kontekstual menggunakan OpenAI, menerapkan *metadata filtering* berdasarkan *role* pengguna, dan menyertakan sumber referensi dokumen dalam respon.
- f. Manajemen Riwayat Percakapan: Sistem harus dapat menyimpan riwayat percakapan untuk setiap sesi, serta menerapkan *Time To Live* untuk *Chat* riwayat.
- g. Pencarian Semantik Berbasis *Vector* : Sistem harus dapat mengkonversi *query* menjadi *vector Embedding*, melakukan pencarian kemiripan pada *Pinecone Vector Database*, mengambil *top-k* dokumen paling relevan, serta mendukung *metadata filtering* untuk implementasi akses kontrol berdasarkan *role* pengguna.

3.2.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional menjelaskan aspek-aspek kualitas yang harus dimiliki oleh sistem agar dapat beroperasi secara optimal dan memberikan pengalaman terbaik bagi pengguna. Aspek ini tidak berhubungan langsung dengan fungsi utama sistem, tetapi mencakup kinerja, keamanan, keandalan, skalabilitas, dan kemudahan penggunaan.

- a. Keamanan: Sistem harus melakukan *hashing password* pengguna, menerapkan *JWT* autentikasi dengan waktu kedaluwarsa, dilengkapi *rate limiting* untuk mencegah *abuse* dan *DoS attack*, serta menerapkan *CORS policy*.
- b. Performa: Sistem harus memiliki *respons time* untuk *query RAG* kurang dari 10 detik, dapat menangani pengguna aktif bersamaan, menyelesaikan proses *upload* dan *indexing* dokumen, serta memastikan *vector search* di Pinecone dapat diselesaikan dalam waktu kurang dari 10 detik.
- c. Skalabilitas: Sistem harus memiliki arsitektur modular, dapat di *scale* secara *horizontal* dan dapat menangani multiple sesi konkuren.
- d. Ketersediaan: Sistem harus dapat diakses selama 24 jam tanpa mengalami *downtime* yang terencana. Sistem juga harus memiliki mekanisme *error handling* yang efektif, mampu melakukan *recovery* secara otomatis dari kegagalan umum, serta dilengkapi dengan *logging* yang komprehensif untuk mendukung proses *debugging* dan *monitoring*.
- e. Pemeliharaan: Sistem harus memiliki kode yang modular dengan pemisahan tanggung jawab yang jelas, menggunakan variabel lingkungan untuk konfigurasi, menyediakan mekanisme *logging* terstruktur yang mudah.

3.2.3 Analisis Kebutuhan *Input*

Sistem *chatbot IT Support* membutuhkan berbagai jenis *Input* dari pengguna dan administrator untuk dapat beroperasi dengan baik:

- a. *Input Dokumen Basis Pengetahuan*:
 - 1. *Input*: File dokumen *PDF* berisi Basis Pengetahuan *IT Support*.
 - 2. *Details*: Format file *PDF* dengan ukuran maksimal 10 MB per file, berisi dokumentasi teknis, panduan pemecahan masalah, atau *FAQ IT Support* BSI UIN, dilengkapi metadata berupa kategori dokumen dan tingkat sensitivitas.
- b. *Input User Query*:
 - 1. *Input*: Pertanyaan atau query dalam bentuk teks bahasa natural.
 - 2. *Details*: Teks dalam bahasa Indonesia dengan panjang minimal 1 karakter dan maksimal 150 karakter, berisi pertanyaan terkait pemecahan masalah *IT* atau informasi teknis.
- c. *Input Authentication*:
 - 1. *Input*: *Username*, *password*, *email*, dan *role* pengguna.

2. *Details*: *Username* berupa *string*, *password* minimal 8 karakter dengan kombinasi huruf dan angka, *email* dalam format valid, serta *role* yang diberikan oleh admin.
- d. *Input Session Management*:
 1. *Input*: *Session ID* untuk melacak konteks percakapan.
 2. *Details*: *String* dalam format *UUID* yang digunakan untuk melacak konteks percakapan dan *Chat history*.
- e. *Input Query Parameter*:
 1. *Input*: Parameter konfigurasi untuk proses *query RAG*.
 2. *Details*: Parameter *top-k* berupa integer untuk menentukan jumlah dokumen yang di *retrieve*.
- f. *Input Manajemen API Key*:
 1. *Input*: *User ID* dan deskripsi untuk pembuatan *API key*.
 2. *Details*: *User ID* target yang akan dibuatkan *API key* berupa *integer* atau *UUID*, dilengkapi dengan deskripsi untuk *API key* yang bersifat tidak wajib, digunakan oleh administrator untuk memberikan akses kepada *user* tertentu.

3.2.4 Analisis Kebutuhan Output

Sistem harus menghasilkan berbagai jenis *output* yang informatif dan sesuai dengan kebutuhan pengguna:

- a. *Output Respon Query RAG*:
 1. *Output*: Jawaban kontekstual dalam bahasa natural berdasarkan Basis Pengetahuan.
 2. *Details*: Respon dengan tone profesional dan mudah dipahami, dalam format *JSON* yang mencakup *field answer* dan *session_id*, menggunakan informasi dari dokumen yang paling relevan.
- b. *Output Status Upload Dokumen*:
 1. *Output*: Informasi status proses upload dan pemrosesan dokumen.
 2. *Details*: Status *upload success* atau *failed*, *Document ID* yang di *generate* sistem, pemrosesan status, jumlah *chunks* yang di *generate* dari dokumen, status *vector indexing*.
- c. *Output Pesan Error*:
 1. *Output*: Pesan *error* yang informatif ketika terjadi kesalahan.

2. *Details*: Sistem harus menyediakan kode status *Hypertext Transfer Protocol (HTTP)* yang sesuai, disertai dengan jenis atau kategori kesalahan, pesan kesalahan yang deskriptif dan jelas.

3.2.5 Analisis Kebutuhan Tampilan Antarmuka

Antarmuka pengguna yang diperlukan adalah sebagai berikut:

- a. Halaman Login
 1. *Form Input* untuk *username* atau *email* dan *password*.
 2. *Checkbox "Remember me"* untuk menyimpan *session*.
 3. Tombol untuk *submit form login*.
 4. Pesan *error* untuk *login* yang gagal.
 5. Pesan sukses setelah *login* berhasil.
- b. Halaman Pengelolaan *API Key*
 1. *Form Input* untuk membuat *API key* baru dengan *field User ID* dan deskripsi.
 2. Tombol untuk *submit* pembuatan *API key* baru.
 3. Tabel atau daftar untuk menampilkan semua *API key* yang aktif dengan informasi *key ID*, *creation date*, dan status.
 4. Tombol untuk menonaktifkan *API key* yang dipilih.
 5. Filter atau *search* untuk menampilkan *API key* berdasarkan *user* tertentu.
 6. Pesan konfirmasi untuk aksi pembuatan dan penghapusan *API key* yang berhasil.
 7. Pesan *error* jika terjadi kegagalan dalam proses pengelolaan *API key*.
- c. Halaman *Upload Dokumen*
 1. *Form Input* untuk memilih *file PDF* yang akan di *upload*, support *single* atau *multiple files*.
 2. Area *drag-and-drop* untuk memudahkan *upload file PDF*.
 3. *Form Input* untuk *Uniform Resource Locator (URL)* referensi dokumen.
 4. *Form Input* untuk *custom* metadata berupa kategori dokumen dan tingkat sensitivitas.
 5. Tombol untuk submit *upload* dokumen.
 6. indikator *loading* untuk menunjukkan proses *upload* dan *processing* dokumen.
 7. Daftar file yang berhasil di *upload* dengan status *processing*, *queued*, *processing*, *completed*, *failed*.

8. Pesan konfirmasi untuk upload yang berhasil dengan informasi jumlah *chunks* yang di *generate*.
 9. Pesan *error* jika *upload* gagal.
- d. Halaman *Chatbot Interface*
1. Area tampilan percakapan untuk menampilkan pesan antara *user* dan *chatbot*.
 2. Form *Input* kolom *Chat* untuk mengetikkan pertanyaan.
 3. Tombol *submit* untuk mengirim pesan ke *chatbot*.
 4. Tombol login untuk autentikasi pengguna.
 5. Indikator *loading* saat chatbot sedang memproses jawaban.
 6. Pesan *error* jika *query* gagal diproses.
- e. Halaman *Health Monitoring*
1. *Panel* atau *card* untuk menampilkan status kesehatan *API* sistem.
 2. *Panel* atau *card* untuk menampilkan status kesehatan Pinecone *Vector Database*.
 3. *Panel* atau *card* untuk menampilkan status kesehatan OpenAI *service*.
 4. Indikator visual untuk menunjukkan status setiap *service*.
 5. *Timestamp* untuk setiap status monitoring.
 6. Tombol *refreshes* untuk memperbarui status *health* secara manual.
 7. Pesan *error* atau alert jika ada *service* yang *down* atau mengalami masalah.
- f. Halaman Dashboard Admin
1. *Panel* navigasi atau sidebar untuk mengakses berbagai fitur.
 2. *Card* atau *widget* untuk menampilkan statistik sistem secara ringkas.
 3. *Card* atau *widget* untuk menampilkan status kesehatan sistem dengan indikator visual.
 4. Tabel atau daftar *recent activities*.
 5. *Quick access button* untuk fitur-fitur utama.
 6. Grafik atau *chart* untuk menampilkan statistik penggunaan.
 7. *Alert* atau *notification* panel untuk menampilkan *system alerts* atau *warning*.
 8. *User profile section* dengan informasi *user* yang sedang *login* dan tombol *logout*.

3.3 Perancangan Desain Sistem

Perancangan desain sistem ini menjelaskan bagaimana sistem *chatbot IT Support* dikembangkan untuk memproses pertanyaan pengguna, mengambil informasi relevan dari Basis pengetahuan menggunakan *RAG*, dan menghasilkan respon kontekstual melalui OpenAI

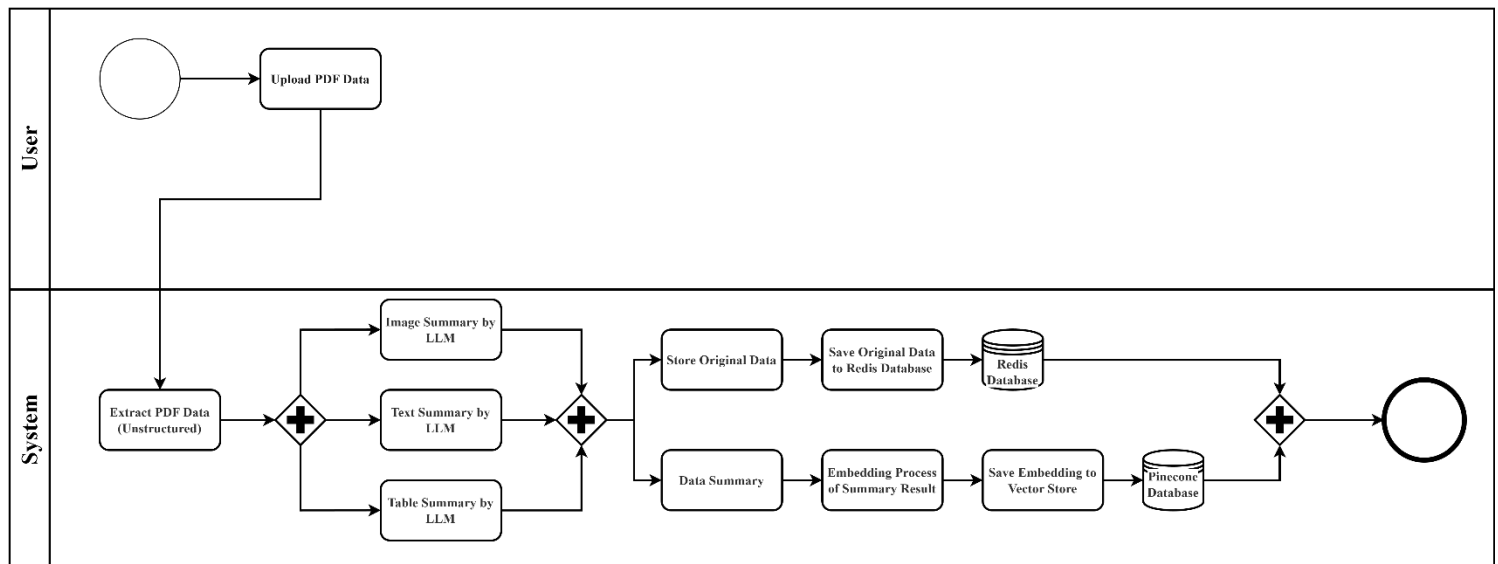
API. Bagian ini mencakup gambaran umum sistem, *use case diagram*, *sequence diagram*, *activity diagram*, perancangan *Entity Relationship Diagram (ERD)*, arsitektur *RAG pipeline*, serta perancangan desain *API endpoints*. Berikut adalah detail dari masing-masing bagian.

3.3.1 Gambaran Umum Sistem

Sistem yang dikembangkan dimodelkan menggunakan *Business Process Model and Notation (BPMN)* untuk menggambarkan dua alur utama yang saling terhubung dalam pengolahan dokumen *IT Support*. Proses pertama adalah vektorisasi dokumen, yang berfokus pada ekstraksi, peringkasan, dan penyimpanan hasil *Embedding* ke basis data vektor dan metadata. Proses kedua adalah kueri pengguna, yang menjelaskan bagaimana sistem menanggapi pertanyaan dengan melakukan pencarian semantik berdasarkan *Embedding* yang telah dihasilkan sebelumnya. Kedua proses ini divisualisasikan dalam diagram BPMN untuk memperlihatkan urutan aktivitas, aliran data, serta interaksi antara pengguna dan sistem.

a. Proses Vektorisasi Dokumen

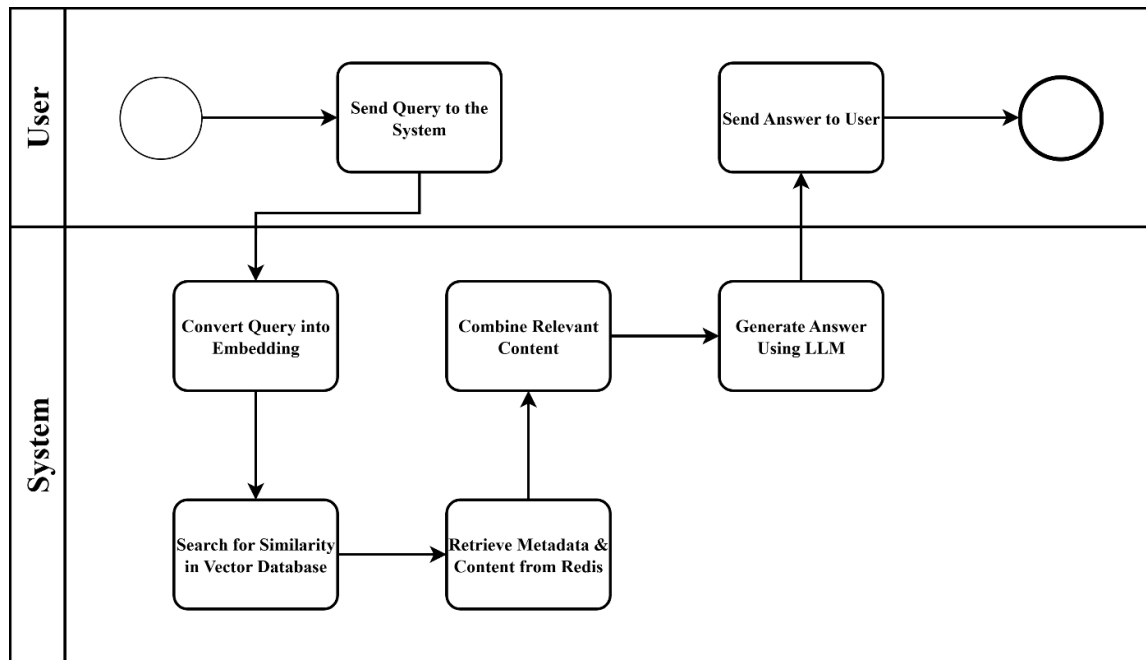
Sistem memproses dokumen *PDF IT Support* melalui ekstraksi *multimodal*, menghasilkan ringkasan menggunakan *OpenAI LLM*, lalu menyimpan hasil *Embedding* ke *Pinecone Vector Database*, serta menyimpan data asli dan *metadata* ke *Redis* untuk kebutuhan pencarian cepat.



Gambar 3.1 *Business Process Model and Notation*.

b. Proses Kueri Pengguna

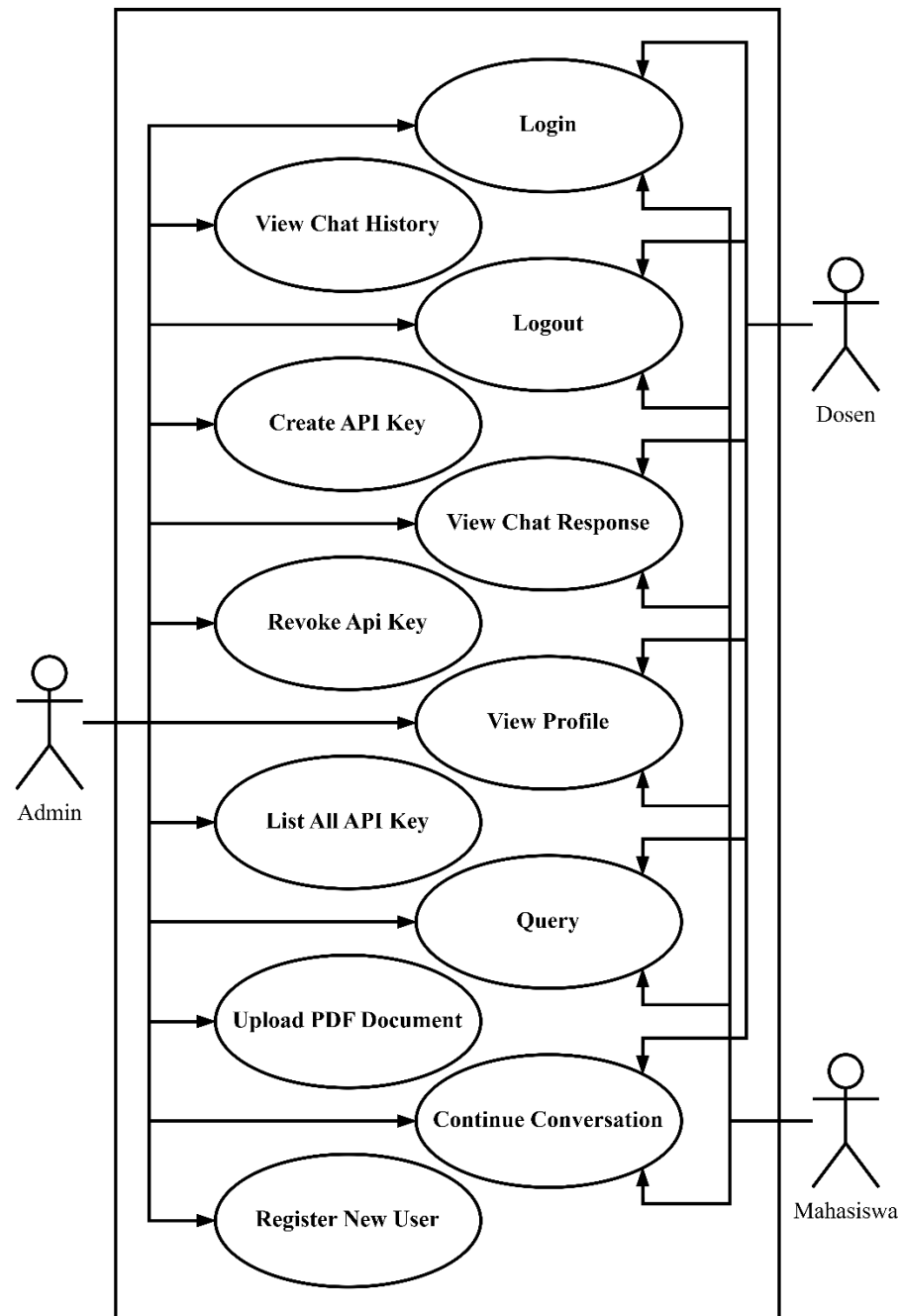
Ketika pengguna mengajukan pertanyaan, sistem mengonversi kueri tersebut menjadi *Embedding*, mencari kesamaan semantik di Pinecone, mengambil konteks relevan dari Redis, dan menghasilkan respon kontekstual menggunakan OpenAI *LLM* dengan mekanisme *role-based metadata filtering* agar hasil sesuai dengan hak akses pengguna.



Gambar 3.2 *Business Process Model and Notation*.

3.3.2 Use Case Diagram

Use case diagram menggambarkan interaksi antara aktor (pengguna) dengan sistem. Berikut adalah diagram *use case* untuk sistem yang dikembangkan. Gambar 3.3 adalah diagram *use case* yang menunjukkan hubungan antara admin dan sistem. Diagram ini menggambarkan fitur-fitur yang bisa admin akses, mulai dari *login*, *logout* dan melanjutkan percakapan. Selain itu admin juga dapat mengelola *API Key*.



Gambar 3.3 Use Case Diagram.

Tabel 3.1 Penjelasan Use case Diagram Admin.

NO	Aktor	Use Case	Deskripsi
1	Admin	<i>Register User</i>	Admin dapat mendaftarkan pengguna baru ke dalam sistem dengan mengisi <i>username</i> , <i>email</i> , <i>password</i> , <i>full name</i> , dan <i>role</i> .
2	Admin	<i>Login</i>	Admin melakukan autentikasi dengan memasukkan <i>username</i> dan <i>password</i> untuk mendapatkan <i>JWT</i> token.

3	Admin	<i>Logout</i>	Admin keluar dari sistem dengan menghapus <i>session</i> dan token autentikasi.
4	Admin	<i>View Profile</i>	Admin dapat melihat informasi profil pribadi mereka yang mencakup <i>username</i> , <i>email</i> , <i>full name</i> , <i>role</i> , dan status akun.
5	Admin	<i>Upload Document</i>	Admin dapat mengunggah dokumen <i>PDF</i> ke sistem untuk diproses dan diindeks ke dalam <i>Vector Database</i> Pinecone.
6	Admin	<i>Ask Question</i>	Admin dapat mengajukan pertanyaan kepada chatbot dengan akses penuh terhadap semua dokumen tanpa filter metadata.
7	Admin	<i>Continue Conversation</i>	Admin dapat melanjutkan percakapan sebelumnya dengan menggunakan <i>session_id</i> yang sama.
8	Admin	Create API Key	Admin dapat membuat <i>API key</i> baru untuk pengguna lain dalam sistem.
9	Admin	<i>List All API Keys</i>	Admin dapat melihat daftar semua <i>API key</i> yang ada dalam sistem dengan informasi <i>key_prefix</i> , nama, <i>user</i> pemilik, status aktif, dan waktu penggunaan terakhir.
10	Admin	<i>View API Key Details</i>	Admin dapat melihat detail informasi dari <i>API key</i> tertentu berdasarkan <i>key_id</i> .
11	Admin	<i>Revoke API Key</i>	Admin dapat menonaktifkan <i>API key</i> yang tidak lagi digunakan atau berpotensi disalahgunakan.
12	Admin	<i>List User API Key</i>	Admin dapat melihat semua <i>API key</i> yang dimiliki oleh seorang <i>user</i> tertentu.

Setelah kita memahami hubungan admin dengan sistem selanjutnya yaitu memahami hubungan dosen dengan sistem. Pada diagram Gambar 3.3, dosen dapat melakukan beberapa aktivitas utama seperti login, melakukan *query*, melihat *profile*, melanjutkan percakapan dan *logout*. Diagram ini menunjukkan apa saja yang dapat dilakukan oleh dosen terhadap sistem.

Tabel 3.2 Penjelasan *Use case* Diagram Dosen.

NO	Aktor	Use Case	Deskripsi
1	Dosen	<i>Login</i>	Dosen melakukan autentikasi dengan memasukkan <i>username</i> dan <i>password</i> untuk mendapatkan <i>JWT</i> token.
2	Dosen	<i>Logout</i>	Dosen keluar dari sistem dengan menghapus <i>session</i> dan token autentikasi.
3	Dosen	<i>View Profile</i>	Dosen dapat melihat informasi profil pribadi mereka yang mencakup <i>username</i> , <i>email</i> , <i>full name</i> , <i>role</i> , dan status akun.
4	Dosen	<i>Ask Question</i>	Dosen dapat mengajukan pertanyaan kepada chatbot dengan akses penuh terhadap semua dokumen tanpa filter metadata.

5	Dosen	<i>Continue Conversation</i>	Dosen dapat melanjutkan percakapan sebelumnya dengan menggunakan <i>session_id</i> yang sama.
6	Dosen	<i>View Chat Response</i>	Dosen dapat melihat hasil chat yang di berikan oleh sistem

Setelah kita memahami hubungan admin, dosen dengan sistem selanjutnya yaitu memahami hubungan mahasiswa dengan sistem. Pada diagram Gambar 3.3, mahasiswa dapat melakukan beberapa aktivitas utama seperti *login*, melakukan *query*, melihat *profile*, melanjutkan percakapan dan *logout*. Diagram ini menunjukkan apa saja yang dapat di lakukan oleh dosen terhadap sistem.

Tabel 3.3 Penjelasan *Use case* Diagram Mahasiswa.

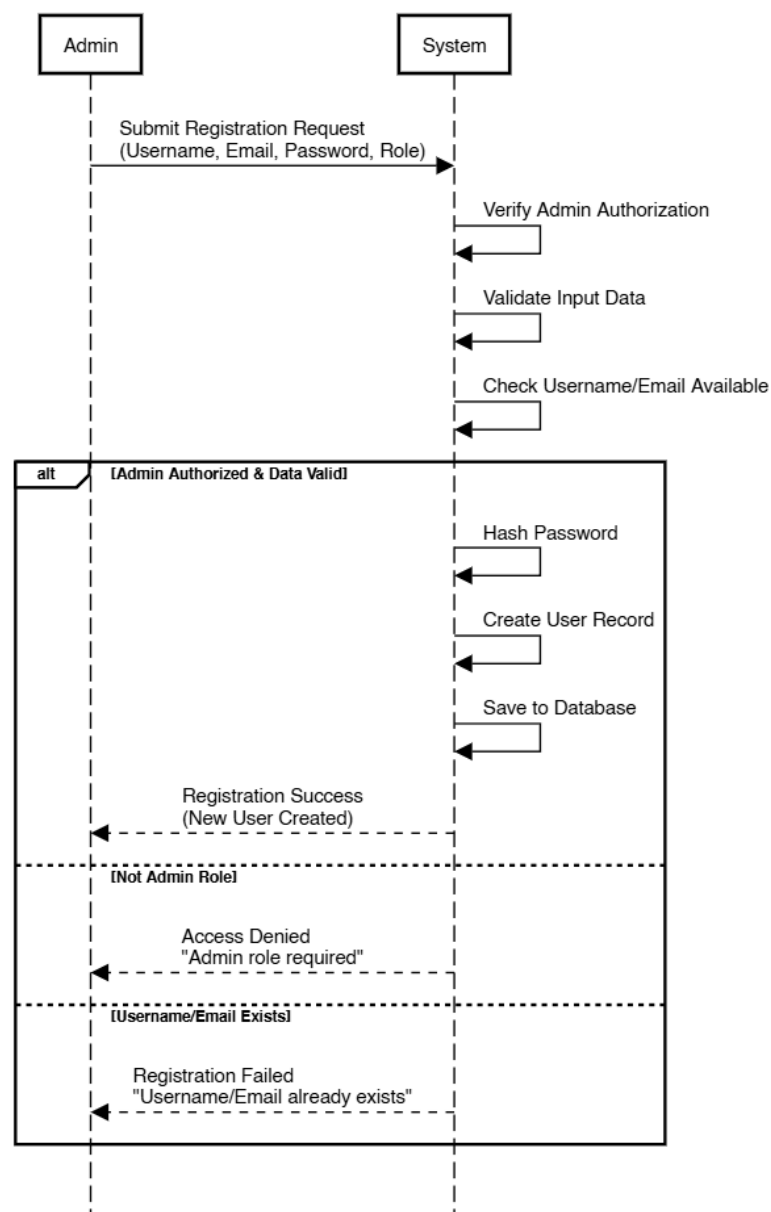
NO	Aktor	Use Case	Deskripsi
1	Mahasiswa	<i>Login</i>	Mahasiswa melakukan autentikasi dengan memasukkan <i>username</i> dan <i>password</i> untuk mendapatkan <i>JWT</i> token.
2	Mahasiswa	<i>Logout</i>	Mahasiswa keluar dari sistem dengan menghapus <i>session</i> dan token autentikasi.
3	Mahasiswa	<i>View Profile</i>	Mahasiswa dapat melihat informasi profil pribadi mereka yang mencakup <i>username</i> , <i>email</i> , <i>full name</i> , <i>role</i> , dan status akun.
4	Mahasiswa	<i>Ask Question</i>	Mahasiswa dapat mengajukan pertanyaan kepada chatbot dengan akses penuh terhadap semua dokumen tanpa filter metadata.
5	Mahasiswa	<i>Continue Conversation</i>	Mahasiswa dapat melanjutkan percakapan sebelumnya dengan menggunakan <i>session_id</i> yang sama.
6	Mahasiswa	<i>View Chat Response</i>	Mahasiswa dapat melihat hasil chat yang di berikan oleh sistem

3.3.3 Sequence Diagram

Berikut adalah *sequence* diagram yang menggambarkan interaksi *user* dengan objek yang terkait di dalam sistem:

a. Proses *Register*

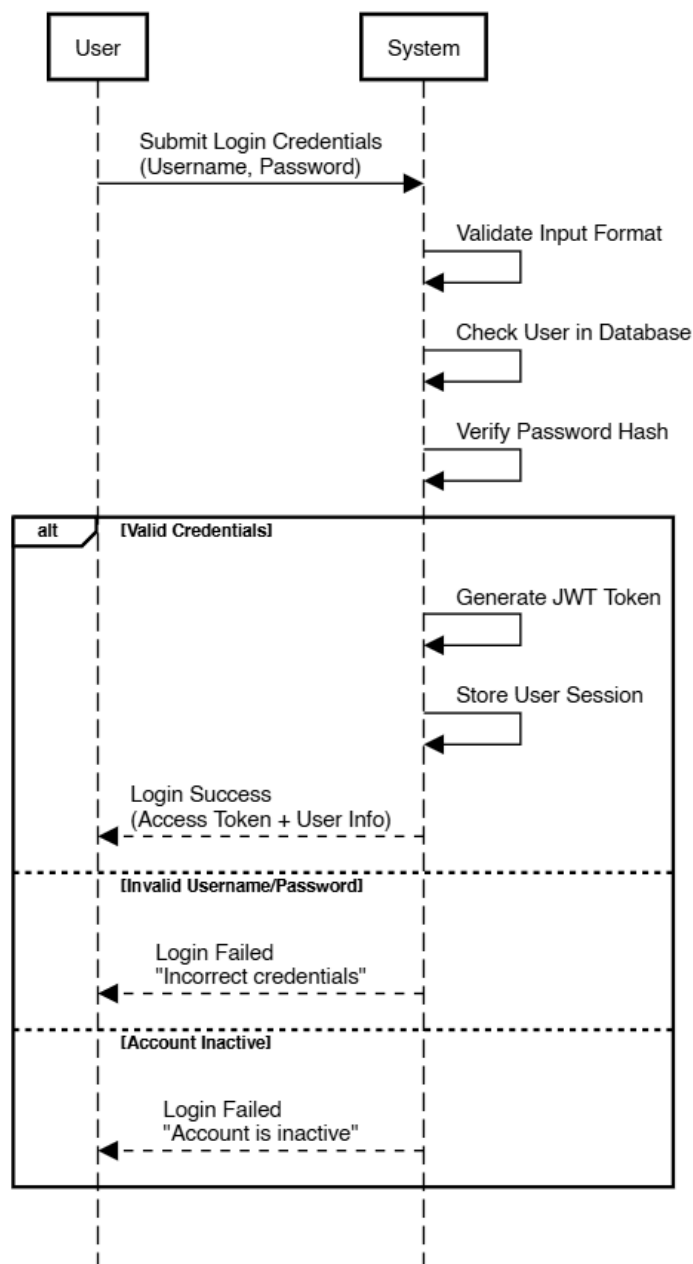
Gambar 3.4 menggambarkan proses registrasi pengguna baru dalam sistem. Proses ini hanya dapat dilakukan oleh admin yang telah terautentikasi. Sistem melakukan verifikasi otorisasi admin, validasi data *Input*, pengecekan ketersediaan *username* dan *email*, *hashing password*, pembuatan *record* pengguna baru, dan penyimpanan ke *database*.



Gambar 3.4 Proses *Register*.

b. Proses *Login*

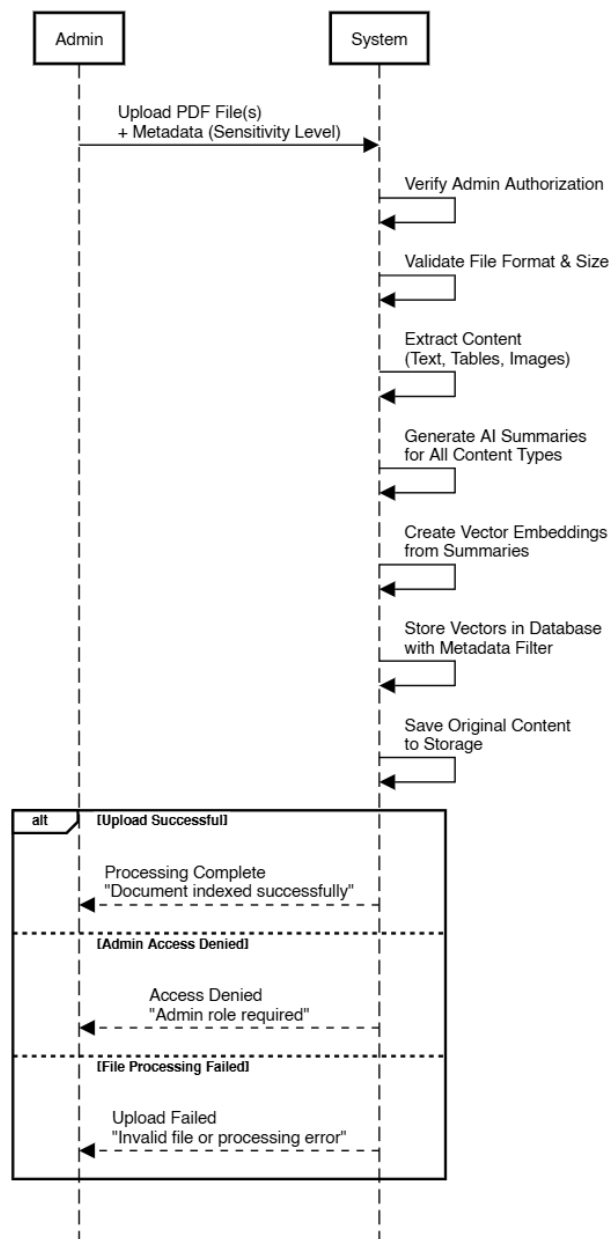
Gambar 3.5 menggambarkan proses autentikasi *login* pengguna dalam sistem. Proses ini melibatkan langkah-langkah validasi kredensial, verifikasi *password*, pembuatan *JWT* token, dan penyimpanan sesi pengguna. Sistem melakukan pengecekan terhadap format *Input*, keberadaan *user* di *database*, serta status akun sebelum memberikan akses. Jika kredensial valid dan akun aktif, sistem akan menghasilkan *JWT* token dan mengembalikan informasi pengguna beserta akses token untuk autentikasi permintaan selanjutnya.



Gambar 3.5 Proses *Login*.

c. Proses *Upload* dan *Indexing* Dokumen PDF

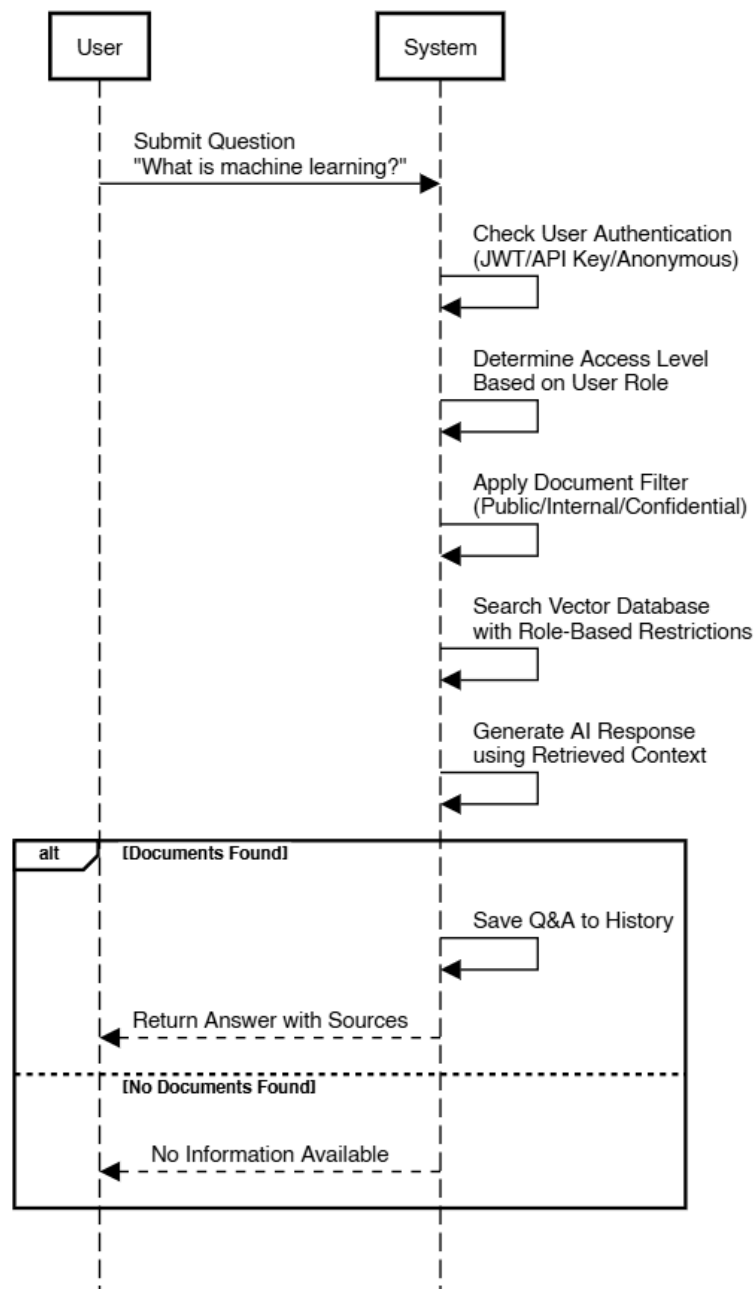
Gambar 3.6 menggambarkan proses *upload* dan *indexing* dokumen PDF dalam sistem. Proses ini hanya dapat dilakukan oleh admin yang telah terautentikasi. Sistem melakukan verifikasi otorisasi admin, validasi format dan ukuran *file*, ekstraksi konten *multi-modal*, pembuatan *summary* menggunakan *AI* untuk semua tipe konten, pembentukan *Vector Embeddings* dari *summary*, penyimpanan vektor ke *database* dengan metadata filter, serta penyimpanan konten asli.



Gambar 3.6 Proses *Upload* dan *Indexing* Dokumen PDF.

d. Proses *Query* RAG

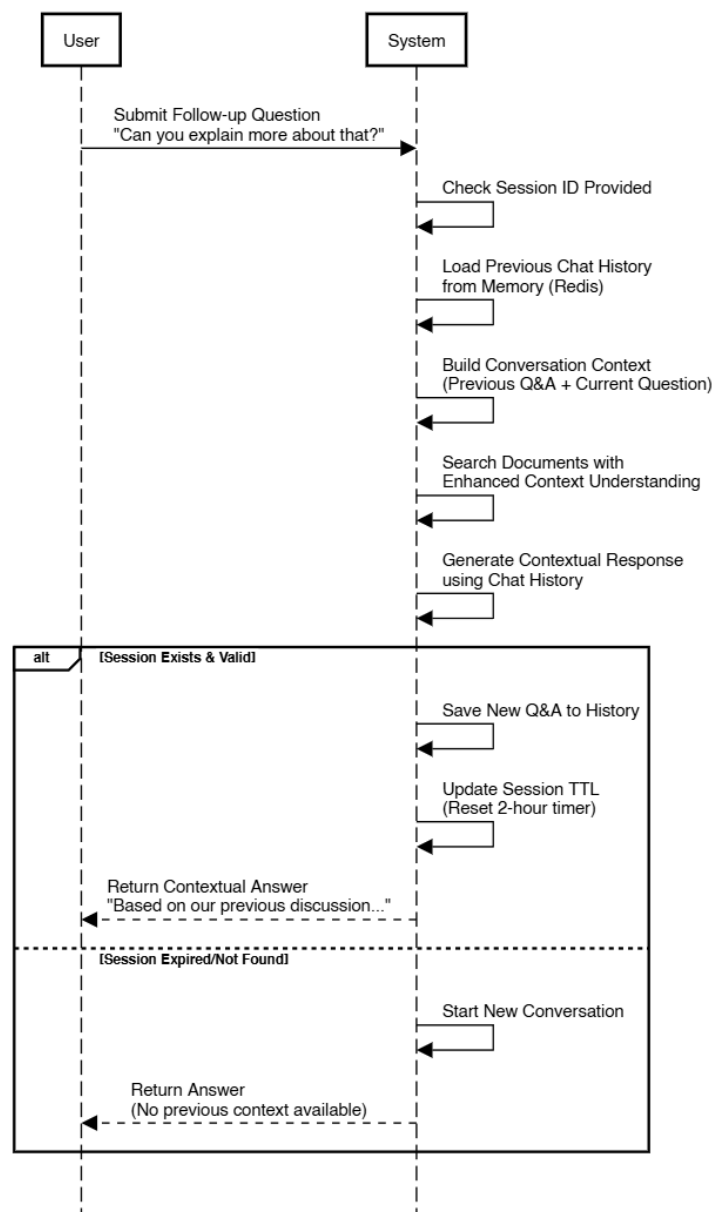
Gambar 3.7 menggambarkan proses *query* menggunakan *RAG* dengan mekanisme *RBAC*. Proses dimulai ketika *user* mengirimkan pertanyaan ke sistem. Sistem melakukan pengecekan autentikasi *user*, menentukan tingkatan akses berdasarkan *role user*, menerapkan filter dokumen sesuai tingkatan sensitivitas, melakukan pencarian di *Vector Database* dengan pembatasan *RBAC*, dan menghasilkan respon *AI* menggunakan konteks yang ditemukan.



Gambar 3.7 Proses *Query* RAG.

e. Proses pecakapan berkelanjutan

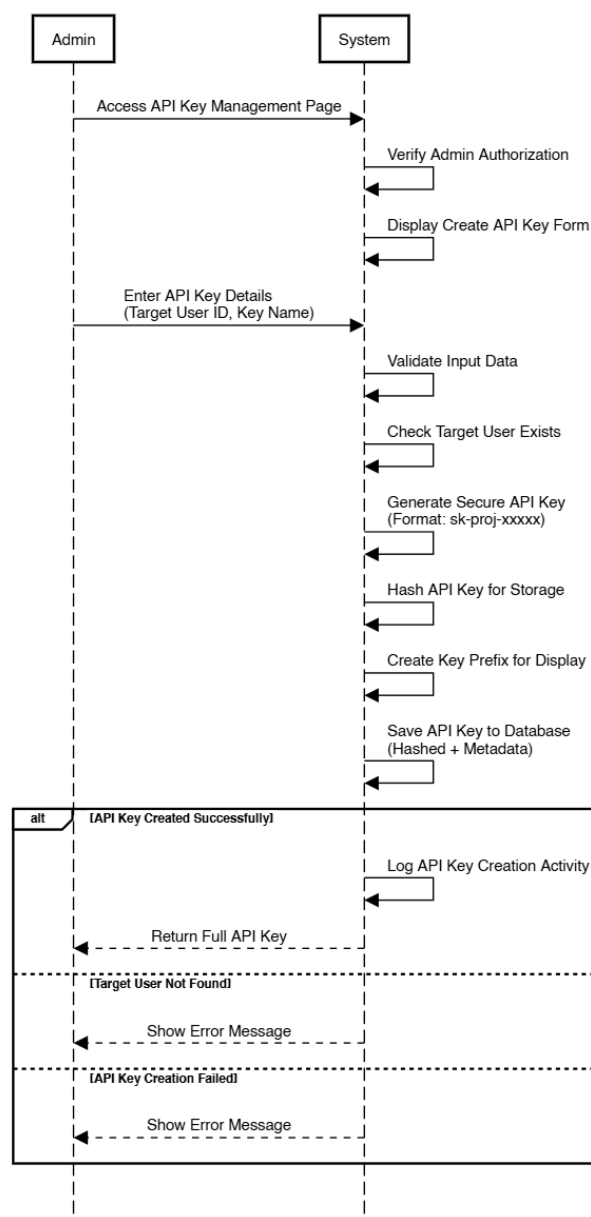
Gambar 3.8 menggambarkan proses melanjutkan percakapan dalam sistem chatbot. Proses dimulai ketika *user* mengirimkan pertanyaan lanjutan dengan menyertakan *session ID* yang sama dari percakapan sebelumnya. Sistem melakukan pengecekan keberadaan *session ID*, memuat riwayat percakapan sebelumnya dari Redis, membangun konteks percakapan dengan menggabungkan riwayat dan pertanyaan saat ini, melakukan pencarian dokumen dengan pemahaman konteks yang lebih baik, dan menghasilkan respon kontekstual menggunakan *Chat history*.



Gambar 3.8 Proses *Continue Conversation*.

f. Proses Create *API Key*

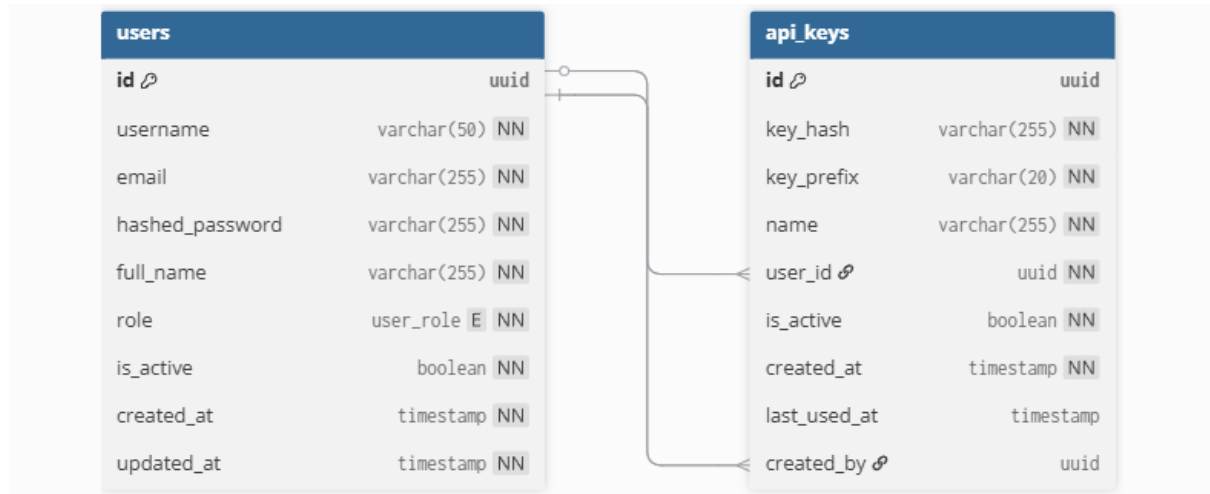
Gambar 3.9 menggambarkan proses pembuatan *API key* untuk pengguna dalam sistem. Proses ini hanya dapat dilakukan oleh admin yang telah terautentikasi. Sistem melakukan verifikasi otorisasi admin, menampilkan *form* pembuatan *API key*, validasi data *Input*, pengecekan keberadaan target *user*, pembuatan *secure API key* dengan format standar, *hashing API key* untuk penyimpanan, pembuatan *key prefix* untuk tampilan, penyimpanan *API key* beserta metadata ke *database*, pencatatan aktivitas pembuatan, dan pengembalian *full API key* yang hanya ditampilkan satu kali.



Gambar 3.9 Proses Create *API Key*.

3.3.4 Perancangan *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) menggambarkan hubungan antara entitas dalam basis data. ERD sistem ini meliputi tabel *users* dan *API key* yang saling berelasi untuk mendukung autentikasi, manajemen pengguna, dan akses programatik melalui *API key*.



Gambar 3.10 *Entity Relationship Diagram*

a. Tabel *User*:

1. **Id (uuid):** *Primary key* bertipe UUID.
2. **username (varchar(50)):** *Username* untuk login.
3. **email (varchar(255)):** *Email address* pengguna.
4. **hashed_password (varchar(255)):** *Password* enkripsi.
5. **full_name (varchar(255)):** Nama lengkap pengguna.
6. **role (user_role):** *Role* pengguna.
7. **is_active (boolean):** Status aktif pengguna dalam sistem.
8. **created_at (timestamp):** Waktu pembuatan akun pengguna.
9. **updated_at (timestamp):** Waktu terakhir data pengguna di *update*.

b. Tabel *API KEY*:

1. **id (uuid):** *Primary key* bertipe UUID.
2. **key_hash (varchar(255)):** *Hash* dari *API key*.
3. **key_prefix (varchar(20)):** *Prefix* untuk *display*.
4. **name (varchar(255)):** Nama deskriptif *API key* untuk identifikasi.
5. **user_id (uuid):** *Foreign key* yang merujuk ke pemilik *API key* di tabel *users*.
6. **is_active (boolean):** Status aktif *API key*.

7. *created_at* (timestamp): Waktu pembuatan *API key*.
8. *last_used_at* (timestamp): Waktu terakhir kali *API key* digunakan.
9. *created_by* (uuid): *Foreign key* yang merujuk ke admin pembuat *API key*

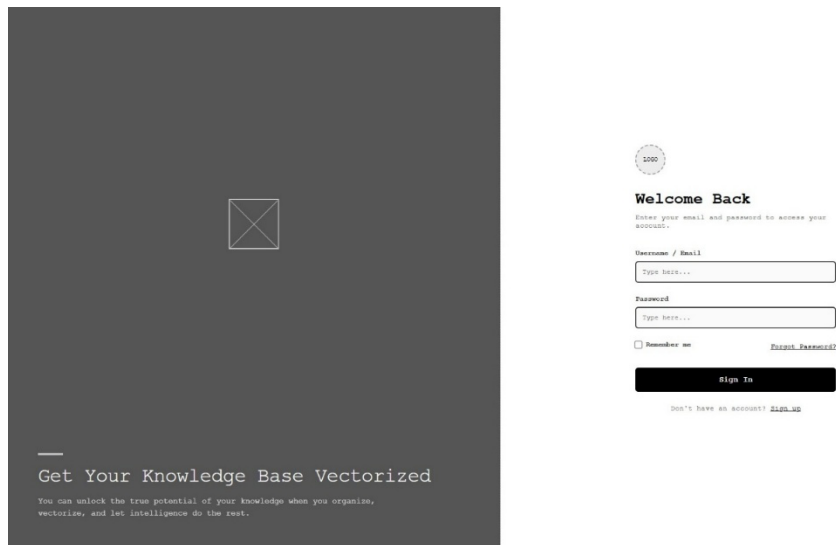
Tabel *Users* memiliki relasi satu ke banyak dengan tabel *API Key*, yang berarti seorang pengguna dapat memiliki banyak *API key*. Relasi ini diimplementasikan dengan kolom *user_id* dalam tabel *API Key*, yang merupakan *foreign key* yang merujuk ke kolom *id* dalam tabel *Users* dengan *constraint delete cascade*. Hal ini memungkinkan sistem melacak siapa pemilik setiap *API key*, dan ketika *user* dihapus, semua *API key* yang dimilikinya akan otomatis terhapus untuk menjaga integritas data.

Selain itu, terdapat relasi tambahan melalui kolom *created_by* pada tabel *API Keys* yang juga merujuk ke tabel *Users*. Relasi ini menunjukkan admin mana yang membuat *API key* tersebut, dengan *constraint delete set null* sehingga ketika admin dihapus, *record API key* tetap tersimpan namun kolom *created_by* akan diset menjadi *null*. Relasi antara *Users* dan *API Key* memungkinkan sistem memahami struktur kepemilikan *API key*, melacak siapa yang membuat *API key* untuk keperluan audit, serta menerapkan *role-based access control* dimana hanya admin yang dapat membuat dan mengelola *API key* untuk pengguna lain dalam sistem.

3.3.5 Perancangan Desain Antarmuka Pengguna

a. Perancangan Halaman *Login*

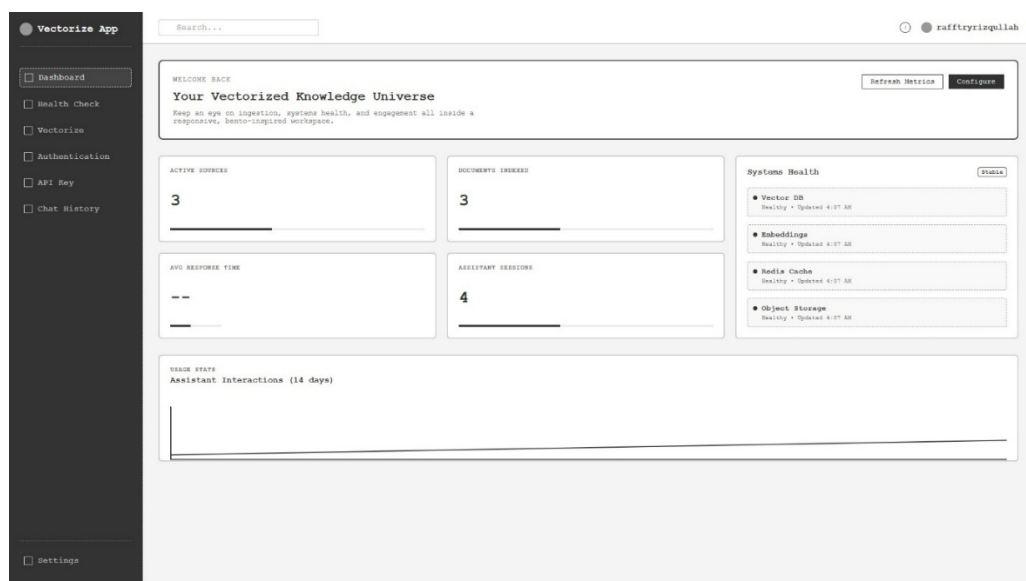
Halaman *Login* dirancang untuk menjadi titik autentikasi yang sederhana dan intuitif. Perancangan ini harus mencakup formulir *Input* yang jelas untuk *username* atau *email* dan *password*. Elemen penting lainnya adalah *checkbox* "*Remember me*" untuk menyimpan sesi, tombol *submit* untuk proses *login*, dan mekanisme tampilan pesan *error* yang informatif jika *login* gagal, serta pesan sukses setelah autentikasi berhasil.



Gambar 3.11 Tampilan *Wireframe* Halaman *Login*

b. Perancangan Halaman *Dashboard* Admin

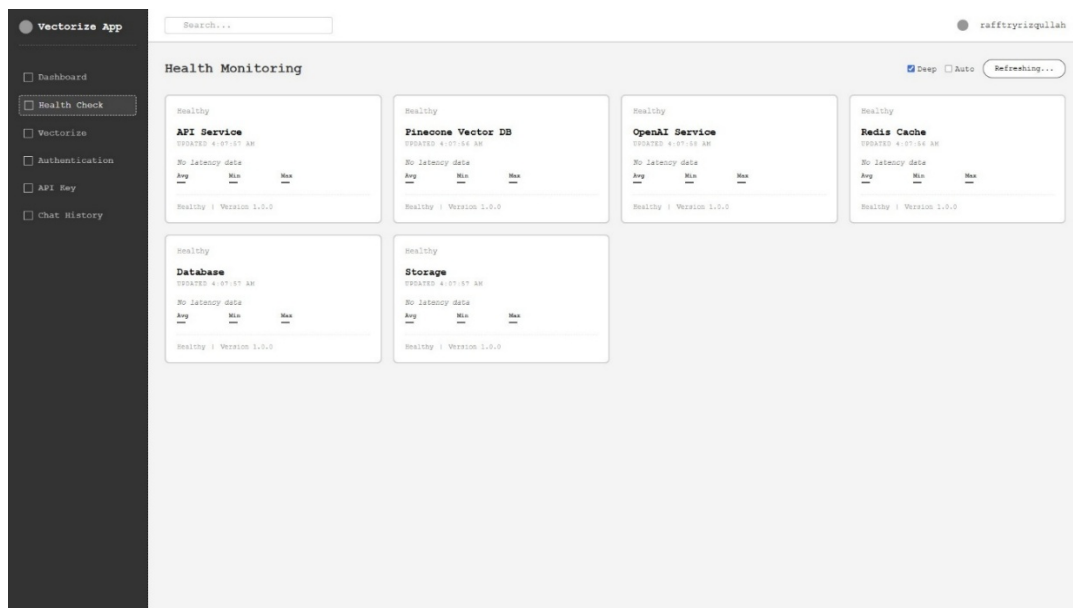
Antarmuka *Dashboard* Admin berfungsi sebagai pusat kendali dan ringkasan bagi Admin. Desainnya harus memuat *Panel* navigasi atau *sidebar* untuk memudahkan akses ke berbagai fitur. Halaman ini menampilkan *Card* atau *widget* untuk statistik sistem secara ringkas dan juga untuk menunjukkan status kesehatan sistem dengan indikator visual. Selain itu, *Dashboard* dilengkapi dengan tabel *recent activities*, *quick access button*, grafik statistik penggunaan, dan *panel alert* atau notifikasi untuk peringatan *system*.



Gambar 3.12 Tampilan *Wireframe* Halaman *Dashboard*

c. Perancangan Halaman *Health Monitoring*

Halaman ini sangat penting untuk memastikan sistem dapat diakses selama 24 Jam dan memiliki mekanisme *error handling* yang efektif. Perancangannya berfokus pada menampilkan status diagnostik sistem, yaitu melalui *card* yang menunjukkan status kesehatan *API* sistem, status Pinecone *Vector Database*, dan status layanan OpenAI. Desain ini harus menyertakan indikator visual, timestamp untuk waktu *monitoring* terakhir, tombol *refresh* manual, dan menampilkan pesan *error* atau *alert* jika ada layanan yang *down*.



Gambar 3.13 Tampilan *Wireframe* Halaman *Health Monitoring*

d. Perancangan Halaman *Upload Dokumen*

Antarmuka ini didedikasikan untuk Manajemen Dokumen Basis Pengetahuan. Perancangan harus memfasilitasi *upload file PDF*, mendukung satu atau beberapa *file*, dan menyediakan area *drag-and-drop* untuk kemudahan *Input*. *Admin* juga dapat memasukkan *URL* referensi dan *metadata* seperti kategori dokumen dan tingkat sensitivitas. Setelah *submit*, antarmuka harus menunjukkan indikator *loading* dan daftar file yang di *upload* dengan status pemrosesannya lengkap dengan pesan konfirmasi yang mencantumkan jumlah *chunks* yang di *generate*.

Vectorize App Search... raffryrizqillah

Dashboard
Health Check
Vectorize
Authentication
API Key
Chat History

Upload Documents (PDF)
Upload one or more PDFs to process and index.

Choose files or drag and drop
PDF only, reasonably sized for the server.

Source links (optional)
Add files to show per-file source link inputs.
Field count automatically matches the number of files.

Author
Author name

Department
Unit / Division

Sensitivity
public internal

Category
☒ System & Service Documentation
☐ Security & Networking
☐ Reports & Audits
☐ IT Procedures & Policies
☐ Training & Tutorials

Upload 0 files selected

Indexed Documents

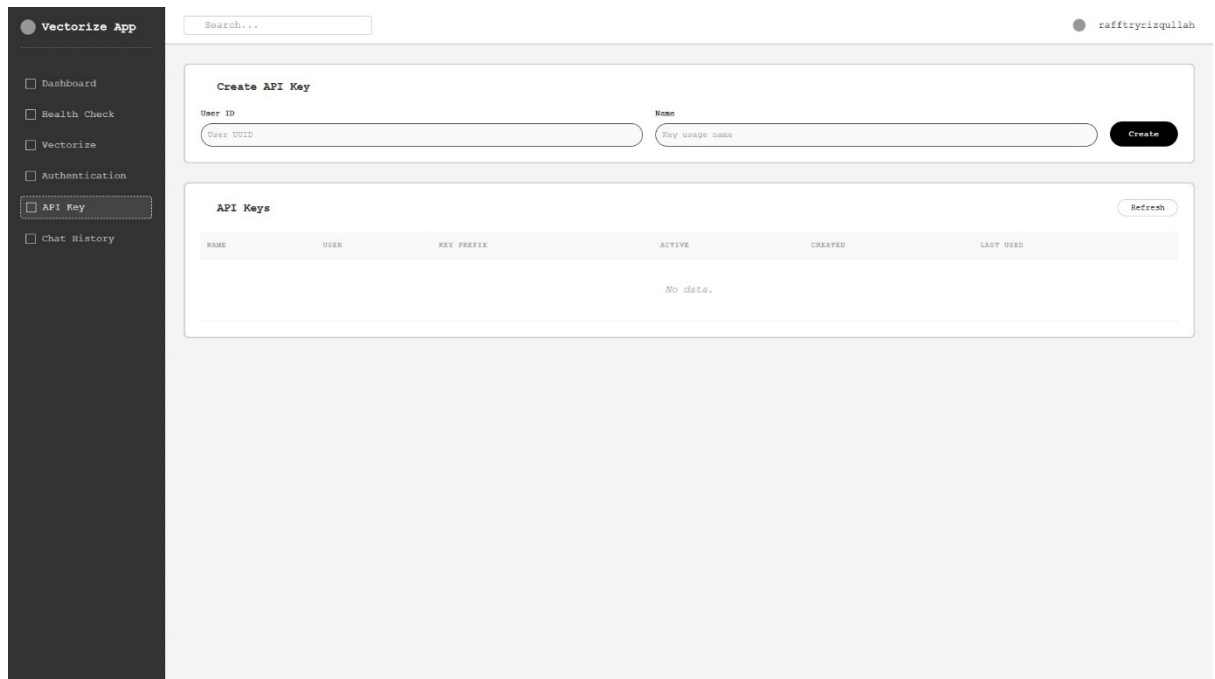
Filter Refresh

DOCUMENT	AUTHOR	SOURCE LINKS	STATUS	DEPLOYED AT	SENSITIVITY
Install EAP-TLS certificate di Android OS	raffry rizqillah	Link 1	COMPLETED	2025-10-14 06:43:14	public
Panduan Backup Data Storage Google UII	raffry rizqillah	Link 1	COMPLETED	2025-10-14 10:57:22	internal

Gambar 3.14 Tampilan *Wireframe* Halaman *Upload Document*

e. Perancangan Halaman Pengelolaan *API Key*

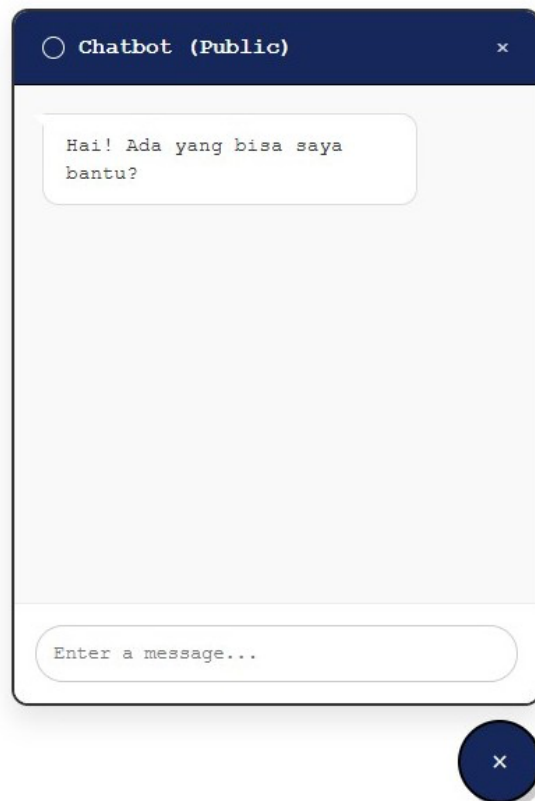
Antarmuka ini dikelola oleh Admin untuk membuat dan mengelola *API Key* guna mendukung autentikasi untuk akses programmatik. Desainnya harus menyediakan *form Input* untuk membuat *API key* baru dengan *field User ID* dan deskripsi. Fitur utama termasuk tabel untuk menampilkan semua *API key* yang aktif, tombol untuk menonaktifkan *API key* yang dipilih, serta filter atau mencari berdasarkan pengguna tertentu.



Gambar 3.15 Halaman Pengelolaan *API Key*

f. Perancangan Halaman *Chatbot Interface*

Ini adalah antarmuka interaksi utama bagi pengguna, di mana perancangan harus memastikan pengalaman percakapan yang lancar. Desain ini mencakup area tampilan percakapan dan form *Input* kolom *Chat* untuk mengetikkan pertanyaan. Untuk membedakan sumber pesan, serta menyediakan tombol *login* untuk autentikasi pengguna



Gambar 3.16 Tampilan *Wireframe* Halaman *Chatbot Interface*

3.4 Implementasi Sistem

Implementasi sistem merupakan tahap realisasi teknis dari rancangan arsitektur yang telah didefinisikan sebelumnya, di mana seluruh komponen perangkat lunak dikembangkan menggunakan pendekatan modular berbasis *Graph-Modular Architecture* untuk menjamin skalabilitas dan efisiensi. Pada tahap ini, spesifikasi desain diterjemahkan ke dalam kode program yang fungsional dengan memanfaatkan kerangka kerja FastAPI sebagai fondasi *backend*, yang mengintegrasikan logika bisnis, mekanisme keamanan, serta orkestrasi layanan eksternal seperti OpenAI, Pinecone, dan Redis guna menghadirkan ekosistem *chatbot IT Support* yang cerdas, aman, dan responsif terhadap kebutuhan pengguna.

3.4.1 Implementasi Autentikasi dan Otorisasi

a. Implementasi Proses *Login* dan *JWT* Generation

Sistem mengimplementasikan autentikasi berbasis *JWT* untuk memastikan keamanan akses pengguna. Proses *login* dimulai ketika pengguna mengirimkan *username* dan *password*.

Sistem melakukan verifikasi *password* menggunakan *bcrypt hashing*, kemudian jika valid, membuat *JWT* token yang berisi informasi *user*. Token ini di *encode* menggunakan algoritma HS256 dengan *secret key* dan memiliki masa berlaku sesuai konfigurasi. *JWT* token yang dihasilkan kemudian dikirim kembali ke *client* dan digunakan untuk autentikasi *request* selanjutnya melalui *Authorization header* dengan format *Bearer* token.

```
async def login_user(db: AsyncSession, username: str, password:
str) -> Optional[Dict[str, Any]]:

    user = await authenticate_user(db, username, password)
    if not user:
        return None

    token_data = {
        "sub": str(user.id),
        "username": user.username,
        "role": user.role.value,
    }

    access_token=create_access_token(data=token_data)
    return {
        "access_token": access_token,
        "token_type": "bearer",
        "user": {...},
    }
```

Gambar 3.17 Implementasi Proses *Login* dan *JWT* Generation

b. Implementasi *Role-Based Access Control*

Sistem *RBAC* diimplementasikan dengan tiga tingkatan *role* Admin,Dosen, dan Mahasiswa yang didefinisikan menggunakan *enum* untuk memastikan *type safety*. Setiap *endpoint* dapat dibatasi aksesnya menggunakan *dependency require role* yang menerima satu atau lebih *role* yang diizinkan. Ketika *user* mengakses *endpoint* yang dilindungi, sistem akan meng ekstrak *JWT* token dari *header*, *decode payload* untuk mendapatkan *role user*, kemudian memvalidasi apakah *role* tersebut termasuk dalam daftar. Jika tidak sesuai, sistem akan mengembalikan *HTTP 403 Forbidden*.

```
def require_role(*allowed_roles: UserRole) -> Callable[[User],
Awaitable[User]]:
```

```

async def role_checker(user:User =
Depends(get_current_active_user)) -> User:
    if user.role not in allowed_roles:
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN,
            detail=f"Access denied. Required role: {'',
'.join(r.value for r in allowed_roles)}",
        )
    return user

return role_checker

```

Gambar 3.18 Implementasi *RBAC*

3.4.2 Implementasi *Pipeline* Vektorisasi Dokumen

a. Implementasi *Upload* dan Ekstraksi Konten

Sistem mengimplementasikan proses *upload* dokumen PDF dengan ekstraksi konten *multi-modal* menggunakan *library unstructured*. Ketika admin *upload* file *PDF* melalui *endpoint*, file divalidasi ukurannya, kemudian di *upload* ke *Cloudflare R2 storage* untuk penyimpanan. Proses ekstraksi menggunakan *partition pdf* dengan *OCR Tesseract* untuk memisahkan konten menjadi tiga tipe teks *chunks*, tabel, dan gambar. Sistem menerapkan *chunking* strtegi dengan parameter yang dapat dikonfigurasi untuk memecah teks panjang menjadi *chunk* optimal. Gambar secara otomatis dikonversi ke format yang didukung *OpenAI Vision API* untuk memastikan kompatibilitas dengan model *multimodal*.

```

def process_pdf_from_bytes(self, file_obj: Union[BytesIO,
BinaryIO], filename: str = "document.pdf") -> ExtractedContent:

    chunks = partition_pdf(
        file=file_obj,
        infer_table_structure=True,
        strategy="hi_res",
        extract_image_block_types=["Image"],
        extract_image_block_to_payload=True,
        chunking_strategy=self.chunking_strategy,
        max_characters=self.max_characters,
        languages=self.ocr_languages,
    )
    texts, tables = self._separate_text_and_tables(chunks)
    images = self._extract_images(chunks)
    return ExtractedContent(texts=texts, tables=tables,
images=images)

```

Gambar 3.19 Implementasi *Upload* dan Ekstraksi Konten

b. Implementasi Pembentukan *Embedding*

Sistem menggunakan metode *add content type* untuk memproses berbagai jenis konten seperti teks, tabel dan gambar. Fungsi ini tidak hanya menyimpan *embedding* standar, tetapi juga memeriksa apakah *Hybrid Search* diaktifkan. Jika aktif, sistem akan membuat *sparse vector* menggunakan *BM25 encoder* dan menyimpannya bersamaan dengan *dense vector* ke Pinecone menggunakan metode *hybrid upsert*. Konten asli tetap disimpan di Redis penyimpanan dokumen untuk pengambilan cepat.

```
def _add_content_type(
    self,
    content_items: List[Union[CompositeElement, Table, str]],
    summaries: List[str],
    document_id: str,
    content_type: str,
    source_link: Optional[str] = None,
    custom_metadata: Optional[Dict[str, Any]] = None,
) -> List[str]:

    # Generate unique IDs
    content_ids = [str(uuid.uuid4()) for _ in content_items]

    # Build metadata & extract keywords (simplified)
    all_metadata = []
    for i, summary in enumerate(summaries):
        metadata = {
            self.id_key: content_ids[i],
            "document_id": document_id,
            "content_type": content_type,
        }
        if source_link: metadata["source_link"] = source_link
        all_metadata.append(metadata)

    # Check hybrid search configuration
    sparse_encoder = self._get_sparse_encoder()

    if sparse_encoder and sparse_encoder.is_fitted:
        # Hybrid upsert: dense + sparse vectors
        self._hybrid_upsert(
            ids=content_ids,
            texts=summaries,
            metadatas=all_metadata,
            sparse_encoder=sparse_encoder,
        )
    else:
        # Standard upsert: dense vectors only
        summary_docs = [
            Document(page_content=s, metadata=m)
            for s, m in zip(summaries, all_metadata)
        ]
```

```

    ]
    self.vectorstore.add_documents(summary_docs)

    # Store original content in Redis docstore
    self.docstore.mset(list(zip(content_ids, content_items)))
    return content_ids

```

Gambar 3.20 Implementasi Pembentukan *Embedding*

3.4.3 Implementasi *Proses Retrieval-Augmented Generation*

a. Implementasi Pencarian Semantik

Implementasi pencarian semantik menggunakan metode *hybrid search*. Berbeda dengan pendekatan manual, sistem memanfaatkan kemampuan *server-side* dari Pinecone untuk menggabungkan skor *dense vector* dan *sparse vector*. Fungsi ini membuat *query vector* dan *sparse vector* secara terpisah, lalu mengirimkannya dalam satu permintaan ke indeks Pinecone untuk mendapatkan hasil yang paling relevan secara efisien.

```

def hybrid_search(
    self,
    query: str,
    k: Optional[int] = None,
    metadata_filter: Optional[Dict[str, Any]] = None,
    # ... args lain ...
) -> List[Any]:

    effective_k = k or settings.rag_top_k
    sparse_encoder = self._get_sparse_encoder()

    # Generate vectors
    dense_vector = self.embeddings.embed_query(query)
    sparse_vector = sparse_encoder.encode_query(query)

    # Execute hybrid query on Pinecone
    results = self._index.query(
        vector=dense_vector,
        sparse_vector=sparse_vector,
        top_k=effective_k,
        include_metadata=True,
        filter=metadata_filter
    )

    # Process results & fetch originals from Redis
    summary_docs = []
    for match in results["matches"]:
        metadata = match["metadata"]
        text = metadata.pop("text", "")

```

```

summary_docs.append(Document(page_content=text,
metadata=metadata))

# Fetch original content from Redis based on IDs
# ... (logika fetch Redis disembunyikan) ...

return retrieved_docs

```

Gambar 3.21 Implementasi Pencarian Semantik

b. Implementasi Metadata *Filtering*

Dalam fungsi *retrieve*, sistem menerapkan metadata *filtering* untuk membatasi akses dokumen sesuai *role* pengguna. Jika hasil pencarian kosong, sistem memanggil fungsi *check authorization restrictions*. Fungsi ini melakukan pengecekan ganda, mencari ulang tanpa filter untuk memastikan apakah dokumen tersebut benar-benar tidak ada, atau ada tetapi pengguna tidak memiliki izin akses, lalu mengembalikan pesan penolakan yang spesifik.

```

@tool(response_format="content_and_artifact")
def retrieve(query: str) -> Tuple[str, List[Any]]:
    # ... (logika search) ...

    # Handle empty results
    if not retrieved_docs:
        logger.warning("No documents retrieved for query")

    # Check authorization restrictions via helper function
    auth_result = _check_authorization_restrictions(
        vectorstore_service=vectorstore_service,
        query=query,
        metadata_filter=metadata_filter,
        metadata_callback=metadata_callback,
    )

    if auth_result:
        return auth_result

    return "No relevant documents found in knowledge
base.", []

```

Gambar 3.22 Implementasi Metadata *Filtering*

c. Implementasi *Generation*

Sistem mengimplementasikan answer generation menggunakan GPT-4o dengan *prompt engineering* yang terstruktur, mencakup *formatting guidelines* untuk menghasilkan jawaban berkualitas tinggi yang berkaitan pada Basis Pengetahuan. *Prompt generation* terdiri dari komponen utama, *System prompt* dengan aturan ketat yang melarang LLM menjawab dari basis

pengetahuan general. Sistem juga menggunakan *LangGraph StateGraph* untuk orkestrasi, dengan *node query* atau respon yang menggunakan *LLM System Prompt* untuk secara cerdas menentukan langkah selanjutnya. *Router* memutuskan apakah harus memanggil *retrieve tool* untuk pertanyaan teknis atau menjawab langsung untuk sapaan, supaya memberikan fleksibilitas percakapan yang lebih natural.

```
def router_node(
    state: RAGState,
    llm: ChatOpenAI,
    retrieve_tool: Any,
) -> dict[str, Any]:

    # 1. Build Conversation Context & System Prompt
    # System prompt berisi instruksi kapan harus retrieve vs jawab langsung
    conversation = [SystemMessage(content=ROUTER_SYSTEM_PROMPT)]

    # Tambahkan history chat jika ada
    if state.get("chat_history"):
        # ... (logika format history disederhanakan untuk ringkas)
        ...

    conversation.append(SystemMessage(content=formatted_history))

    # Tambahkan pesan user saat ini
    conversation.extend(state["messages"])

    # 2. Bind Retrieve Tool ke LLM
    # Memberi kemampuan pada LLM untuk memanggil fungsi retrieval
    llm_with_tools = llm.bind_tools([retrieve_tool])

    # 3. LLM Membuat Keputusan (Routing)
    # LLM akan mengembalikan tool_calls jika butuh data,
    # atau konten teks biasa jika hanya sapaan.
    response = llm_with_tools.invoke(conversation)

    return {"messages": [response]}
```

Gambar 3.23 Implementasi *Generation*

3.4.4 Chat Interaktif

Chat interaktif merupakan komponen inti yang memungkinkan pengguna berkomunikasi langsung dengan *chatbot IT Support* melalui mekanisme *RAG*. Pada proses ini, setiap pertanyaan pengguna diproses melalui deteksi maksud untuk menentukan apakah pertanyaan termasuk sapaan, permintaan teknis, atau di luar cakupan layanan. Pertanyaan teknis kemudian

diubah menjadi *Embedding* dan dilakukan pencarian semantik pada Pinecone *Vector Database* untuk memperoleh konteks yang paling relevan dari Basis Pengetahuan BSI UII. Informasi tersebut digabungkan dengan *prompting engineering* sehingga model GPT hanya memberikan jawaban berdasarkan sumber resmi dan tidak menggunakan pengetahuan di luar konteks.

3.4.5 Details Proses Prompting

Proses *prompting* merupakan bagian penting dalam implementasi *chatbot IT Support*, karena menentukan bagaimana model bahasa merespon pertanyaan pengguna berdasarkan konteks yang diperoleh melalui mekanisme *RAG*. Sistem ini menerapkan *prompt engineering* yang terstruktur untuk memastikan bahwa model hanya memberikan jawaban yang sesuai dengan Basis Pengetahuan BSI UII. Proses *prompting* terdiri dari tiga lapisan utama, yaitu *system prompt*, *intent classification prompt*, dan *respon generation prompt*, yang masing-masing dirancang untuk mengendalikan perilaku model secara presisi.

System prompt digunakan untuk menetapkan batasan ketat ruang lingkup informasi yang boleh digunakan oleh model. Sistem memastikan bahwa model hanya menjawab berdasarkan dokumen yang diambil dari *Vector Database* melalui *RAG*, dan dilarang memberikan jawaban menggunakan pengetahuan umum. Pendekatan ini menghindari halusinasi serta memastikan konsistensi dengan kebijakan *IT Support* BSI UII.

```
Kamu adalah asisten IT Support BSI UII. Jawab HANYA berdasarkan
konteks knowledge base.

=== ATURAN KETAT ===
1. HANYA jawab berdasarkan konteks yang diberikan
2. DILARANG KERAS menggunakan pengetahuan umum di luar konteks
3. DILARANG menjawab pertanyaan non-IT (presiden, film, olahraga,
dll)
4. Jika informasi tidak ada → jawab:
    "Maaf, informasi tersebut tidak tersedia di knowledge base."

=== FORMAT JAWABAN ===
- Bahasa Indonesia, jelas dan ringkas
- Gunakan numbered list jika berisi langkah-langkah
- Sertakan nada ramah dengan emoji seperlunya 😊

Konteks:
{context}
```

Gambar 3.24 *System prompt*

- a. *Intent classification prompt* untuk menentukan kategori pertanyaan pengguna apakah berupa sapaan, pertanyaan teknis, atau di luar lingkup layanan. *Prompting* ini membantu *chatbot* memutuskan apakah perlu menjalankan proses *retrieval* atau hanya memberikan respon ringan, seperti menyapa atau menolak dengan sopan.

```
Kamu adalah asisten IT Support BSI UII.

=== SCOPE YANG DIIZINKAN ===
- Login, SSO, reset password
- Email, Microsoft 365, Google Workspace
- WiFi kampus, VPN
- Sistem akademik (SISKA, e-learning)
- Perangkat kampus (laptop, printer, proyektor)

=== PERTANYAAN DI LUAR SCOPE ===
Tolak dengan ramah dan arahkan kembali ke masalah IT.

=== CARA ROUTING ===
1. Sapaan → Balas ramah
2. Terima kasih → Balas singkat
3. Pertanyaan IT → Lakukan retrieval
4. Pertanyaan non-IT → Tolak ramah
```

Gambar 3.25 *Intent classification*

- b. *Respon generation prompt* digunakan untuk menggabungkan konteks hasil *retrieval* dan riwayat percakapan, sehingga model mampu menghasilkan respon yang koheren dalam percakapan *multi-turn*. Mekanisme ini memastikan *chatbot* dapat memahami referensi lanjutan seperti “itu”, “yang tadi”, atau “sebelumnya”.

```
Gunakan konteks berikut untuk menjawab pertanyaan pengguna.
Jika jawaban tidak ditemukan, jangan mengarang.

{history}
{context}

Jawaban:
```

Gambar 3.26 *Respon generation prompt*

Dengan penerapan struktur *prompt* yang ketat dan terarah ini, *chatbot* mampu menghasilkan respon yang relevan, akurat, dan sesuai standar layanan *IT Support BSI UII*. Pendekatan proses

prompting ini menggabungkan praktik terbaik *prompting engineering modern* sebagaimana dipaparkan dalam penelitian terdahulu (Banghao dkk., 2024), (Sahoo dkk., 2025), serta mengikuti pola implementasi yang telah terbukti pada sistem chatbot berbasis AI.

3.5 Pengembangan Sistem

Pada tahap ini, kebutuhan yang telah dianalisis pada Subbab 3.4 dikembangkan menjadi desain sistem yang komprehensif. Perancangan ini mencakup empat komponen teknis utama yaitu desain struktur data untuk organisasi penyimpanan yang efisien, *ERD* untuk memodelkan relasi antar entitas, arsitektur sistem berbasis *Representational State Transfer Application Programming Interface (REST API)* yang menjamin keamanan komunikasi melalui *JWT*, serta desain *endpoint* untuk interaksi sistem. Seluruh proses perancangan ini dilaksanakan melalui tiga iterasi pengembangan, di mana setiap iterasi memberikan ruang untuk penyesuaian, penyempurnaan, dan evaluasi bertahap guna memastikan hasil akhir sistem selaras dengan spesifikasi kebutuhan dan tujuan penelitian.

3.5.1 Iterasi I

Iterasi pertama berfokus pada pembangunan fondasi sistem *Minimum Viable Product (MVP)* untuk memvalidasi konsep utama *RAG*. Pengembangan mencakup implementasi arsitektur dasar, sistem autentikasi berbasis *JWT*, mekanisme *upload* dokumen *PDF* tunggal, dan fungsi tanya jawab sederhana. Tahap ini bertujuan memastikan integrasi yang stabil antara modul *backend*, *vector database*, dan *LLM* sebelum pengembangan fitur lanjutan dilakukan.

a. Desain Struktur Tabel

Desain basis data pada iterasi ini menggunakan *PostgreSQL* dengan tabel tunggal *User* sebagaimana dirincikan pada Tabel 3.4. Struktur ini menerapkan integritas data melalui penggunaan *UUID* sebagai *primary key* untuk skalabilitas, batasan unik pada kredensial, serta mekanisme penghapusan lunak melalui kolom status aktif. Keamanan kata sandi dijamin menggunakan enkripsi *bcrypt* yang diproses pada lapisan aplikasi sebelum penyimpanan untuk melindungi kerahasiaan data pengguna.

Tabel 3.4 Desain Struktur Tabel User Iterasi I

No	Nama Field	Tipe Data	Keterangan
1	<i>id</i>	<i>UUID</i>	<i>Primary key</i> , identifikasi unik pengguna.
2	<i>username</i>	<i>VARCHAR(50)</i>	<i>Username</i> untuk <i>login</i> , bersifat unik.
3	<i>email</i>	<i>VARCHAR(255)</i>	<i>Email</i> pengguna, bersifat unik.
4	<i>hashed_password</i>	<i>VARCHAR(255)</i>	<i>Password</i> yang telah di <i>hash</i> menggunakan <i>bcrypt</i> .
5	<i>full_name</i>	<i>VARCHAR(255)</i>	Nama lengkap pengguna.
6	<i>is_active</i>	<i>BOOLEAN</i>	Status aktif pengguna.
7	<i>created_at</i>	<i>TIMESTAMP</i>	Waktu pembuatan.
8	<i>updated_at</i>	<i>TIMESTAMP</i>	Waktu terakhir <i>update</i> .

b. Desain *ERD*

ERD pada iterasi pertama Gambar 3.27 menggambarkan struktur basis data tunggal yang berfokus pada manajemen entitas *User*. Desain ini menjadi fondasi autentikasi sebelum relasi kompleks dikembangkan pada iterasi berikutnya.

Gambar 3.27 *ERD* Iterasi I

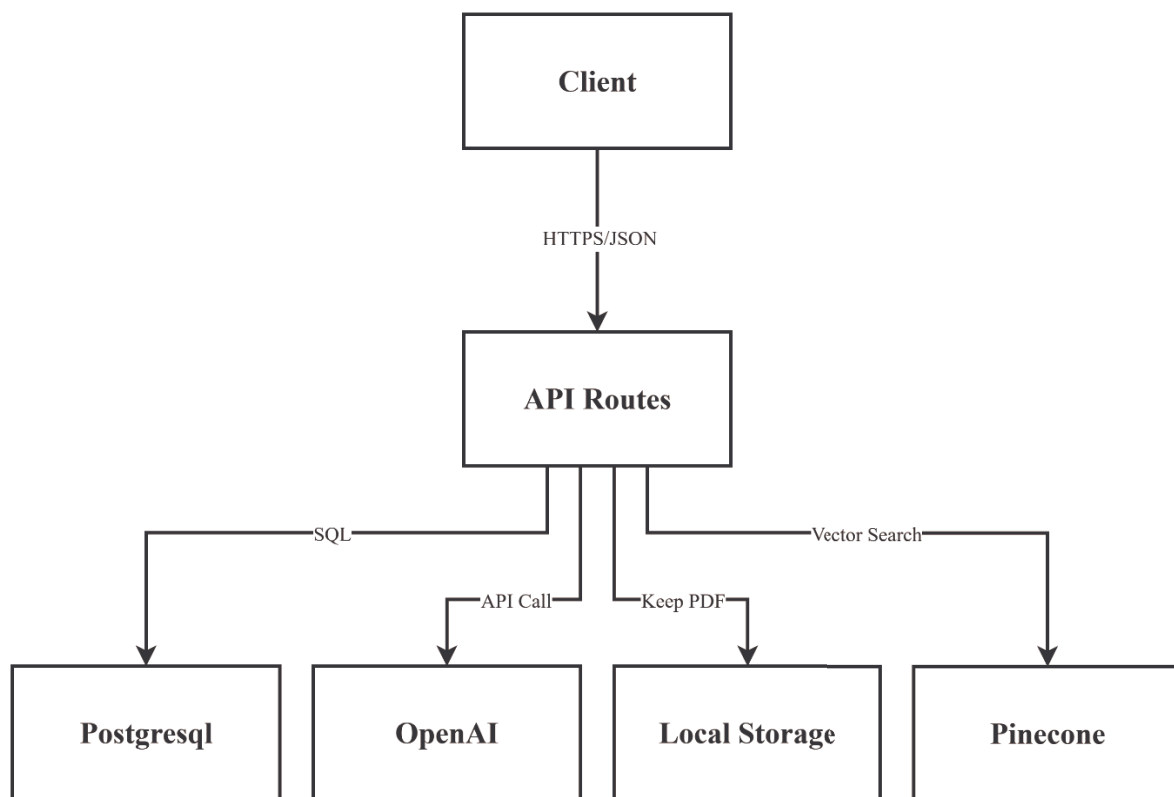
ERD pada Gambar 3.27 menampilkan entitas *User* beserta atribut-atributnya. Pada iterasi pertama ini, tidak terdapat relasi dengan entitas lain karena fokus pengembangan diarahkan

pada pembangunan fondasi autentikasi dan manajemen pengguna. *Primary key* ditandai dengan *id* bertipe *UUID* yang menjamin keunikan setiap *record* pengguna dalam sistem.

Meskipun sederhana, desain *ERD* ini telah mempertimbangkan aspek skalabilitas untuk iterasi selanjutnya. Penggunaan *UUID* sebagai *primary key* memudahkan integrasi dengan sistem eksternal dan mendukung *distributed system architecture*. Struktur ini juga dirancang untuk mendukung penambahan atribut dan relasi baru tanpa memerlukan perubahan signifikan pada skema yang sudah ada.

c. Desain Arsitektur Sistem

Sistem mengadopsi arsitektur *three-tier* yang menghubungkan klien berbasis web dengan *backend* FastAPI berkinerja tinggi Gambar 3.28. Arsitektur ini memisahkan lapisan presentasi, logika bisnis, dan data secara tegas. Lapisan data mengintegrasikan PostgreSQL untuk manajemen pengguna, *Local Storage* untuk penyimpanan objek, untuk penyimpanan dokumen *PDF*, Pinecone untuk penyimpanan *vector* semantik, serta layanan OpenAI untuk pemrosesan *LLM*, menciptakan alur komunikasi yang efisien mulai dari autentikasi, pemrosesan dokumen, hingga respon *chat*.



Gambar 3.28 Desain Arsitektur Sistem Iterasi I

d. Desain Endpoint *API*

Implementasi *API* mencakup enam *endpoint* esensial Tabel 3.5 yang melayani fungsi autentikasi, operasional *RAG*, dan pemantauan sistem. Seluruh pertukaran data menggunakan format *JSON* standar dengan penanganan status *HTTP* yang konsisten, di mana akses ke fitur utama dilindungi oleh mekanisme validasi token *JWT* pada *header* otorisasi.

Tabel 3.5 Daftar *Endpoint* Iterasi I

No	Kebutuhan	Endpoint	Metode	Autentikasi	Deskripsi
1	Autentikasi	<i>/API/v1/auth/login</i>	<i>POST</i>	Tidak	<i>Endpoint</i> untuk proses <i>login</i> pengguna. Menerima <i>username</i> dan <i>password</i> , mengembalikan <i>JWT</i> akses token jika kredensial valid.
2	Autentikasi	<i>/API/v1/auth/register</i>	<i>POST</i>	Tidak	<i>Endpoint</i> untuk registrasi pengguna baru. Menerima data lengkap pengguna dan membuat akun baru di <i>database</i> .

3	Autentikasi	<i>/API/v1/auth/me</i>	<i>GET</i>	Ya	<i>Endpoint</i> untuk mendapatkan informasi pengguna yang sedang <i>login</i> berdasarkan <i>JWT</i> token yang dikirimkan.
4	<i>Query RAG</i>	<i>/API/v1/query</i>	<i>POST</i>	Ya	<i>Endpoint</i> utama untuk mengirimkan pertanyaan ke <i>chatbot</i> . Melakukan <i>retrieval</i> dari <i>Vector Database</i> dan generation menggunakan <i>LLM</i> .
5	Dokumen	<i>/API/v1/documents/upload</i>	<i>POST</i>	Ya	<i>Endpoint</i> untuk <i>upload</i> dokumen PDF secara <i>synchronous</i> . Melakukan ekstraksi teks dan <i>indexing</i> ke <i>Vector Database</i>
6	Monitoring	<i>/health</i>	<i>GET</i>	Tidak	<i>Endpoint health check</i> sederhana untuk mengecek status <i>API</i> . Mengembalikan response 200 OK jika <i>API</i> berjalan normal.

e. Evaluasi Iterasi Pertama

Iterasi pertama sistem Chatbot IT Support BSI UII telah berhasil diimplementasikan dan didemonstrasikan kepada tim IT Support BSI UII pada hari Jumat, 07 Februari 2025. Sistem

prototipe awal berhasil menunjukkan konsep dasar RAG yang mengintegrasikan OpenAI GPT-4o untuk pembangkitan jawaban dan *text-embedding-3-large* untuk vektorisasi dokumen. Fungsionalitas fundamental yang telah diimplementasikan mencakup autentikasi *JWT*, manajemen pengguna dengan satu tabel *database User*, proses *upload* dan *indexing* dokumen *PDF* ke Pinecone, serta *endpoint query RAG* dengan total 6 *endpoint API*. *Frontend* menyediakan *Landing Page* dengan antarmuka *chatbot* dan halaman *Login*. Arsitektur sistem dirancang dengan pendekatan *three-tier* yang memisahkan *Client*, *Backend*, serta lapisan data. Dari hasil demo, beberapa kebutuhan tambahan diidentifikasi untuk iterasi berikutnya, yaitu implementasi *RBAC* untuk membedakan hak akses pengguna, sistem *caching* menggunakan Redis untuk meningkatkan performa, penyimpanan riwayat percakapan untuk konteks percakapan, serta manajemen *API Key* untuk integrasi eksternal. Umpan balik ini menjadi dasar perencanaan iterasi kedua guna meningkatkan fungsionalitas dan pengalaman pengguna sistem.

3.5.2 Iterasi II

Iterasi kedua memperluas kapabilitas sistem dengan fokus pada keamanan dan manajemen sesi melalui implementasi *RBAC* dan penyimpanan riwayat percakapan. Pengembangan ini mencakup penambahan tabel *API Key* untuk akses programatik, perluasan skema pengguna, serta integrasi teknologi *caching* guna mendukung fungsi *monitoring* kesehatan sistem, manajemen pengguna, dan interaksi *chat*.

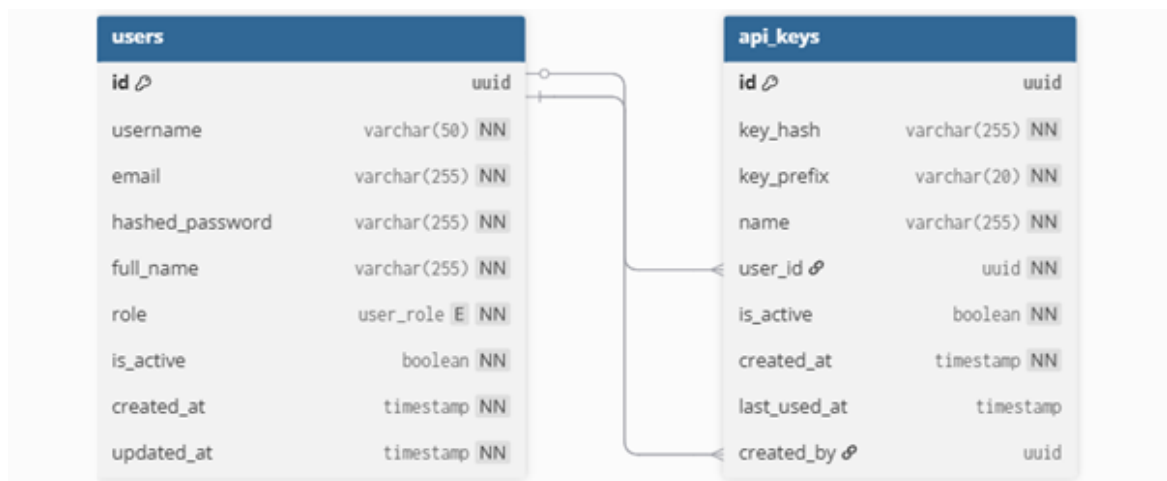
a. Desain Struktur Tabel

Struktur basis data dikembangkan dengan menambahkan tabel *API Key* dan atribut *role* pada tabel *User* Tabel 3.6 untuk memfasilitasi kontrol akses bertingkat. Hubungan antar entitas dimodelkan dalam *ERD* Gambar 3.29 menggunakan relasi *one-to-many*, di mana satu pengguna dapat memiliki banyak *API Key*.

Tabel 3.6 Desain Struktur Tabel Iterasi II

No	Nama Tabel	Atribut	Tipe Data	Keterangan
1	<i>User</i>	<i>id</i>	<i>UUID</i>	<i>Primary Key</i> , identifikasi unik untuk setiap pengguna
		<i>username</i>	<i>VARCHAR(50)</i>	Nama pengguna untuk <i>login</i> , bersifat unik
		<i>email</i>	<i>VARCHAR(100)</i>	Alamat <i>email</i> pengguna, bersifat unik

		<i>hashed_password</i>	<i>VARCHAR(255)</i>	Password yang telah di-hash menggunakan <i>bcrypt</i>
		<i>full_name</i>	<i>VARCHAR(100)</i>	Nama lengkap pengguna
		<i>role</i>	<i>ENUM</i>	Peran pengguna, admin, dosen, mahasiswa
		<i>is_active</i>	<i>BOOLEAN</i>	Status aktif pengguna
		<i>created_at</i>	<i>TIMESTAMP</i>	Waktu pembuatan
		<i>updated_at</i>	<i>TIMESTAMP</i>	Waktu terakhir diperbarui
2	<i>APIKey</i>	<i>id</i>	<i>UUID</i>	<i>Primary Key</i> , identifikasi unik untuk setiap <i>API key</i>
		<i>key_hash</i>	<i>VARCHAR(255)</i>	Hash dari <i>API key</i> yang digunakan untuk validasi
		<i>key_prefix</i>	<i>VARCHAR(10)</i>	<i>Prefix API key</i> untuk identifikasi
		<i>name</i>	<i>VARCHAR(100)</i>	Nama deskriptif dari <i>API key</i>
		<i>user_id</i>	<i>UUID</i>	<i>Foreign Key</i> ke tabel User, pemilik <i>API key</i>
		<i>is_active</i>	<i>BOOLEAN</i>	Status aktif <i>API key</i>
		<i>created_at</i>	<i>TIMESTAMP</i>	Waktu pembuatan <i>API key</i>
		<i>last_used_at</i>	<i>TIMESTAMP</i>	Waktu terakhir <i>API key</i> digunakan

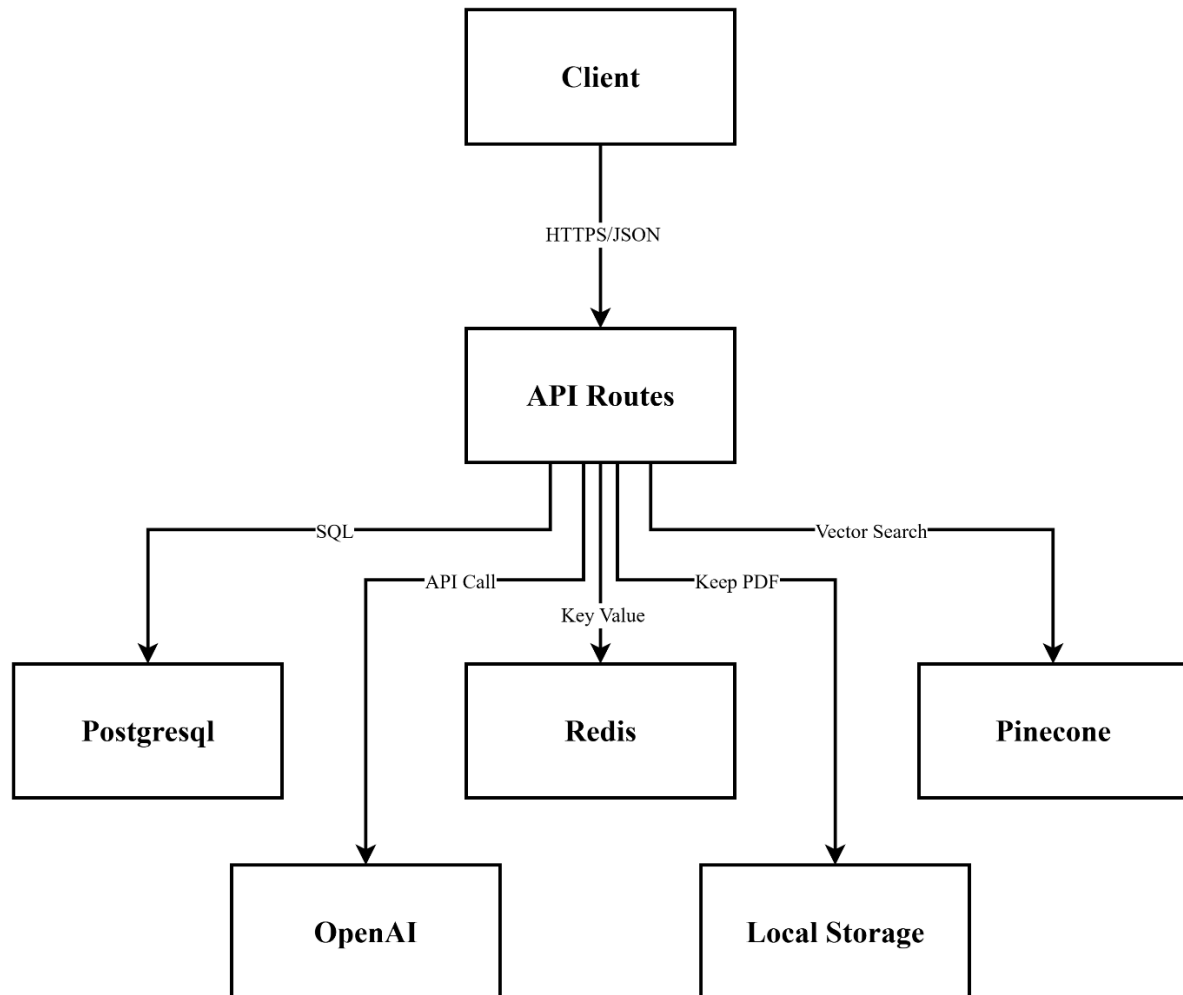


Gambar 3.29 ERD Iterasi I

b. Desain Arsitektur Sistem

Arsitektur sistem ditingkatkan dengan integrasi Redis sebagai *in memory data store* yang berfungsi menangani *caching* dan manajemen sesi percakapan secara *real-time* Gambar 3.30.

Pada lapisan *backend*, struktur modular diperluas melalui pemisahan logika bisnis menjadi layanan-layanan spesifik yang terhubung ke PostgreSQL, Redis, Pinecone, dan OpenAI secara terpusat melalui lapisan layanan.



Gambar 3.30 Desain Arsitektur Sistem Iterasi II

c. Desain Endpoint *API*

Fungsionalitas *API* diperluas mencakup 17 endpoint Tabel 3.7 yang dikelompokkan ke dalam empat kategori utama yaitu *Health Check* untuk pemantauan konektivitas layanan eksternal, *Chat Managemen* untuk pengelolaan sesi dan riwayat percakapan, *User Managemen* untuk administrasi pengguna, serta *Dokumen Managemen* untuk pengawasan indeks dokumen. Seluruh endpoint administratif kini dilindungi oleh mekanisme *RBAC*.

Tabel 3.7 Daftar *Endpoint* Iterasi II

No	Kategori	Endpoint	Metode	Deskripsi	Role Access
----	----------	----------	--------	-----------	-------------

1	Autentikasi	<i>/API/v1/auth/login</i>	<i>POST</i>	Login pengguna menggunakan <i>username</i> dan <i>password</i> , mengembalikan JWT token	PUBLIC
2	Autentikasi	<i>/API/v1/auth/register</i>	<i>POST</i>	Registrasi pengguna baru dengan <i>role</i> tertentu	ADMIN
3	Autentikasi	<i>/API/v1/auth/me</i>	<i>GET</i>	Mengambil informasi pengguna yang sedang <i>login</i>	ADMIN
4	<i>Query</i>	<i>/API/v1/query</i>	<i>POST</i>	Melakukan <i>query RAG</i> ke basis pengetahuan dengan <i>metadata filtering</i> berdasarkan <i>role</i>	ADMIN
5	Dokumen	<i>/API/v1/documents/upload</i>	<i>POST</i>	<i>Upload</i> dokumen <i>PDF</i> dan proses ekstraksi teks untuk <i>indexing</i>	ADMIN
6	<i>Health Check</i>	<i>/health</i>	<i>GET</i>	<i>Health check</i> dasar untuk mengecek status <i>API</i>	Semua
7	<i>Health Check</i>	<i>/API/v1/health/OpenAI</i>	<i>GET</i>	Mengecek konektivitas dan status <i>OpenAI API</i>	ADMIN
8	<i>Health Check</i>	<i>/API/v1/health/pinecone</i>	<i>GET</i>	Mengecek konektivitas dan status <i>Pinecone Vector Database</i>	ADMIN
9	<i>Health Check</i>	<i>/API/v1/health/redis</i>	<i>GET</i>	Mengecek konektivitas dan status <i>Redis cache</i>	ADMIN
10	<i>Health Check</i>	<i>/API/v1/health/postgres</i>	<i>GET</i>	Mengecek konektivitas dan status <i>PostgreSQL database</i>	ADMIN

11	Chat	/API/v1/Chat/sessions	GET	Mengambil daftar semua sesi <i>Chat</i> yang tersimpan di Redis	ADMIN
12	Chat	/API/v1/Chat/history/{session_id}	GET	Mengambil riwayat percakapan lengkap dari sesi tertentu	ADMIN
13	Chat	/API/v1/Chat/history/{session_id}	DELETE	Menghapus riwayat percakapan dari sesi tertentu	ADMIN
14	Admin	/API/v1/admin/users	GET	Mengambil daftar semua pengguna terdaftar di sistem	ADMIN
15	Admin	/API/v1/admin/users/{user_id}	DELETE	Menghapus pengguna tertentu dari sistem	ADMIN
16	Dokumen	/API/v1/documents/list	GET	Mengambil daftar semua dokumen yang telah diindeks dengan metadata	ADMIN
17	Dokumen	/API/v1/documents/{doc_id}	DELETE	Menghapus dokumen tertentu dari <i>Vector Database</i> dan storage	ADMIN

d. Evaluasi Iterasi Kedua

Iterasi kedua sistem *Chatbot IT Support* BSI UII telah diimplementasikan dan didemonstrasikan kepada tim *IT Support* BSI UII pada hari Jumat, 19 September 2025. Sistem pada iterasi ini menunjukkan peningkatan signifikan dengan penambahan 11 *endpoint* baru menjadi total 17 *endpoint API*, implementasi *RBAC* dengan tiga tingkatan peran Admin, Dosen, Mahasiswa, serta penambahan tabel *API Key* dalam database untuk mendukung autentikasi alternatif. Integrasi Redis sebagai lapisan *caching* berhasil meningkatkan performa sistem. Fitur riwayat percakapan telah diimplementasikan untuk menyimpan konteks percakapan pengguna, memungkinkan sistem memberikan respon yang lebih kontekstual. *Frontend* berkembang dari 2 halaman menjadi 5 halaman dengan penambahan *Dashboard*, *Health Check*, dan *Vectorize*. Dari hasil demo dan diskusi dengan tim IT Support, beberapa kebutuhan lanjutan diidentifikasi untuk iterasi ketiga, yaitu implementasi *asynchronous document upload* untuk menangani file berukuran besar, penggunaan *cross-encoder reranking* untuk meningkatkan akurasi *retrieval*, integrasi Lang Graph untuk manajemen alur kerja yang lebih

kompleks, serta penambahan halaman manajemen *API Key*, *Chat History*. Umpan balik ini menjadi roadmap pengembangan iterasi ketiga sebagai versi final sistem.

3.5.3 Iterasi III

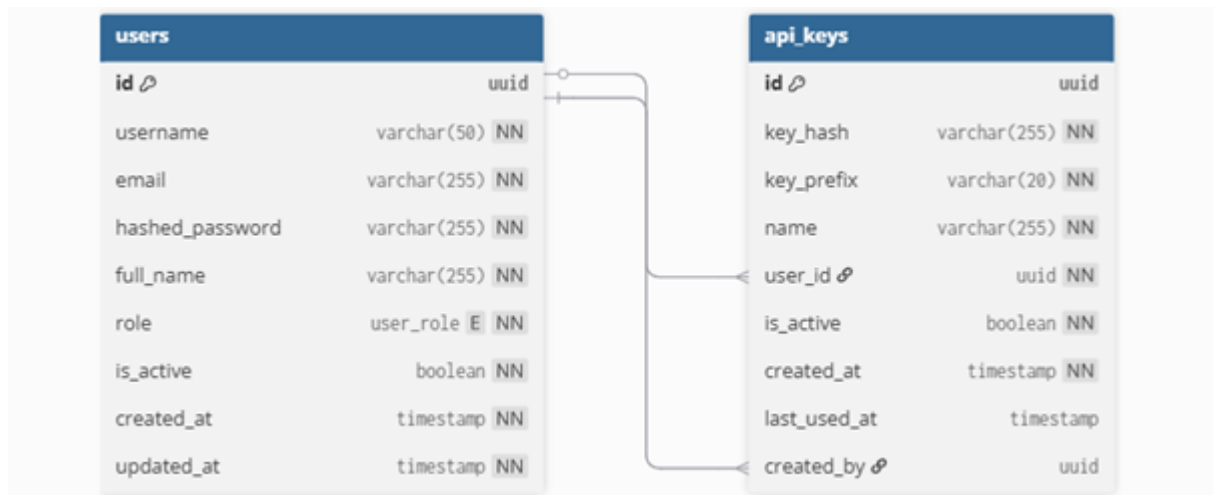
Iterasi ketiga merupakan tahap finalisasi yang mengintegrasikan seluruh komponen menjadi sistem produksi yang utuh. Fokus pengembangan meliputi implementasi fitur tingkat lanjut seperti manajemen *API Key* untuk akses programatik, pemrosesan dokumen asinkron dengan pelacakan progres, serta penerapan alur kerja Lang Graph untuk orkestrasi *RAG*.

a. Desain Struktur Tabel

Struktur basis data disempurnakan Tabel 3.8 dengan penambahan kolom audit *created by* pada tabel *API Key* untuk melacak akuntabilitas administrator. Relasi antar entitas dalam *ERD* Gambar 3.31 diperkuat dengan dua hubungan *one-to-many* antara User dan *API Key* untuk kepemilikan *API key* akses dan jejak audit pembuatnya.

Tabel 3.8 Desain Struktur Tabel Iterasi III

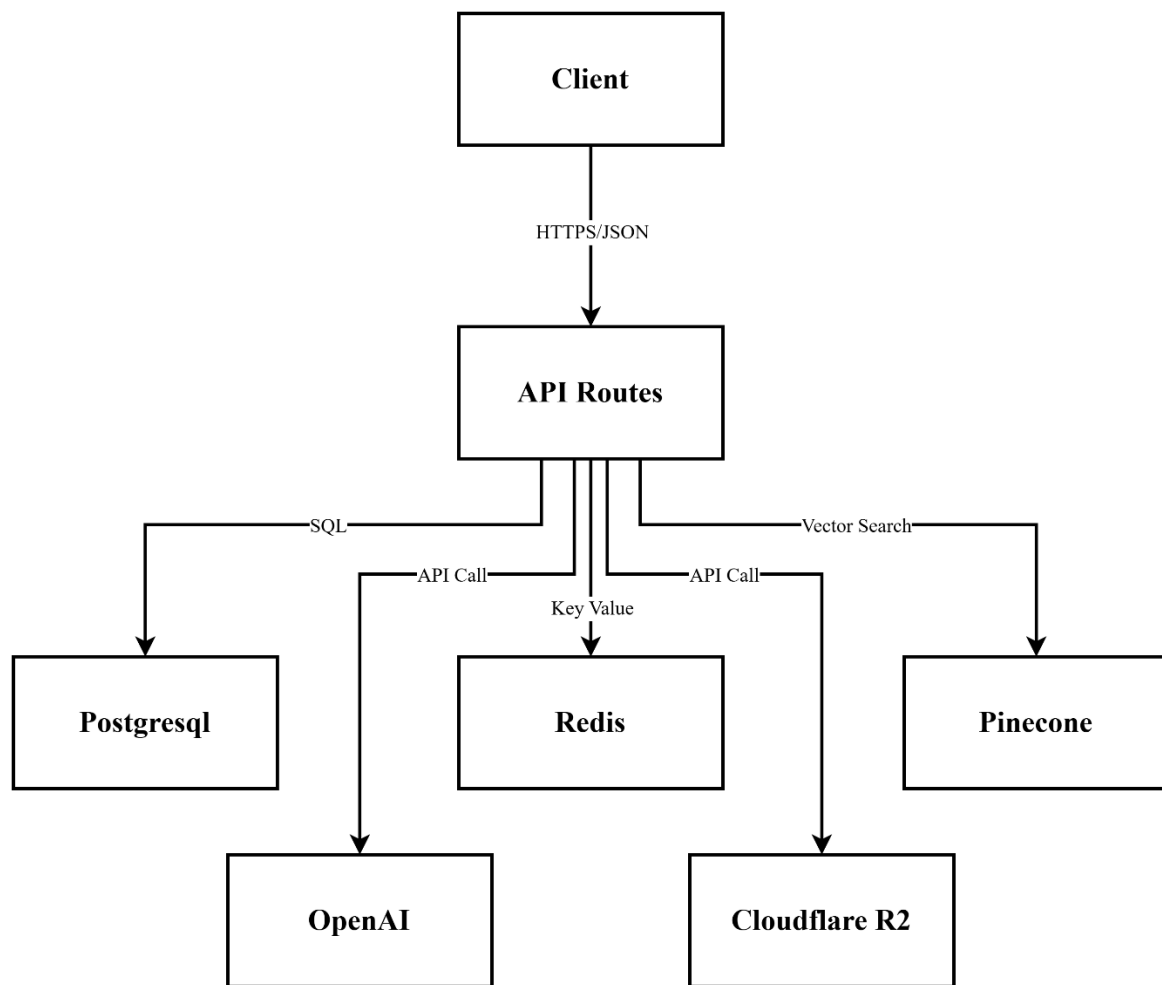
No	Nama Tabel	Atribut	Tipe Data	Keterangan
1	Users	<i>id</i>	<i>UUID</i>	<i>Primary key.</i>
		<i>username</i>	<i>VARCHAR(255)</i>	<i>Username unik pengguna.</i>
		<i>email</i>	<i>VARCHAR(255)</i>	<i>Email unik pengguna.</i>
		<i>hashed_password</i>	<i>VARCHAR(255)</i>	<i>Password terenkripsi dengan bcrypt.</i>
		<i>full_name</i>	<i>VARCHAR(255)</i>	<i>Nama lengkap pengguna.</i>
		<i>role</i>	<i>ENUM</i>	<i>Role pengguna.</i>
		<i>is_active</i>	<i>BOOLEAN</i>	<i>Status aktif pengguna.</i>
		<i>created_at</i>	<i>TIMESTAMP</i>	<i>Waktu pembuatan akun.</i>
		<i>updated_at</i>	<i>TIMESTAMP</i>	<i>Waktu terakhir update.</i>
2	<i>APIKey</i>	<i>id</i>	<i>UUID</i>	<i>Primary key.</i>
		<i>key_hash</i>	<i>VARCHAR(255)</i>	<i>Hash dari API key.</i>
		<i>key_prefix</i>	<i>VARCHAR(16)</i>	<i>Prefix API untuk identifikasi.</i>
		<i>name</i>	<i>VARCHAR(255)</i>	<i>Nama deskriptif API key.</i>
		<i>user_id</i>	<i>UUID</i>	<i>Foreign key ke tabel Users.</i>
		<i>created_by</i>	<i>UUID</i>	<i>Foreign key ke tabel Users.</i>
		<i>is_active</i>	<i>BOOLEAN</i>	<i>Status aktif API key.</i>
		<i>created_at</i>	<i>TIMESTAMP</i>	<i>Waktu pembuatan API key.</i>
		<i>last_used_at</i>	<i>TIMESTAMP</i>	<i>Waktu terakhir digunakan.</i>



Gambar 3.31 ERD Iterasi III

b. Desain Arsitektur Sistem

Arsitektur sistem mencapai bentuk finalnya dengan penerapan Arsitektur berlapis yang komprehensif Gambar 3.32. Lapisan layanan kini mengintegrasikan Cloudflare R2 sebagai penyimpanan objek untuk penyimpanan dokumen *PDF*, melengkapi PostgreSQL, Redis, dan Pinecone. Inti dari pemrosesan cerdas dikelola oleh LangGraph Alur kerja yang mengorkestrasikan klasifikasi maksud, penulisan ulang *query*, dan pembangkitan jawaban.



Gambar 3.32 Desain Arsitektur Sistem Iterasi III

c. Desain *Endpoint API*

API difinalisasi menjadi 25 *endpoint* yang mencakup seluruh spektrum fungsionalitas sistem. Penambahan signifikan pada iterasi ini meliputi *endpoint* asinkron untuk manajemen dokumen skala besar dan *endpoint* administratif untuk pengelolaan kunci *API*. Seluruh *endpoint* dilindungi oleh lapisan keamanan berlapis yang mencakup autentikasi, otorisasi *RBAC*, serta mekanisme *rate limiting*.

Tabel 3.9 Daftar Endpoint Iterasi III

No	Kebutuhan	Endpoint	Metode	Autentikasi	Deskripsi
1	Autentikasi	<i>/API/v1/auth/register</i>	<i>POST</i>	ADMIN	Registrasi pengguna baru dengan <i>role assignment</i> .
2	Autentikasi	<i>/API/v1/auth/login</i>	<i>POST</i>	PUBLIC	Login dan <i>generate JWT</i> akses token.

3	Autentikasi	/API/v1/auth/me	GET	JWT	Mendapatkan informasi <i>user</i> yang sedang <i>login</i> .
4	Query RAG	/API/v1/query	POST	JWT	Query ke basis pengetahuan dengan LangGraph alur kerja.
5	Manajemen Dokumen	/API/v1/documents/upload	POST	ADMIN	Upload dokumen PDF <i>synchronous</i> dengan metadata, ekstraksi teks, <i>summarization</i> , <i>chunking</i> , dan <i>indexing</i> ke Pinecone.
6	Manajemen Dokumen	/API/v1/documents/upload/async	POST	ADMIN	Upload dokumen PDF <i>asynchronous</i> dengan <i>background processing</i> .
7	Manajemen Dokumen	/API/v1/documents/upload/status/{task_id}	GET	ADMIN	Polling status <i>async upload task</i> dengan progress info.
8	Manajemen Dokumen	/API/v1/documents/list	GET	ADMIN	List semua dokumen yang terindex dengan filter dan <i>pagination</i> .
9	Manajemen Dokumen	/API/v1/documents/{doc_id}	DELETE	ADMIN	Hapus dokumen dari Pinecone index.
10	Chat History	/API/v1/Chat/sessions	GET	ADMIN	List semua <i>Chat sessions</i> dengan metadata <i>user</i> dan <i>timestamp</i> .
11	Chat History	/API/v1/Chat/history/{session_id}	GET	ADMIN	Mendapatkan riwayat percakapan dari <i>session</i> tertentu dengan <i>pagination</i> .
12	Chat History	/API/v1/Chat/session/{session_id}	GET	ADMIN	mendapatkan informasi metadata session.
13	Chat History	/API/v1/Chat/history/{session_id}	DELETE	ADMIN	Membersihkan riwayat percakapan dari Redis untuk session tertentu.
14	Manajemen API Key	/API/v1/admin/API-keys	POST	ADMIN	Buat API key baru untuk <i>user</i> tertentu dengan nama dan deskripsi.

15	Manajemen <i>API Key</i>	<i>/API/v1/admin</i> <i>/API-keys</i>	<i>GET</i>	ADMIN	List semua <i>API key</i> dengan filter.
16	Manajemen <i>API Key</i>	<i>/API/v1/admin</i> <i>/API-</i> <i>keys/{key_id}</i>	<i>GET</i>	ADMIN	Mendapatkan detail <i>API key</i> .
17	Manajemen <i>API Key</i>	<i>/API/v1/admin</i> <i>/API-</i> <i>keys/{key_id}</i>	<i>DELETE</i>	ADMIN	Penghapusan Lunak <i>API key</i> .
18	Manajemen <i>API Key</i>	<i>/API/v1/admin</i> <i>/users/{user_</i> <i>id}/API-keys</i>	<i>GET</i>	ADMIN	List semua <i>API key</i> milik <i>user</i> tertentu
19	Manajemen <i>User</i>	<i>/API/v1/admin</i> <i>/users</i>	<i>GET</i>	ADMIN	List semua <i>user</i> dengan filter <i>by role</i> .
20	Manajemen <i>User</i>	<i>/API/v1/admin</i> <i>/users/{user_</i> <i>id}</i>	<i>DELETE</i>	ADMIN	Penghapusan <i>user</i> .
21	<i>Health Check</i>	<i>/health</i>	<i>GET</i>	PUBLIC	Basic health check <i>API</i> .
22	<i>Health Check</i>	<i>/API/v1/healt</i> <i>h/OpenAI</i>	<i>GET</i>	ADMIN	Cek koneksi ke OpenAI <i>API</i> .
23	<i>Health Check</i>	<i>/API/v1/healt</i> <i>h/pinecone</i>	<i>GET</i>	ADMIN	Cek koneksi ke Pinecone.
24	<i>Health Check</i>	<i>/API/v1/healt</i> <i>h/redis</i>	<i>GET</i>	ADMIN	Cek koneksi ke Redis.
25	<i>Health Check</i>	<i>/API/v1/healt</i> <i>h/postgres</i>	<i>GET</i>	ADMIN	Cek koneksi ke PostgreSQL.

d. Evaluasi Iterasi Ketiga

Iterasi ketiga sistem *Chatbot IT Support* BSI UII telah diimplementasikan dan didemonstrasikan kepada tim IT Support BSI UII pada hari Jumat, 17 Oktober 2025. Sistem pada iterasi ini merupakan versi final yang menunjukkan kematangan dengan penambahan 8 *endpoint* baru menjadi total 25 *endpoint API*. Implementasi *asynchronous upload* dokumen dengan status *polling* berhasil menangani file berukuran besar tanpa *blocking* server, sementara *cross-encoder reranking* meningkatkan akurasi relevansi dokumen yang diambil dari *vector database*. Integrasi *Lang Graph* memungkinkan sistem melakukan *intent classification* untuk membedakan sapaan umum dan pertanyaan teknis, sehingga mengoptimalkan penggunaan *retrieval* hanya saat diperlukan. Fitur *API Key* manajemen telah lengkap dengan kemampuan *create*, *list*, *view*, *revoke*, dan jejak audit melalui kolom *created by* yang mereferensi admin

pembuat. Frontend mencapai kelengkapan dengan total 10 halaman termasuk Autentikasi, *API Key*, *Chat History*, *Profile*, dan *Settings*, serta implementasi *drag-and-drop upload* dengan pelacakan progres dan health check. Sistem *database* telah stabil dengan struktur final 2 tabel yaitu *User* dan *APIKey* yang mencakup relasi *one-to-many* dan jejak audit. Berdasarkan evaluasi akhir bersama tim IT Support, sistem dinilai telah memenuhi seluruh kebutuhan fungsional dan non-fungsional yang ditetapkan, siap untuk tahap *deployment* dan pengujian *usability*.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Diagram Proses Bisnis

Sistem *Chatbot IT Support* BSI UII yang dikembangkan mengadopsi arsitektur berbasis *RAG* dengan Lang Graph sebagai lapisan orkestrasi untuk mengelola alur percakapan secara deterministik. Proses bisnis dimulai dari pengguna yang mengirimkan pertanyaan melalui antarmuka, kemudian request diteruskan ke *backend* yang melakukan autentikasi menggunakan *JWT* dan otorisasi berbasis *RBAC*. Setelah validasi berhasil, sistem mengklasifikasikan maksud pengguna menggunakan alur kerja Lang Graph yang membedakan antara sapaan, pemecahan masalah teknis, dan diluar cakupan. Untuk pertanyaan yang memerlukan konteks dokumentasi, sistem melakukan pencarian hybrid pada Pinecone *Vector Database* dengan metadata *filtering* berdasarkan *role* pengguna, kemudian *retrieved* dokumen dikombinasikan dengan riwayat chat dari Redis dan dikirim ke OpenAI untuk menghasilkan respon kontekstual yang akurat. Seluruh riwayat percakapan disimpan dengan *Time-to-Live (TTL)* untuk kepatuhan privasi, sementara *file* lampiran dikelola melalui Cloudflare R2 dengan *URL* untuk keamanan akses. Arsitektur ini memastikan sistem dapat beroperasi 24 jam dengan latensi rendah, skalabilitas tinggi, dan kepatuhan terhadap kebijakan keamanan data.

4.1.1 Diagram Proses Bisnis

Implementasi sistem *Chatbot IT Support* BSI UII mengacu pada BPMN yang telah dirancang pada BAB III. Sebagaimana ditunjukkan pada Gambar 3.2, alur interaksi pengguna dengan sistem dimulai dari autentikasi hingga proses klasifikasi *intent* dan *retrieval* informasi dari basis pengetahuan. Sementara itu, Gambar 3.1 mengilustrasikan proses pengelolaan basis pengetahuan oleh Administrator, mencakup *upload* dokumen, ekstraksi teks, *chunking*, *embedding generation*, hingga penyimpanan ke Pinecone vector database. Kedua diagram tersebut berhasil direalisasikan sepenuhnya dalam sistem versi Iterasi III. Selain itu, implementasi alur kerja internal sistem menggunakan Lang Graph yang mengatur alur pemrosesan *query* secara dinamis, mulai dari klasifikasi maksud, eksekusi pencarian hybrid, konteks *retrieval*, hingga respon *generation* dengan pengamanan anti halusinasi. Diagram alur Lang Graph ini memastikan setiap *query* pengguna diproses secara terstruktur dan konsisten sesuai dengan kebutuhan fungsional yang telah ditetapkan.

4.1.2 Arsitektur Sistem Final

Arsitektur sistem *Chatbot IT Support* BSI UII mencapai bentuk finalnya pada Iterasi ke III dengan penerapan *arsitektur berlapis* yang komprehensif. Arsitektur ini dirancang untuk memisahkan tanggung jawab setiap komponen secara tegas, memastikan skalabilitas, kemudahan pemeliharaan, dan fleksibilitas dalam pengembangan lebih lanjut. Berdasarkan Gambar 3.32 Desain Arsitektur Sistem Iterasi III, arsitektur sistem terdiri dari empat lapisan utama yang saling terintegrasi. Lapisan pertama adalah Lapisan *Client* yang merupakan antarmuka pengguna berbasis web, berkomunikasi dengan *backend* melalui protokol *HTTP*. Lapisan kedua adalah *API Routes* yang dibangun menggunakan kerangka kerja FastAPI, bertanggung jawab untuk menerima dan merutekan seluruh permintaan *HTTP* ke layanan yang sesuai dengan total 25 *endpoint API* yang mencakup autentikasi, manajemen dokumen, *query RAG*, manajemen *API Key*, dan *health monitoring*. Lapisan ketiga adalah Lapisan *Service* yang mengenkapsulasi seluruh logika bisnis sistem, termasuk alur kerja Lang Graph sebagai orkestrasi utama yang mengelola klasifikasi maksud, penulisan ulang kueri, proses *retrieval*, dan pembangkitan respon. Lapisan keempat adalah Lapisan *Data* yang mengintegrasikan berbagai sistem penyimpanan dan layanan eksternal untuk mendukung fungsionalitas sistem secara menyeluruh. Tabel berikut menyajikan komponen teknologi yang terintegrasi dalam arsitektur sistem.

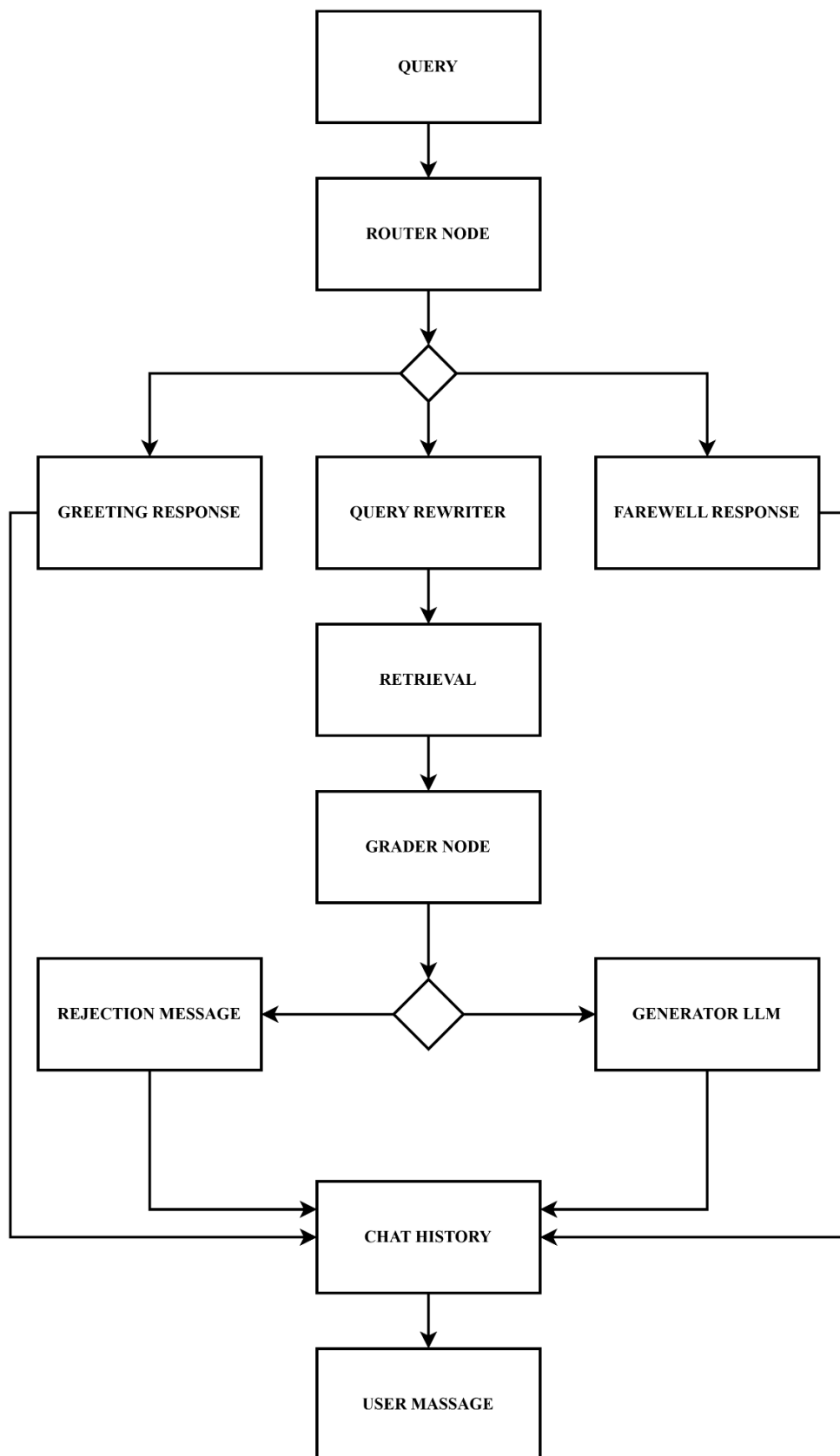
Tabel 4.1 Komponen Teknologi Arsitektur Sistem

Komponen	Teknologi	Fungsi
<i>Backend Framework</i>	FastAPI	<i>REST API</i> dengan dukungan <i>async</i> dan dokumentasi otomatis.
<i>Relational Database</i>	PostgreSQL	Penyimpanan data pengguna dan <i>API Key</i> .
<i>In-Memory Store</i>	Redis	Riwayat Percakapan, manajemen session, <i>Docstore</i> , dan <i>caching</i> .
<i>Vector Database</i>	Pinecone	Penyimpanan <i>vector embedding</i> dengan <i>Hybrid Search</i> .
<i>Object Storage</i>	Cloudflare R2	Penyimpanan file <i>PDF</i> dokumen asli.
<i>LLM Provider</i>	OpenAI API	GPT-4o dan <i>text-embedding-3-large</i>
Alur kerja Orkestrasi	LangGraph	Orkestrasi klasifikasi maksud, penulisan ulang <i>query</i> , dan <i>RAG pipeline</i> .

Berdasarkan Tabel 4.1, setiap komponen teknologi dipilih untuk memenuhi kebutuhan spesifik sistem. FastAPI dipilih sebagai kerangka kerja *backend* untuk menangani permintaan

konkuren dan integrasi dengan layanan eksternal. PostgreSQL berfungsi sebagai basis data relasional untuk menyimpan data pengguna dan *API Key* dengan dukungan *constraint* dan relasi yang kuat. Redis digunakan sebagai *in-memory store* untuk menyimpan riwayat percakapan dengan mekanisme TTL, manajemen *session*, dan *Docstore* untuk konten asli dokumen. Pinecone menyediakan infrastruktur *Vector Database* yang mendukung Pencarian *Hybrid* dengan kombinasi *dense* dan *sparse vector*. Cloudflare R2 dipilih sebagai *penyimpanan objek* untuk menyimpan file *PDF* asli. OpenAI *API* menyediakan model GPT-4o untuk pembangkitan respon dan *text-embedding-3-large* untuk vektorisasi. Lang Graph berfungsi sebagai orkestrator yang mengelola alur kerja *RAG* secara modular.

Komponen kunci dalam arsitektur sistem adalah alur kerja Lang Graph yang berfungsi sebagai orkestrator untuk seluruh proses *RAG*. Lang Graph memungkinkan definisi alur kerja dalam bentuk *state machine* dengan *node* dan *edge* yang dapat dikonfigurasi, memberikan fleksibilitas dalam mengelola *routing* kondisional berdasarkan klasifikasi maksud pengguna.



Gambar 4.1 LangGraph Alur kerja untuk Orkestrasi RAG Pipeline

Berdasarkan Gambar 4.2, Alur kerja Lang Graph terdiri dari beberapa *node* yang masing-masing bertanggung jawab untuk tugas spesifik dalam *pipeline RAG*. *Router Node* melakukan klasifikasi maksud untuk menentukan apakah pertanyaan pengguna merupakan sapaan, pertanyaan teknis, atau akhir percakapan, kemudian mengarahkan alur ke *node* yang sesuai. Penulisan ulang *Query Node* memperbaiki pertanyaan dengan mempertimbangkan konteks dari riwayat percakapan untuk menangani referensi ambigu. *Retrieval Node* melakukan pencarian semantik di Pinecone dengan menerapkan *metadata filtering* berdasarkan *role* pengguna. *Grader Node* mengevaluasi relevansi dokumen yang ditemukan dan menentukan apakah konteks cukup untuk menjawab pertanyaan. *Generator Node* membangkitkan respon final menggunakan OpenAI GPT-4o dengan konteks dari dokumen yang relevan dan *System Prompt* yang ketat untuk mencegah halusinasi.

Arsitektur sistem yang telah diuraikan di atas mendemonstrasikan implementasi dari sistem Chatbot *IT Support* berbasis *RAG*. Pemisahan tanggung jawab yang jelas antar lapisan, pemilihan teknologi yang tepat untuk setiap kebutuhan, serta orkestrasi yang cerdas melalui Alur kerja Lang Graph memastikan bahwa sistem dapat beroperasi secara efisien, aman, dan dapat diskalakan sesuai dengan pertumbuhan kebutuhan layanan *IT Support* BSI UII.

4.2 Implementasi

Implementasi sistem *Chatbot IT Support* BSI UII dilakukan secara iteratif melalui tiga fase pengembangan, dengan Iterasi III sebagai versi final yang mengintegrasikan seluruh kebutuhan fungsional yang telah didefinisikan pada Bab III. Proses implementasi mencakup pengembangan *backend*, integrasi komponen *RAG* dengan Pinecone sebagai *vector database*, penggunaan Lang Graph untuk alur kerja orkestrasi, serta pembangunan *frontend* responsif.

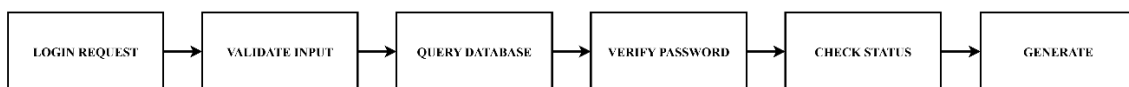
4.2.1 Implementasi Autentikasi

Implementasi autentikasi merupakan fondasi keamanan sistem yang mencakup mekanisme *login* berbasis *JWT*, *RBAC* dengan empat tingkatan peran, dan manajemen *API Key* untuk akses programatik. Bagian ini menjelaskan realisasi teknis dari setiap komponen autentikasi yang dirancang untuk memastikan hanya pengguna terotorisasi dapat mengakses sumber sistem sesuai dengan wewenang yang dimiliki.

a. Implementasi Proses *Login* dan Pembangkitan *JWT* Token

Sistem mengimplementasikan mekanisme autentikasi berbasis *JWT* untuk menjamin keamanan akses pengguna terhadap seluruh layanan *Chatbot IT Support* BSI UII. Proses

autentikasi dimulai ketika pengguna mengirimkan kredensial berupa *username* dan *password* melalui *endpoint login*. Sistem melakukan verifikasi *password* menggunakan algoritma *bcrypt hashing* yang telah diimplementasikan pada tahap registrasi, kemudian apabila kredensial valid, sistem membangkitkan *JWT* token yang mengandung *payload* berisi informasi identitas pengguna. Token ini di *encode* menggunakan algoritma HS256 dengan *secret key* yang tersimpan secara aman dalam variabel lingkungan, dan memiliki masa berlaku selama 60 menit sesuai konfigurasi sistem. *JWT* token yang dihasilkan kemudian dikirim kembali kepada *client* sebagai akses token dengan tipe *Bearer*, yang selanjutnya digunakan untuk autentikasi pada setiap *request API* berikutnya melalui *Authorization header*.



Gambar 4.2 Diagram Alur Proses Login

Berdasarkan Gambar 4.3, proses autentikasi dimulai dengan validasi format input kredensial, dilanjutkan dengan pencarian data pengguna dalam *database PostgreSQL*. Sistem melakukan verifikasi *password* menggunakan fungsi *bcrypt* untuk membandingkan *password plaintext* dengan hash yang tersimpan. Apabila verifikasi berhasil dan status akun aktif, sistem membangkitkan *JWT* token dengan *payload* yang berisi identitas dan peran pengguna. Token ini memiliki masa berlaku 60 menit dan dikembalikan ke *client* untuk digunakan pada *request* selanjutnya. Mekanisme ini memastikan bahwa hanya pengguna yang terverifikasi dapat mengakses layanan sistem.

```

async def login_user(db: AsyncSession, username: str, password:
str) -> Optional[Dict[str, Any]]:

    # Verify credentials
    user = await authenticate_user(db, username, password)
    if not user:
        return None

    # Prepare token payload
    token_data = {
        "sub": str(user.id),
        "username": user.username,
        "role": user.role.value,
    }
  
```

```

# Generate JWT token
access_token = create_access_token(data=token_data)

return {
    "access_token": access_token,
    "token_type": "bearer",
    "user": {
        "id": str(user.id),
        "username": user.username,
        "email": user.email,
        "full_name": user.full_name,
        "role": user.role.value,
    },
}

```

Gambar 4.3 Implementasi Fungsi *Login User*

Implementasi fungsi *login user* pada Kode yang ada di Gambar 4.4 menunjukkan alur autentikasi yang terstruktur dengan memanfaatkan fungsi *authenticate user* untuk verifikasi kredensial. Setelah validasi berhasil, sistem menyusun *payload* token yang mengandung identitas pengguna dan perannya, kemudian memanggil fungsi *create access token* untuk membangkitkan *JWT* dengan algoritma HS256. Fungsi ini mengembalikan objek yang berisi akses token dan informasi lengkap pengguna, memastikan *client* memiliki data yang diperlukan untuk sesi autentikasi.

b. Implementasi *RBAC*

Sistem *Chatbot IT Support* BSI UII menerapkan mekanisme *RBAC* untuk mengatur hak akses pengguna terhadap *endpoint API* dan dokumen Basis pengetahuan berdasarkan peran yang dimiliki. Implementasi *RBAC* dalam sistem ini menggunakan tiga tingkatan role yang terdefinisi yaitu Admin, Dosen dan Mahasiswa, di mana setiap role memiliki kebijakan akses yang berbeda sesuai dengan kebutuhan keamanan dan privasi dokumen BSI UII. Mekanisme pembatasan akses diimplementasikan melalui *dependency function require role* yang menerima satu atau lebih *role* yang diizinkan sebagai parameter, kemudian melakukan ekstraksi *JWT* token dari *Authorization header*, *decode payload* untuk mendapatkan informasi *role* pengguna, dan memvalidasi apakah *role* tersebut termasuk dalam daftar. Apabila validasi gagal, sistem akan mengembalikan *HTTP* respons dengan status kode 403 *Forbidden* disertai pesan *error* yang informatif.

Tabel 4.2 Matriks *RBAC* Sistem Chatbot *IT Support*

<i>Role</i>	Tingkat Akses Dokumen	Deskripsi Hak Akses
ADMIN	Publik, Internal	Memiliki akses penuh tanpa filter metadata terhadap seluruh dokumen dalam Basis Pengetahuan.
DOSEN	Publik, Internal	Dapat mengakses dokumen dengan sensitivitas level "publik" dan "internal".
MAHASISWA	Publik	Hanya dapat mengakses dokumen dengan tingkat sensitivitas "publik".
ANONIM	Publik	Pengguna tanpa autentikasi yang dapat mengakses chatbot dengan batasan dokumen "publik" saja.

Berdasarkan Tabel 4.2, sistem mengimplementasikan empat tingkatan role dengan hak akses yang berbeda sesuai kebijakan keamanan dokumen BSI UII. Admin memiliki akses penuh ke semua dokumen tanpa filter metadata, sedangkan Dosen dapat mengakses dokumen dengan sensitivitas "publik" dan "internal". Mahasiswa dan pengguna Anonim hanya dapat mengakses dokumen dengan sensitivitas "publik", dimana perbedaannya terletak pada status autentikasi dimana Mahasiswa memerlukan login sementara Anonim dapat mengakses tanpa autentikasi. Implementasi *RBAC* ini memastikan bahwa setiap pengguna hanya dapat mengakses informasi sesuai dengan wewenang yang dimiliki.

```
def require_role(*allowed_roles: UserRole) -> Callable[[User],
Awaitable[User]]:
    async def role_checker(user: User =
Depends(get_current_active_user)) -> User:
        if user.role not in allowed_roles:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN,
                detail=f"Access denied. Required role: {'',
'.join(r.value for r in allowed_roles)}",
            )
        return user

    return role_checker

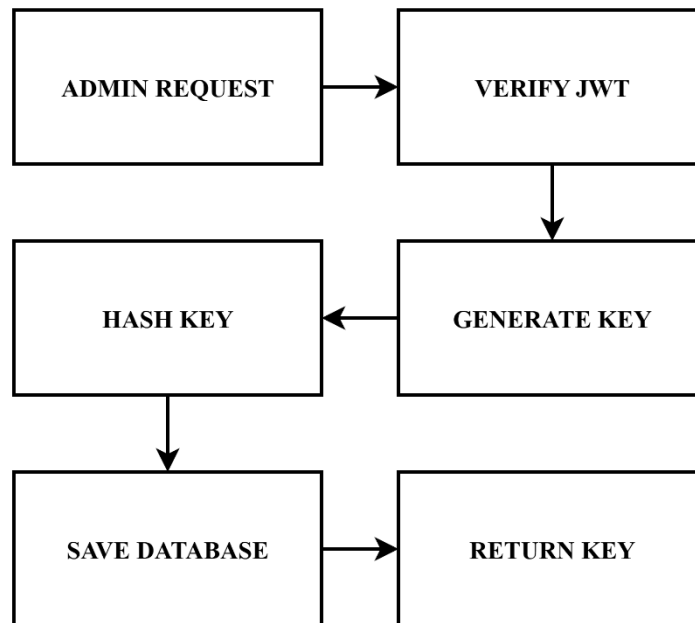
# Example usage in endpoint
@router.post("/documents/upload")
async def upload_document(
    current_user: User = Depends(require_role(UserRole.ADMIN))
):
```

Gambar 4.4 Implementasi *Dependency Function require role*

Implementasi *dependency function require role* pada Kode yang ada di Gambar 4.5 menunjukkan pendekatan modular untuk penerapan RBAC dalam *Backend*. Fungsi ini merupakan *factory pattern* yang menghasilkan *dependency* ceke berdasarkan *role* yang diizinkan. Ketika *endpoint* dilindungi dengan *dependency* ini, sistem akan otomatis mengekstrak informasi user dari JWT token, memverifikasi *role*-nya, dan menolak akses dengan *HTTP* 403 apabila *role* tidak sesuai. Pendekatan ini memungkinkan deklarasi akses kontrol yang eksplisit pada setiap *endpoint* melalui dekorator *dependency*, menjaga keamanan sistem tanpa duplikasi kode validasi.

c. Implementasi Manajemen *API Key*

Sistem menyediakan mekanisme *API Key* sebagai metode autentikasi alternatif untuk akses programmatik terhadap layanan *chatbot*, yang khusus ditujukan untuk integrasi dengan aplikasi eksternal atau sistem otomatis. Manajemen *API Key* hanya dapat diakses oleh pengguna dengan *role* Admin, yang memiliki wewenang untuk membuat, melihat, menonaktifkan, dan melacak penggunaan *API Key*. Setiap *API Key* yang dibangkitkan memiliki atribut *key prefix* untuk identifikasi visual, *key hash* yang menyimpan *hash* dari *key* untuk validasi keamanan, serta informasi jejak audit berupa *created at*, *last used at*, dan *created by* untuk keperluan *monitoring*. Sistem menerapkan mekanisme *hashing* pada *API Key* menggunakan SHA-256 sebelum disimpan dalam database PostgreSQL, sehingga *key* asli hanya ditampilkan sekali saat pembuatan dan tidak dapat diambil kembali setelahnya.



Gambar 4.5 Alur Proses Pembuatan *API Key*

Berdasarkan Gambar 4.6, proses pembuatan *API Key* dimulai dengan verifikasi bahwa permintaan berasal dari Admin melalui validasi *JWT* dan *role*. Sistem kemudian membangkitkan *string key* acak yang aman secara kriptografis, melakukan *hashing* menggunakan SHA-256, dan menyimpan *hash* tersebut ke database beserta metadata terkait. *Key* asli hanya dikirim sekali kepada Admin pada respon pembuatan, dan tidak dapat diambil kembali setelahnya.

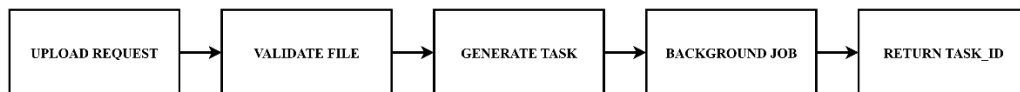
4.2.2 Implementasi Pipeline Vektorisasi Dokumen

Pipeline vektorisasi dokumen merupakan proses kritikal yang mentransformasi dokumen *PDF* Basis pengetahuan *IT Support* menjadi representasi vektor yang dapat dicari secara semantik. Bagian ini menjelaskan implementasi *upload* asinkron untuk menangani file berukuran besar, ekstraksi konten *multi-modal* yang memisahkan teks, tabel, dan gambar, serta pembentukan *embedding* dengan metode *Hybrid* yang mengkombinasikan *dense* dan *sparse vector* untuk pencarian optimal.

a. Implementasi *Upload* Dokumen Asinkron

Sistem mengimplementasikan mekanisme *upload* dokumen asinkron untuk menangani file *PDF* berukuran besar tanpa memblokir permintaan *HTTP*. Proses dimulai ketika Admin

mengunggah dokumen melalui *endpoint* `/api/v1/documents/upload/async`, sistem membangkitkan *task id* unik dan menjalankan pemrosesan di background. *Client* dapat melakukan polling status tugas melalui *endpoint* `/api/v1/documents/tasks/{task_id}` untuk memantau *progress* ekstraksi dan *indexing*. Mekanisme ini memastikan bahwa sistem tetap responsif terhadap permintaan lain selama proses vektorisasi berlangsung, yang dapat memakan waktu beberapa menit untuk dokumen berukuran besar dengan banyak halaman.

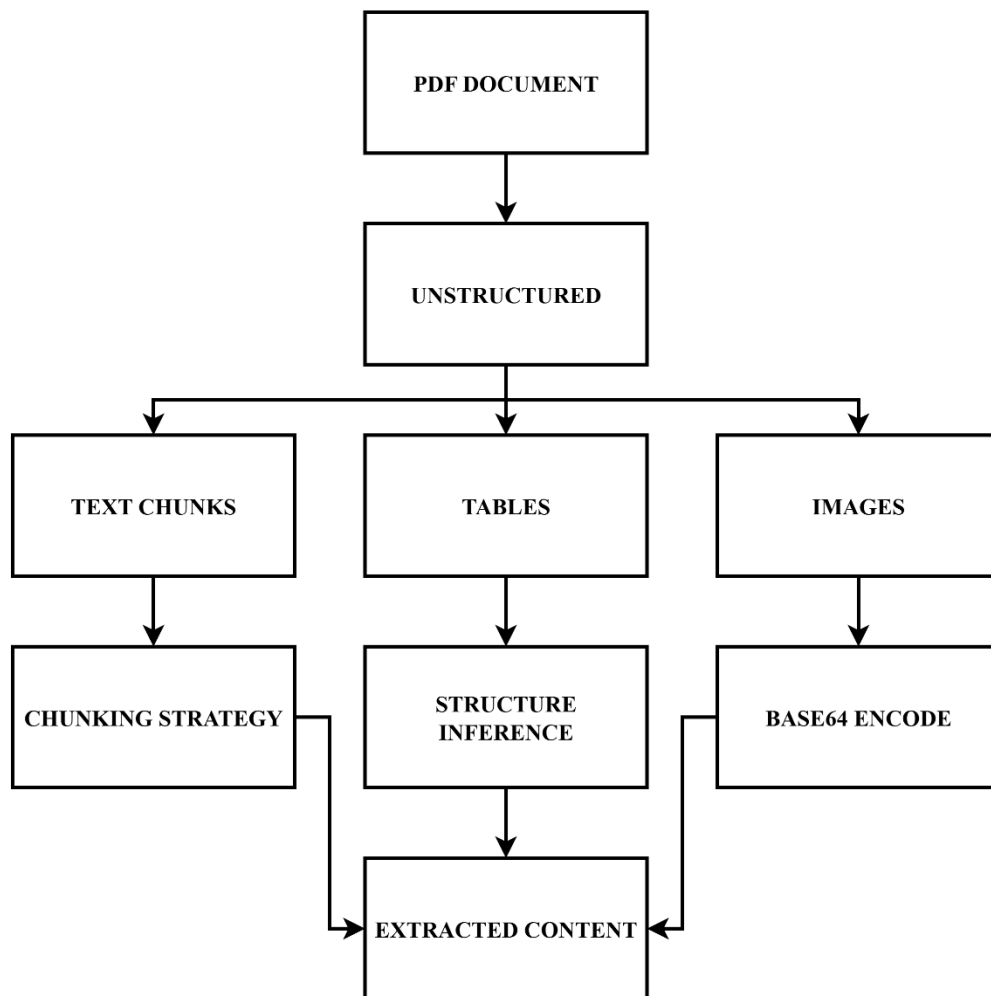


Gambar 4.6 Alur Upload Dokumen Asinkron

Berdasarkan Gambar 4.7, proses *upload* asinkron terbagi menjadi dua alur independen. Alur pertama menangani penerimaan file, validasi format dan ukuran, pembangkitan *task id*, serta penjadwalan *background job* untuk pemrosesan. Alur kedua memungkinkan *client* melakukan *polling* status secara periodik untuk memantau *progress* pemrosesan dokumen. Pemisahan ini memastikan *request upload* dapat selesai dengan cepat tanpa menunggu proses ekstraksi dan *indexing* yang membutuhkan waktu lama. Status tugas mencakup informasi seperti progress persentase, tahap pemrosesan, dan hasil akhir setelah selesai.

b. Implementasi Ekstraksi Konten *Multi-Modal*

Sistem menggunakan *library unstructured* dengan strategi *high-resolution* dan OCR Tesseract untuk mengekstrak konten dari dokumen *PDF* secara *multi-modal*. Proses ekstraksi memisahkan konten menjadi tiga kategori utama yaitu teks *chunk* untuk paragraf dan teks biasa, tabel untuk data tabular yang memerlukan struktur khusus, dan gambar untuk elemen visual yang diekstrak sebagai base64. Setiap tipe konten diproses dengan pendekatan yang berbeda untuk memaksimalkan kualitas informasi yang didapat, dimana teks menggunakan chunking strategy dengan parameter *max characters* yang dapat dikonfigurasi, tabel menggunakan *infer table structure* untuk mempertahankan struktur data, dan gambar dikonversi ke format yang kompatibel dengan OpenAI *Vision API*.



Gambar 4.7 Diagram Ekstraksi Konten Multi-Modal

Berdasarkan Gambar 4.8, proses ekstraksi dimulai dari *parsing* dokumen PDF menggunakan *library unstructured* yang dilengkapi OCR Tesseract untuk mengenali teks dalam gambar. Konten hasil parsing kemudian diklasifikasikan berdasarkan tipenya dan diproses sesuai karakteristik masing-masing. Teks *chunks* dipecah menggunakan strategi *strategy* untuk menghasilkan segmen optimal, tabel diproses dengan *structure inference* untuk mempertahankan relasi antar sel, dan gambar di *encode* ke base64 untuk kompatibilitas *API*. Hasil akhir berupa objek *Extracted Content* yang berisi ketiga tipe konten siap untuk tahap *summarization*.

```

def process_pdf_from_bytes(self, file_obj: BytesIO, filename: str)
-> ExtractedContent:
    # Partition PDF with high-resolution strategy
    chunks = partition_pdf(
        file=file_obj,
        infer_table_structure=True,
        strategy="hi_res",
        extract_image_block_types=["Image"],
        extract_image_block_to_payload=True,
        chunking_strategy=self.chunking_strategy,
        max_characters=self.max_characters,
        languages=self.ocr_languages,
    )

    # Separate content by type
    texts, tables = self._separate_text_and_tables(chunks)
    images = self._extract_images(chunks)

    return ExtractedContent(texts=texts, tables=tables,
images=images)

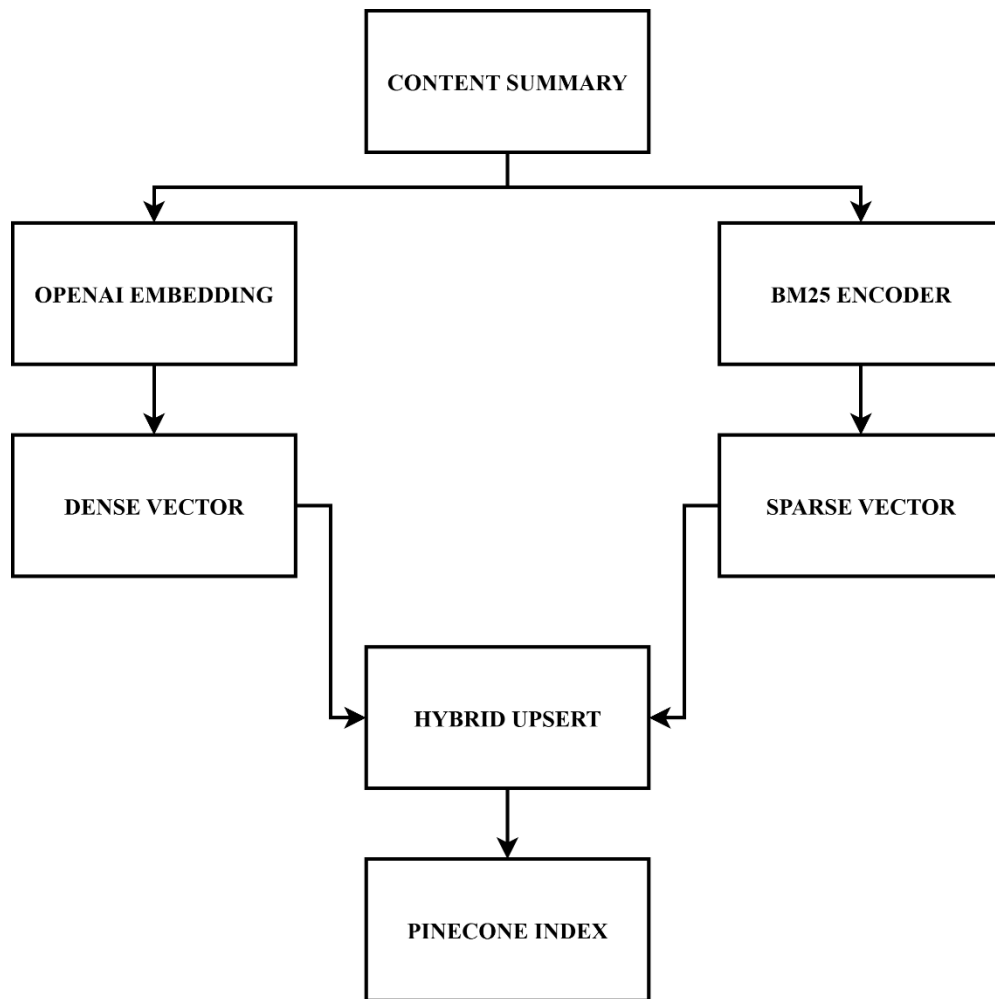
```

Gambar 4.8 Implementasi Fungsi Ekstraksi *PDF*

Implementasi fungsi *process pdf from bytes* pada Kode yang ada di Gambar 4.9 menunjukkan konfigurasi ekstraksi yang komprehensif. Parameter *infer table structure* mengaktifkan deteksi struktur tabel, strategi memastikan kualitas ekstraksi maksimal dengan OCR, *extract image block types* menentukan tipe gambar yang diekstrak, dan *chunking strategy* mengatur pemecahan teks panjang. Fungsi pembantu *separate text and tables* dan *extract images* memisahkan konten berdasarkan tipe untuk pemrosesan lebih lanjut. Pendekatan ini menghasilkan ekstraksi yang akurat dengan mempertahankan konteks dan struktur dokumen asli.

c. Implementasi Pembentukan *Embedding Hybrid*

Sistem mengimplementasikan *Hybrid Search* dengan mengkombinasikan *dense vector* dari OpenAI *text-embedding-3-large* dan *sparse vector* dari BM25. Setiap konten yang telah diekstrak dan diringkas dikonversi menjadi dua representasi vektor yang kemudian disimpan bersama dalam Pinecone *Vector Database*. *Dense vector* menangkap makna semantik kontekstual, sementara *sparse vector* menangkap kecocokan kata kunci. Kombinasi kedua pendekatan ini menghasilkan pencarian yang lebih akurat dibandingkan menggunakan salah satu metode saja, karena dapat menemukan dokumen relevan baik berdasarkan kesamaan makna maupun kecocokan istilah teknis spesifik yang penting dalam konteks *IT Support*.



Gambar 4.9 Diagram Pembentukan *Embedding Hybrid Search*

Berdasarkan Gambar 4.10, setiap *content summary* diproses melalui dua jalur paralel untuk menghasilkan representasi vektor ganda. Jalur pertama menggunakan OpenAI *embedding API* untuk membangkitkan *dense vector* berdimensi 3072 yang merepresentasikan makna semantik. Jalur kedua menggunakan BM25 *encoder* untuk membangkitkan *sparse vector* berbasis kata kunci. Kedua vektor kemudian digabungkan dalam operasi *hybrid upsert* ke Pinecone *Index* bersama metadata untuk *filtering* dan konten asli disimpan di Redis Docstore. Implementasi dua vektor ini memungkinkan sistem melakukan pencarian yang menggabungkan keunggulan *semantic similarity* dan *keyword matching*.

```

def hybrid_search(
    self,
    query: str,
    k: int = 5,
    metadata_filter: Optional[Dict[str, Any]] = None
) -> List[Document]:
    # Generate dense vector from query
    dense_vector = self.embeddings.embed_query(query)

    # Generate sparse vector using BM25
    sparse_encoder = self._get_sparse_encoder()
    sparse_vector = sparse_encoder.encode_query(query)

    # Execute hybrid query on Pinecone
    results = self._index.query(
        vector=dense_vector,
        sparse_vector=sparse_vector,
        top_k=k,
        filter=metadata_filter,
        include_metadata=True
    )

    # Retrieve original content from Redis
    return self._build_documents_from_results(results)

```

Gambar 4.10 Implementasi Fungsi *Hybrid Search*

Implementasi fungsi *hybrid search* pada Kode yang ada di Gambar 4.11 menunjukkan proses pencarian yang memanfaatkan kedua tipe vektor secara bersamaan. Fungsi ini membangkitkan *dense vector* menggunakan model *embedding* yang sama dengan saat indexing, kemudian membangkitkan *sparse vector* menggunakan BM25. Kedua vektor dikirim dalam satu permintaan *query* ke Pinecone yang secara internal menggabungkan skor dari kedua metode. Parameter *metadata filter* menerapkan *RBAC* dengan membatasi hasil pencarian berdasarkan tingkatan sensitivitas sesuai *role* pengguna. Hasil akhir berupa dokumen lengkap dengan konten asli yang diambil dari Redis.

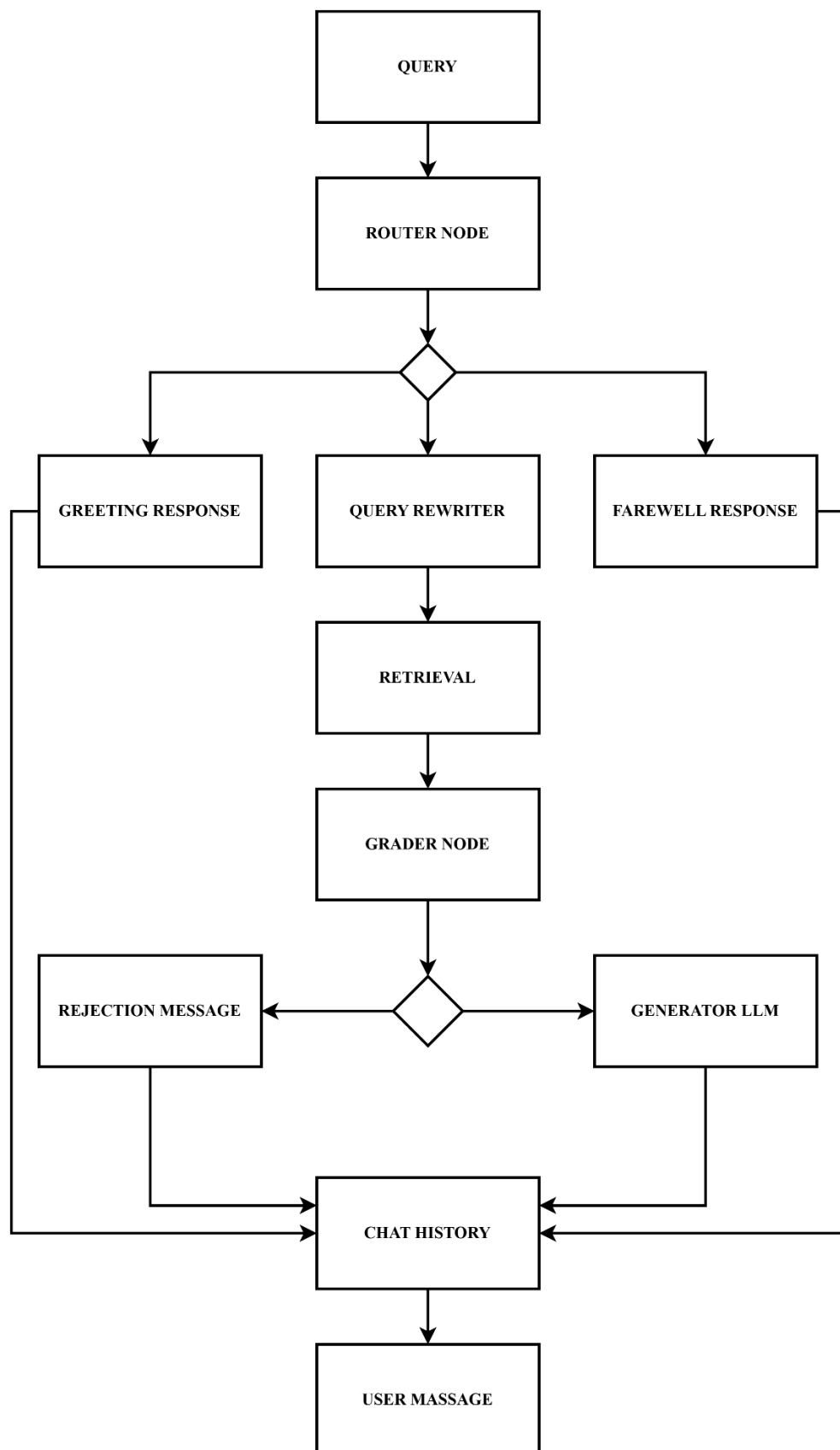
4.2.3 Implementasi Proses *RAG*

Proses *RAG* merupakan inti dari sistem *chatbot* yang mengorkestrasikan pencarian semantik dan pembangkitan respon kontekstual. Bagian ini menjelaskan implementasi Alur kerja Lang Graph untuk *routing* cerdas berdasarkan klasifikasi makna, penerapan *metadata filtering* untuk *RBAC* pada tingkatan *retrieval*, dan penggunaan *System Prompt* dengan batasan

ketat untuk mencegah halusinasi dan memastikan respon hanya berdasarkan Basis Pengetahuan BSI UII.

a. Implementasi Alur kerja LangGraph untuk Orkestrasi

Sistem mengimplementasikan *Lang Graph* sebagai orkestrator untuk mengelola seluruh *pipeline RAG* secara modular dan terstruktur. Lang Graph memungkinkan definisi alur kerja dalam bentuk *state machine* dengan *node* dan *edge* yang dapat dikonfigurasi, memberikan fleksibilitas dalam *routing* kondisional berdasarkan klasifikasi maksud pengguna. *Router Node* melakukan klasifikasi untuk menentukan apakah *query* merupakan sapaan, permintaan teknis, atau di luar cakupan layanan, kemudian mengarahkan alur ke *node* yang sesuai. Penulisan ulang Query memperbaiki pertanyaan dengan konteks riwayat percakapan, *Retrieval Node* melakukan pencarian *hybrid* di Pinecone dengan metadata *filtering*, *Grader Node* mengevaluasi relevansi dokumen, dan *Generator Node* membangkitkan respon menggunakan GPT-4o dengan *System Prompt* ketat.



Gambar 4.11 LangGraph Workflow untuk Orkestrasi RAG *Pipeline*

Berdasarkan Gambar 4.12, Alur kerja LangGraph mengorkestrasikan *pipeline RAG* melalui *routing* cerdas yang mengklasifikasikan maksud *query*. Untuk sapaan, sistem langsung memberikan respon tanpa *retrieval*. Untuk pertanyaan teknis, sistem melalui *Query Rewriter* yang mempertimbangkan riwayat percakapan, *Retrieval Node* melakukan *hybrid search* dengan metadata filtering *RBAC*, *Grader Node* mengevaluasi relevansi dokumen, dan jika relevan, Generator *LLM* membangkitkan respon dengan *System Prompt* ketat. Seluruh percakapan disimpan ke riwayat percakapan untuk konteks percakapan berikutnya. Arsitektur modular ini memastikan efisiensi dan akurasi respon.

b. Implementasi Metadata *Filtering* untuk *RBAC*

Sistem menerapkan metadata *filtering* pada tahap *retrieval* untuk membatasi akses dokumen sesuai *role* pengguna berdasarkan tingkatan sensitivitas. Setiap dokumen dalam *Pinecone Vector Database* memiliki metadata *sensitivity* dengan nilai publik dan internal. Ketika pengguna melakukan *query*, sistem membangkitkan filter berdasarkan *role*, dimana Admin dapat mengakses semua dokumen, Dosen dapat mengakses publik dan internal, sedangkan Mahasiswa dan Anonim hanya dapat mengakses publik. Filter ini diterapkan pada saat pencarian *hybrid* di *Pinecone* menggunakan parameter filter yang mendukung *query*, memastikan hasil pencarian hanya mengembalikan dokumen yang sesuai dengan *authorization* tingkatan pengguna.

Tabel 4.3 Implementasi Metadata *Filtering* Berdasarkan *Role*

Role	Metadata Filter	Implementasi <i>Pinecone Query</i>
ADMIN	Publik dan Internal	-
DOSEN	Publik dan Internal	<code>{"sensitivity": {"\$in": ["public", "internal"]}}</code>
MAHASISWA	Publik	<code>{"sensitivity": "public"}</code>
ANONIM	Publik	<code>{"sensitivity": "public"}</code>

Berdasarkan Tabel 4.3, sistem mengimplementasikan metadata *filtering* dengan *query syntax* yang didukung oleh *Pinecone*. Admin tidak memiliki filter sehingga dapat mengakses seluruh dokumen. Dosen menggunakan operator *\$in* untuk mengakses dokumen dengan *sensitivity public* atau *internal*. Mahasiswa dan Anonim menggunakan kecocokan persis untuk hanya mengakses dokumen publik. Filter diterapkan pada parameter filter dalam fungsi pencarian *hybrid*, dimana *Pinecone* melakukan *filtering* pada tingkatan server sebelum

mengembalikan hasil, memastikan pelaksanaan *RBAC* yang konsisten dan aman tanpa pasca pemrosesan di lapisan aplikasi.

```
def build_metadata_filter(user_role: UserRole) -> Dict[str, Any]:

    # Admin: no filter, access all documents
    if user_role == UserRole.ADMIN:
        return {}

    # Dosen: access public and internal documents
    elif user_role == UserRole.DOSEN:
        return {
            "sensitivity": {"$in": ["public", "internal"]}
        }

    # Mahasiswa and Anonim: only public documents
    else: # MAHASISWA or ANONIM
        return {
            "sensitivity": "public"
        }
```

Gambar 4.12 Implementasi Fungsi *Build Metadata Filter*

Implementasi fungsi *build metadata filter* pada Kode yang ada di Gambar 4.13 menunjukkan konstruksi filter yang dinamis berdasarkan *user role*. Fungsi menerima *user role* dan mengembalikan *dictionary* filter yang kompatibel dengan Pinecone *query syntax*. *Logic branching* menggunakan *if-elif-else* memastikan setiap role mendapat filter yang tepat. *Return value* berupa *dictionary* kosong untuk Admin memberikan akses penuh, sementara operator *\$in* untuk Dosen dan kecocokan persis untuk Mahasiswa atau anonim membatasi akses sesuai kebijakan. Filter ini kemudian diteruskan ke fungsi pencarian *hybrid* sebagai parameter untuk pelaksanaan pada tingkatan database.

c. Implementasi *System Prompt* untuk Kontrol Ketat Respon

Sistem mengimplementasikan *System Prompt* dengan aturan ketat untuk mengontrol perilaku model GPT-4o dalam membangkitkan respon. *System Prompt* dirancang dengan tiga komponen utama yaitu definisi peran sebagai Asisten *IT Support* BSI UII, aturan ketat yang secara eksplisit melarang model menggunakan pengetahuan umum di luar konteks yang diberikan, dan format jawaban yang terstruktur dengan instruksi untuk menjawab singkat dan jelas. *Prompt* ini diterapkan pada *Router Node* untuk klasifikasi maksud dan *Generator Node* untuk respon *generation*, memastikan konsistensi perilaku model di seluruh *pipeline*.

Pendekatan ini efektif mencegah halusinasi dan memastikan respon hanya berdasarkan Basis pengetahuan BSI UII.

```

SYSTEM_PROMPT = """
Kamu adalah Asisten IT Support BSI UII.
Jawab HANYA berdasarkan konteks knowledge base yang diberikan.

=== ATURAN KETAT ===
1. HANYA jawab berdasarkan konteks yang diberikan
2. DILARANG KERAS menggunakan pengetahuan umum di luar konteks
3. DILARANG menjawab pertanyaan non-IT (presiden, film, olahraga, dll)
4. Jika informasi tidak ada dalam konteks → jawab:
   "Maaf, informasi tersebut tidak tersedia di knowledge base BSI UII."

=== FORMAT JAWABAN ===
- Jawab singkat, jelas, dan langsung to the point
- Gunakan bullet points jika ada beberapa langkah
- Sertakan link sumber jika tersedia dalam konteks
- Jangan tambahkan informasi di luar konteks

=== KONTEKS ===
{context}

=== RIWAYAT PERCAKAPAN ===
{chat_history}

=== PERTANYAAN ===
{question}
"""

```

Gambar 4.13 Implementasi *System Prompt*

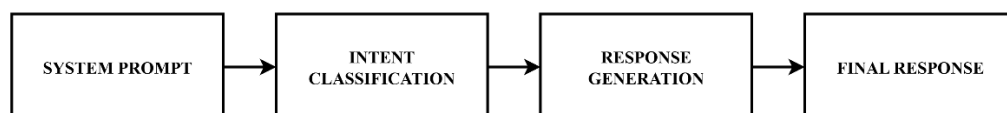
Implementasi *System Prompt* pada Kode yang ada di Gambar 414 menunjukkan struktur *prompting engineering* yang terstruktur dengan pemisahan bagian yang jelas. Bagian Aturan Ketat menggunakan kapitalisasi dan penomoran untuk menegaskan batasan, Bagian Format Jawaban memberikan pedoman presentasi respon, dan *placeholder* {context}, {chat_history}, {question} memungkinkan injeksi dinamis dari hasil *retrieval* dan riwayat percakapan. Penggunaan kata “DILARANG KERAS” dan “HANYA” dengan kapitalisasi penuh memberikan tekanan kuat pada batasan.

4.2.4 Implementasi *Prompt Engineering*

Prompt Engineering merupakan teknik kunci dalam mengendalikan perilaku model GPT-4o untuk menghasilkan respon yang akurat, relevan, dan sesuai dengan kebijakan *IT Support* BSI UII. Bagian ini menjelaskan implementasi tiga lapisan *prompt* yang saling terintegrasi yaitu *System Prompt* dengan aturan ketat untuk mencegah halusinasi, klasifikasi maksud *Prompt* untuk *routing* cerdas, dan *Respon Generation Prompt* yang mengintegrasikan konteks *RAG* dengan riwayat percakapan untuk percakapan *multi-turn* yang koheren.

a. *Arsitektur Prompt Engineering Tiga Lapis*

Sistem mengimplementasikan arsitektur *prompt engineering* berlapis yang terdiri dari tiga komponen hierarkis untuk mengontrol perilaku model pada setiap tahap pipeline *RAG*. Lapisan pertama adalah *System Prompt* yang menetapkan identitas, cakupan, dan batasan fundamental model sebagai Asisten *IT Support* BSI UII. Lapisan kedua adalah klasifikasi maksud *Prompt* yang memandu model untuk mengklasifikasikan pertanyaan pengguna dan menentukan *routing* yang tepat. Lapisan ketiga adalah *Respon Generation Prompt* yang mengintegrasikan konteks dari *retrieval* dan riwayat percakapan untuk membangkitkan respon final. Ketiga layer ini bekerja secara hierarkis dimana *System Prompt* menjadi *foundation* yang konsisten di seluruh *pipeline*, sementara kedua layer lainnya diterapkan pada node spesifik dalam alur kerja Lang Graph.



Gambar 4.14 Arsitektur *Prompt Engineering* Tiga Lapis

Berdasarkan Gambar 4.15, arsitektur *prompt engineering* terdiri dari tiga layer hierarkis yang mengalir secara horizontal. Lapisan satu *System Prompt* merupakan fondasi yang berisi identitas, batasan, dan cakupan untuk seluruh *pipeline*. Lapisan dua klasifikasi maksud membagi *routing* menjadi tiga jalur yaitu sapaan untuk respon langsung, pertanyaan teknik untuk *trigger RAG*, dan diluar cakupan untuk penolakan. Lapisan tiga *Respon Generation* mengintegrasikan konteks dari *retrieval*, riwayat dari percakapan sebelumnya, dan format untuk keluaran terstruktur. Alur horizontal ini menunjukkan bagaimana setiap lapisan membangun di atas lapisan sebelumnya untuk menghasilkan respon terakhir yang akurat dan kontekstual.

b. Implementasi *System Prompt* dengan batasan ketat

System Prompt dirancang dengan struktur terpartisi yang jelas menggunakan pembatas bagian untuk menegaskan setiap komponen aturan. Bagian Aturan Ketat menggunakan penomoran dan kapitalisasi untuk tekanan pada batasan fundamental seperti larangan menggunakan pengetahuan umum dan kewajiban menjawab hanya dari konteks. Bagian Format Jawaban memberikan pedoman presentasi respon yang konsisten dengan instruksi penggunaan poin-poin, bahasa Indonesia yang ringkas, dan penyertaan emoji untuk kesan ramah. Bagian penanganan Riwayat percakapan menjelaskan cara model harus memproses referensi ambigu dari percakapan sebelumnya.

Tabel 4.4 Komponen *System Prompt* dan Fungsinya

Bagian	Komponen Utama	Fungsi
Identitas	Kamu adalah Asisten IT <i>Support</i> BSI UII	Menetapkan peran dan konteks operasi model
Aturan Ketat	HANYA jawab dari konteks DILARANG <i>knowledge</i> umum DILARANG pertanyaan non-IT	Mencegah halusinasi dan respon diluar topik
Format Jawaban	Bahasa Indonesia singkat Daftar nomor untuk langkah Emoji untuk kesan ramah	Mengatur struktur dan <i>tone</i> respon
Penanganan riwayat	Cek riwayat untuk referensi Gunakan konteks percakapan	Mendukung percakapan <i>multi-turn</i> koheren
<i>Fallback</i>	Jika info tidak ada: "Maaf, informasi tidak tersedia"	Respon standar untuk pertanyaan tanpa konteks

Berdasarkan Tabel 4.4, *System Prompt* terdiri dari lima bagian utama dengan fungsi spesifik. bagian identitas menetapkan peran model sebagai Asisten *IT Support* BSI UII untuk konteks yang jelas. Aturan Ketat menerapkan tiga batasan fundamental menggunakan kapitalisasi HANYA dan DILARANG untuk teknan. Format Jawaban mengatur presentasi dengan pedoman bahasa Indonesia ringkas, daftar nomor, dan emoji. Penanganan riwayat memungkinkan model memproses referensi ambigu dari riwayat percakapan. *Fallback* menyediakan respon standar ketika informasi tidak tersedia. Struktur terpartisi ini memastikan model memahami batasan dengan jelas dan menghasilkan respon konsisten.

c. Implementasi klasifikasi maksud

Klasifikasi maksud *Prompt* menggunakan teknik *few-shot prompting* dengan contoh kategori sapaan, teknis, dan diluar cakupan untuk memandu model dalam keputusan *routing*. *Prompt* memberikan instruksi eksplisit untuk memanggil *retrieve tool* ketika pertanyaan bersifat teknis dan memberikan respon langsung untuk sapaan atau penolakan sopan untuk diluar cakupan. Respon *Generation Prompt* mengintegrasikan *placeholder* dinamis untuk konteks, riwayat percakapan, dan pertanyaan yang di injeksi saat waktu eksekusi menggunakan Python *f-string formatting*. Kombinasi kedua prompt ini dengan *System Prompt* sebagai fondasi menciptakan *pipeline RAG* yang kokoh, dimana model dapat melakukan *routing* cerdas, mencari konteks relevan, dan membangkitkan respon yang akurat dengan mempertimbangkan riwayat percakapan untuk mendukung interaksi *multi-turn* yang natural.

```
ROUTER_SYSTEM_PROMPT = """Kamu adalah router untuk sistem chatbot
IT Support BSI UII.

=== TUGAS ===
Tentukan apakah pertanyaan perlu retrieval atau bisa dijawab
langsung.

=== KATEGORI ===
1. GREETING - Sapaan, salam → Jawab langsung, JANGAN retrieve
2. TECHNICAL - Pertanyaan IT teknis → PANGGIL retrieve tool
3. OUT-OF-SCOPE - Non-IT (presiden, film, etc) → Tolak ramah,
JANGAN retrieve

=== CONTOH ===
"Halo" → GREETING → Jawab: "Halo! Ada yang bisa saya bantu?"
"Bagaimana cara reset password?" → TECHNICAL → RETRIEVE
"Siapa presiden Indonesia?" → OUT-OF-SCOPE → Tolak ramah

=== INSTRUKSI ===
- Jika GREETING/OUT-OF-SCOPE: Jawab langsung dalam teks biasa
- Jika TECHNICAL: WAJIB panggil tool 'retrieve' dengan query yang
relevan
"""
```

Gambar 4.15 Implementasi *Intent Classification Prompt*

Implementasi klasifikasi maksud *Prompt* pada Kode yang ada di Gambar 4.16 menggunakan struktur terpartisi dengan bagian Tugas, Kategori, Contoh, dan Instruksi. Bagian Kategori mendefinisikan tiga klasifikasi dengan aksi yang jelas sapaan untuk respon langsung, teknis untuk pemicu *retrieval*, dan diluar konteks untuk penolakan. Bagian Contoh memberikan *few-shot* untuk setiap kategori membantu model memahami pola klasifikasi.

Bagian Instruksi memberikan arahan penggunaan *retrieve tool* untuk pertanyaan teknis. *Prompt* ini diterapkan pada *Router Node* dalam LangGraph dan berhasil melakukan routing decision tanpa *hardcoded* regex, memberikan fleksibilitas dalam *handling* variasi *query* pengguna.

4.3 Pengujian

Pengujian sistem dilakukan untuk memvalidasi bahwa implementasi *Chatbot IT Support* BSI UII telah memenuhi seluruh spesifikasi kebutuhan fungsional dan non-fungsional yang ditetapkan pada Bab III. Proses pengujian mencakup dua kategori utama, Pengujian Fungsional yang memverifikasi kelengkapan dan kebenaran 25 *endpoint API* beserta fitur-fitur kunci seperti *RBAC*, *metadata filtering*, unggah dokumen asinkron, dan alur kerja LangGraph, serta Pengujian Non-Fungsional yang mengukur aspek kualitas sistem seperti waktu respon, ketersediaan 24 Jam, dan kegunaan melalui interaksi dengan pengguna akhir. Metode pengujian yang digunakan adalah *Black-box* Testing dengan pendekatan kasus uji berbasis skenario untuk memastikan sistem dapat beroperasi sesuai ekspektasi dalam kondisi penggunaan nyata.

4.3.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk memvalidasi bahwa sistem *Chatbot IT Support* BSI UII telah mengimplementasikan seluruh kebutuhan fungsional yang ditetapkan pada Sub-bab 3.2.1. Metode pengujian yang digunakan adalah *Black-box* Testing dengan pendekatan *test case* berbasis skenario pengguna, di mana setiap fitur utama diuji berdasarkan *input* yang valid, *output* yang diharapkan, dan perilaku sistem yang sesuai spesifikasi. Pengujian mencakup 25 *endpoint API* yang telah dikembangkan hingga Iterasi III, dengan fokus utama pada verifikasi implementasi *RBAC*, *Metadata Filtering*, dan fungsionalitas kunci seperti *Upload* Dokumen, *Query RAG*, *Create API Key*, dan *Delete User*. Hasil pengujian disajikan dalam bentuk tabel dengan status kelulusan yang menunjukkan tingkat kesesuaian implementasi terhadap kebutuhan fungsional yang telah didefinisikan.

a. Pengujian Autentikasi dan Manajemen *User*

Pengujian pada modul autentikasi dan manajemen user difokuskan untuk memvalidasi mekanisme registrasi pengguna baru, proses *login* dengan *JWT generation*, serta pengelolaan data pengguna oleh administrator. Setiap *endpoint* diuji dengan berbagai skenario *input*,

termasuk kondisi valid dan tidak valid, untuk memastikan sistem memberikan respon yang tepat sesuai spesifikasi keamanan dan otorisasi yang telah ditetapkan.

Tabel 4.5 Hasil Pengujian Autentikasi dan Manajemen *User*

No	Fitur	Endpoint	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Registrasi User	<i>/api/v1/auth/register</i>	Input <i>username</i> , <i>email</i> , <i>password</i> , dan <i>role</i> yang valid	User berhasil terdaftar dengan respon 201 dan data <i>user</i>	LULUS
2	Registrasi User	<i>/api/v1/auth/register</i>	Input dengan <i>username</i> yang sudah terdaftar	Sistem menolak dengan respon 400 " <i>Username already exists</i> "	LULUS
3	Login	<i>/api/v1/auth/login</i>	Input <i>username</i> dan <i>password</i> yang benar	Sistem menghasilkan <i>JWT</i> token dengan respon 200	LULUS
4	Login	<i>/api/v1/auth/login</i>	Input <i>password</i> yang salah	Sistem menolak dengan respon 401 " <i>Invalid credentials</i> "	LULUS
5	Get User Profile	<i>/api/v1/auth/me</i>	Request dengan <i>JWT</i> token valid	Sistem mengembalikan data user yang sedang <i>login</i>	LULUS
6	List All Users	<i>/api/v1/admin/users</i>	Admin melakukan <i>request</i> tanpa filter	Sistem mengembalikan daftar semua <i>user</i> dengan paginasi	LULUS
7	Delete User	<i>/api/v1/admin/users/{id}</i>	Admin menghapus <i>user</i> dengan <i>ID</i> valid	<i>User</i> dihapus dengan respon 200	LULUS

Berdasarkan Tabel 4.5, seluruh skenario pengujian pada modul autentikasi dan manajemen user berhasil lulus dengan implementasi yang sesuai kebutuhan fungsional. Sistem berhasil melakukan validasi input, menghasilkan *JWT* token dengan struktur yang benar, dan menerapkan mekanisme *soft delete* untuk penghapusan *user*. Pengujian juga memvalidasi bahwa sistem menolak akses dengan kredensial yang tidak valid dan memberikan pesan error yang deskriptif sesuai spesifikasi.

b. Pengujian *RBAC*

Pengujian *RBAC* dilakukan untuk memverifikasi bahwa sistem menerapkan kontrol akses berbasis peran dengan benar, di mana setiap *role* memiliki batasan akses yang berbeda terhadap *endpoint* dan data. Pengujian mencakup skenario akses yang diizinkan dan ditolak untuk memastikan implementasi *require role* berfungsi sesuai dengan kebutuhan keamanan sistem.

Tabel 4.6 Hasil Pengujian *RBAC*

No	Fitur	Endpoint	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Admin Akses	<i>/api/v1/admin/users</i>	User dengan <i>role</i> Admin mengakses <i>endpoint</i> admin	Akses diberikan dengan respon 200 dan data <i>users</i>	LULUS
2	Mahasiswa Akses	<i>/api/v1/admin/users</i>	User dengan <i>role</i> Mahasiswa mengakses <i>endpoint</i> admin	Akses ditolak dengan respon 403 " <i>Insufficient permissions</i> "	LULUS
3	Dosen Akses	<i>/api/v1/documents/upload</i>	User dengan <i>role</i> Dosen mengakses <i>endpoint</i> <i>upload</i>	Akses ditolak dengan respons 403	LULUS
4	Anonymous Akses	<i>/api/v1/chat/query</i>	User tanpa autentikasi mengakses <i>endpoint</i> <i>query</i> publik	Akses diberikan dengan metadata <i>filtering</i> "public"	LULUS
5	API Key	<i>/api/v1/chat/query</i>	Request dengan <i>API Key</i> valid menggunakan header <i>X-API-Key</i>	Sistem mengautentikasi dan memberikan akses sesuai <i>role</i> pemilik <i>key</i>	LULUS
6	Token Kedaluarsa	<i>/api/v1/auth/me</i>	Request dengan <i>JWT</i> token yang kedaluarsa	Akses ditolak dengan respon 401 " <i>Token expired</i> "	LULUS

Berdasarkan Tabel 4.6, implementasi *RBAC* telah berfungsi dengan baik di mana sistem berhasil membedakan akses berdasarkan *role* pengguna. *Require role* yang diterapkan pada setiap *endpoint* berhasil memvalidasi *role* dari *JWT* token atau *API Key*, dan sistem memberikan respon 403 ketika *user* mencoba mengakses sumber yang tidak sesuai dengan tingkatan otorisasi mereka. Pengujian juga memvalidasi bahwa penanganan token kedaluarsa telah diimplementasikan dengan benar.

c. Pengujian Manajemen Dokumen dan *RAG Pipeline*

Pengujian pada modul manajemen dokumen dan *RAG pipeline* dilakukan untuk memvalidasi kemampuan sistem dalam menerima *upload* dokumen, memproses dokumen menjadi *chunks*, melakukan *indexing* ke Pinecone, dan menghasilkan respon yang relevan berdasarkan *query* pengguna. Pengujian juga mencakup verifikasi metadata *filtering* untuk memastikan pengguna hanya dapat mengakses dokumen sesuai dengan level akses role mereka.

Tabel 4.7 Hasil Pengujian Manajemen Dokumen dan *RAG*

No	Fitur	Endpoint	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Upload Document	/api/v1/documents/upload	Admin upload file <i>PDF single</i> dengan metadata valid	File di-upload, diproses, dan di-index dengan respons <i>task_id</i>	LULUS
2	Batch Upload	/api/v1/documents/upload	Admin upload <i>multiple PDF files</i> secara bersamaan	Sistem memproses semua <i>file asynchronously</i> dengan status <i>polling</i>	LULUS
3	Upload Status	/api/v1/documents/upload/status/{task_id}	<i>Request</i> status dari <i>task_id</i> yang sedang diproses	Sistem mengembalikan status " <i>processing</i> " atau " <i>completed</i> " dengan <i>progress</i>	LULUS
4	List Documents	/api/v1/documents	Admin <i>request</i> daftar semua dokumen yang <i>ter-index</i>	Sistem mengembalikan list dokumen dengan metadata lengkap	LULUS
5	Delete Document	/api/v1/documents/{doc_id}	Admin menghapus	Dokumen dihapus dari Pinecone dan Cloudflare R2	LULUS

			dokumen dengan <i>ID</i> valid		
6	<i>Query RAG</i>	<i>/api/v1/chat/query</i>	User mengirim pertanyaan teknis <i>IT</i> dengan <i>session_id</i>	Sistem melakukan <i>retrieval</i> , <i>re-ranking</i> , dan <i>generate</i> respons akurat	LULUS
7	<i>Metadata Filtering</i>	<i>/api/v1/chat/query</i>	Mahasiswa query dokumen dengan metadata "internal"	Sistem tidak mengembalikan dokumen internal, hanya dokumen "public"	LULUS
8	<i>Chat History</i>	<i>/api/v1/chat/history/{session_id}</i>	Request riwayat percakapan	Sistem mengembalikan <i>chat history</i> dari Redis dengan TTL	LULUS
9	<i>Intent Classification</i>	<i>/api/v1/chat/query</i>	User mengirim sapaan	LangGraph Router merespons langsung tanpa memanggil RAG <i>retrieval</i>	LULUS
10	<i>Out-of-Scope Query</i>	<i>/api/v1/chat/query</i>	User menanyakan topik non- <i>IT</i>	Sistem menolak dengan respons "Maaf, saya hanya bisa membantu IT <i>Support</i> "	LULUS
11	<i>List Chat Sessions</i>	<i>/api/v1/chat/sessions</i>	Admin request list semua chat <i>sessions</i>	Sistem mengembalikan list session IDs dengan total count	LULUS
12	<i>Get Session Info</i>	<i>/api/v1/chat/session/{session_id}</i>	Request metadata <i>session</i> dengan <i>session_id</i> valid	Sistem mengembalikan <i>exists</i> , <i>message_count</i> , dan TTL	LULUS
13	<i>Clear Chat History</i>	<i>/api/v1/chat/history/{session_id}</i>	Admin menghapus <i>chat history</i> dengan <i>session_id</i> valid	<i>Chat history</i> dihapus dari Redis dengan respons <i>success</i>	LULUS

Berdasarkan Tabel 4.7, sistem *RAG pipeline* telah berhasil diimplementasikan dengan semua fitur kunci berfungsi sesuai spesifikasi. Proses *upload* dokumen yang asinkron memungkinkan sistem menangani file berukuran besar, sementara status *polling* memberikan *feedback real-time* kepada administrator. Metadata *filtering* telah bekerja dengan benar di mana sistem hanya mengembalikan dokumen yang sesuai dengan tingkatan akses role pengguna, dan alur kerja Lang Graph berhasil melakukan klasifikasi maksud untuk mengoptimalkan

penggunaan *retrieval* hanya pada query yang relevan. Implementasi Redis untuk riwayat percakapan dengan TTL juga berfungsi sesuai kebutuhan untuk mendukung percakapan *multi-turn*, termasuk kemampuan *list sessions*, *get session* metadata, dan pembersihan riwayat yang memungkinkan administrator mengelola riwayat percakapan dengan efektif.

d. Pengujian Manajemen *API Key* dan *Health Monitoring*

Pengujian pada modul manajemen *API Key* dan health monitoring dilakukan untuk memvalidasi kemampuan administrator dalam mengelola *API Key* untuk akses programmatik, serta memastikan sistem *health check* dapat memantau status koneksi ke seluruh layanan eksternal yang terintegrasi dengan sistem *chatbot*.

Tabel 4.8 Hasil Pengujian *API Key* dan *Health Monitoring*

No	Fitur	Endpoint	Skenario Pengujian	Hasil yang Diharapkan	Status
1	Create <i>API Key</i>	<code>/api/v1/admin/users/{id}/api-keys</code>	Admin membuat <i>API Key</i> untuk <i>user</i> dengan <i>description</i>	<i>API Key</i> ter-generate dengan format yang aman dan tersimpan di database	LULUS
2	List <i>API Keys</i>	<code>/api/v1/admin/users/{id}/api-keys</code>	Admin request daftar <i>API Keys</i> milik <i>user</i> tertentu	Sistem mengembalikan list <i>API Keys</i> dengan metadata dan <i>last_used</i> timestamp	LULUS
3	Revoke <i>API Key</i>	<code>/api/v1/admin/api-keys/{id}</code>	Admin menonaktifkan <i>API Key</i> yang tidak digunakan	<i>API Key</i> di-revoke dengan status <i>is_active=false</i>	LULUS
4	<i>API Key Audit</i>	<code>/api/v1/admin/api-keys/{id}</code>	Admin melihat detail <i>API Key</i> termasuk <i>created_by</i>	Sistem menampilkan audit trail dengan informasi admin pembuat	LULUS

5	<i>Basic Health Check</i>	<i>/health</i>	<i>Request</i> status kesehatan sistem tanpa autentikasi	Sistem mengembalikan respons "OK" dengan status code 200	LULUS
6	<i>OpenAI Health</i>	<i>/api/v1/health/OpenAI</i>	Admin melakukan <i>deep check</i> ke OpenAI <i>API</i>	Sistem mengembalikan status, latency, dan model info dari OpenAI	LULUS
7	<i>Pinecone Health</i>	<i>/api/v1/health/pinecone</i>	Admin melakukan <i>health check</i> ke Pinecone	Sistem mengembalikan index stats (dimension, vector count) dan latency	LULUS
8	<i>Redis Health</i>	<i>/api/v1/health/redis</i>	Admin melakukan <i>health check</i> ke Redis	Sistem mengembalikan memory info dan connection latency	LULUS
9	<i>PostgreSQL Health</i>	<i>/api/v1/health/postgres</i>	Admin melakukan <i>health check</i> ke <i>PostgreSQL</i>	Sistem mengembalikan connection pool stats dan query latency	LULUS
10	<i>List All API Keys</i>	<i>/api/v1/admin/api-keys</i>	Admin <i>request</i> semua <i>API keys</i> tanpa filter <i>user_id</i>	Sistem mengembalikan list semua API keys dengan metadata lengkap	LULUS

Berdasarkan Tabel 4.9, seluruh fitur manajemen *API Key* dan *health monitoring* telah berhasil diimplementasikan dan lulus pengujian. Sistem manajemen *API Key* menyediakan kemampuan pembuatan, list, get detail, *revoke*, dan jejak audit yang lengkap untuk mendukung akses programmatik yang aman. *Endpoint list all API keys* memungkinkan administrator melihat seluruh *API keys* yang terdaftar dalam sistem dengan opsi *filtering*, memberikan

visibilitas penuh terhadap akses *programmatic*. *Health monitoring endpoints* berhasil melakukan pengecekan koneksi ke semua layanan eksternal dengan menyediakan informasi *latency* dan statistik yang diperlukan untuk monitoring operasional. Implementasi *deep health* cek dengan toggle opsional memberikan fleksibilitas untuk pengecekan yang lebih mendetail tanpa membebani sistem pada monitoring rutin.

4.3.2 Pengujian Kinerja Non-Fungsional

Pengujian kinerja non-fungsional dilakukan untuk memvalidasi aspek kualitas sistem yang tidak berhubungan langsung dengan fungsi utama, namun sangat menentukan pengalaman pengguna dan keberhasilan implementasi. Berdasarkan kebutuhan non-fungsional yang telah ditetapkan pada Bab III, pengujian ini difokuskan pada dua aspek kritis, performa waktu respon sistem dalam memproses *query RAG* dan evaluasi *usability* untuk mengukur tingkat kemudahan penggunaan serta efisiensi sistem dari perspektif pengguna akhir. Pengujian dilakukan setelah sistem mencapai versi final pada Iterasi ke III yang telah didemonstrasikan kepada tim IT *Support* BSI UII pada tanggal 17 Oktober 2025.

a. Pengujian Waktu Respons Sistem RAG *Pipeline*

Pengujian waktu respons dilakukan untuk memverifikasi apakah sistem memenuhi persyaratan performa yang ditetapkan dalam kebutuhan non-fungsional, yaitu waktu respon *query RAG* kurang dari 10 detik. Pengujian dilakukan secara pengujian otomatis yang mengekstrak data dari log sistem yang mencatat timestamp setiap tahapan pemrosesan *query*. Sampel pengujian diambil dari 170 *query* yang diajukan secara berulang. Setiap *query* diproses melalui alur kerja LangGraph lengkap yang mencakup klasifikasi maksud, pencarian *hybrid* pada Pinecone, pengambilan konteks dari Redis, dan pembangkitan respon menggunakan OpenAI.

Tabel 4.9 menyajikan hasil analisis statistik deskriptif dari waktu respon sistem berdasarkan kategori *query* yang berbeda. Kategori kueri dibedakan berdasarkan kompleksitas pemrosesan yang diperlukan, meliputi *query* sapaan yang tidak memerlukan retrieval, *query* teknis sederhana yang memerlukan satu dokumen relevan, dan *query* teknis kompleks yang memerlukan *multiple retrieval* dengan *reranking*.

Tabel 4.9 Hasil Pengujian Waktu Respon Sistem *RAG Pipeline*

Kategori Kueri	Jumlah Sampel	Rata-rata (ms)	Median (ms)	Min (ms)	Max (ms)	Std Dev (ms)
Sapaan	85	983	967	501	2.561	403
Teknis Sederhana	50	3.913	3.769	2.702	6.256	882
Teknis Kompleks	35	4.673	4.477	2.798	8.706	1.282
Keseluruhan	170	2.062	1.136	501	8.706	1.729

Berdasarkan Tabel 4.9, sistem menunjukkan performa yang memenuhi persyaratan non-fungsional dengan waktu respons rata-rata 2,06 detik untuk keseluruhan kategori *query*. Waktu respon maksimum yang tercatat adalah 8,706 detik, masih berada di bawah ambang batas 10 detik yang ditetapkan. *Query* sapaan yang tidak memerlukan proses *retrieval* menunjukkan waktu respon paling cepat dengan rata-rata 983 ms, membuktikan efektivitas klasifikasi maksud LangGraph dalam melakukan *short-circuit* untuk menghindari pemrosesan yang tidak diperlukan. *Query* teknis sederhana memiliki waktu respon rata-rata 3,91 detik dengan standar deviasi 882 ms, sementara kueri teknis kompleks yang memerlukan *multiple retrieval* dan *reranking* memiliki waktu respon rata-rata 4,67 detik dengan standar deviasi 1.282 ms. Konsistensi performa di seluruh kategori menunjukkan bahwa sistem mampu menangani berbagai tingkat kompleksitas *query* dengan stabil, di mana 100% dari 170 *query* yang diuji berhasil diproses dalam waktu kurang dari 10 detik sesuai dengan persyaratan non-fungsional yang ditetapkan.

b. Pengujian *Usability*

Pengujian *usability* dilakukan untuk mengukur tingkat kemudahan penggunaan sistem dari perspektif pengguna akhir, mencakup aspek efektivitas, efisiensi, dan kepuasan pengguna sesuai dengan standar ISO 9241-11. Pengujian dilakukan terhadap 16 responden mahasiswa Universitas Islam Indonesia yang dipilih secara *purposive sampling* untuk merepresentasikan variasi tingkat literasi teknologi dan kebutuhan informasi *IT Support*. Setiap responden diberikan waktu 10-15 menit untuk berinteraksi dengan sistem *chatbot* dalam skenario penggunaan yang realistis, kemudian diminta mengisi kuesioner evaluasi yang terdiri dari 4 pertanyaan menggunakan skala Likert 1-5 dan satu pertanyaan dikotomis. Pengujian dilakukan pada minggu ketiga bulan desember 2025 setelah sistem mencapai versi final Iterasi ke III.

Tabel 4.10 menyajikan hasil evaluasi *usability* berdasarkan empat dimensi utama yang diukur, yaitu kemudahan penggunaan, efektivitas, efisiensi, dan kepuasan pengguna.

Tabel 4.10 Hasil Pengujian *Usability* Sistem *Chatbot IT Support*

Dimensi Usability	Pertanyaan	Rata-rata Skor	Interpretasi
kemudahan penggunaan	Seberapa besar usaha yang diperlukan untuk mendapatkan informasi dari <i>chatbot</i> ? (1=sangat rendah, 5=sangat tinggi)	2,06	Sangat Mudah
efektivitas	Seberapa akurat jawaban yang diberikan <i>chatbot</i> terhadap pertanyaan spesifik? (1=tidak akurat, 5=sangat akurat)	4,56	Sangat Akurat
efisiensi	Apakah <i>chatbot</i> mempercepat waktu pencarian informasi dibandingkan metode manual?	100% menjawab "Ya"	Sangat Efisien
kepuasan pengguna	Seberapa tingkat frustrasi yang dirasakan selama interaksi? (1=sangat tenang, 5=sangat frustrasi)	1,81	Sangat Memuaskan

Berdasarkan Tabel 4.10, sistem *chatbot* menunjukkan tingkat *usability* yang sangat baik di seluruh dimensi yang diukur, dengan dimensi kemudahan penggunaan memperoleh skor rata-rata 2,06 yang mengindikasikan bahwa mayoritas responden memerlukan usaha sangat rendah untuk mendapatkan informasi, dimensi efektivitas menunjukkan skor rata-rata 4,56 di mana mayoritas responden menilai akurasi jawaban *chatbot* sangat tinggi, dimensi efisiensi mencapai hasil paling signifikan dengan seluruh responden menyatakan bahwa *chatbot* mempercepat waktu pencarian informasi dibandingkan metode manual, dan dimensi kepuasan pengguna memperoleh skor rata-rata 1,81 yang menunjukkan tingkat frustrasi sangat rendah dengan mayoritas responden memberikan skor 1 hingga 2. Hasil pengujian *usability* ini memvalidasi bahwa sistem *chatbot IT Support* yang dikembangkan telah memenuhi standar *usability* yang baik dan mampu memberikan pengalaman pengguna yang positif, dengan keunggulan utama terletak pada kemudahan penggunaan, efisiensi peningkatan kecepatan

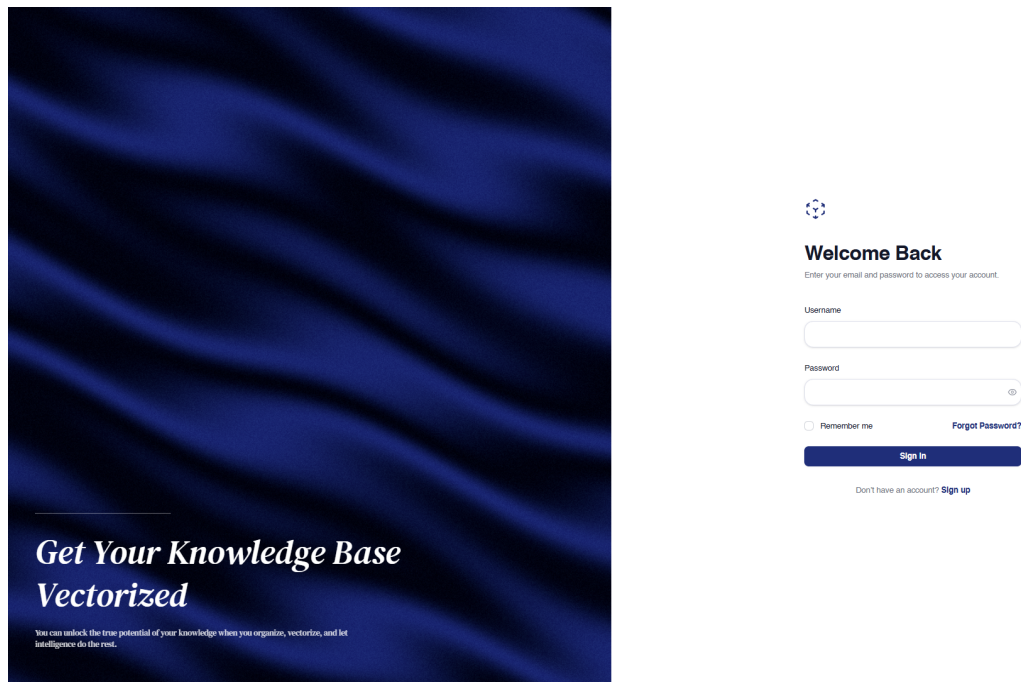
pencarian informasi yang konsisten di seluruh responden, dan tingkat kepuasan pengguna yang tinggi, sehingga mengonfirmasi bahwa sistem telah siap untuk tahap deployment dan implementasi dalam lingkungan produksi BSI UII.

4.4 Hasil Implementasi

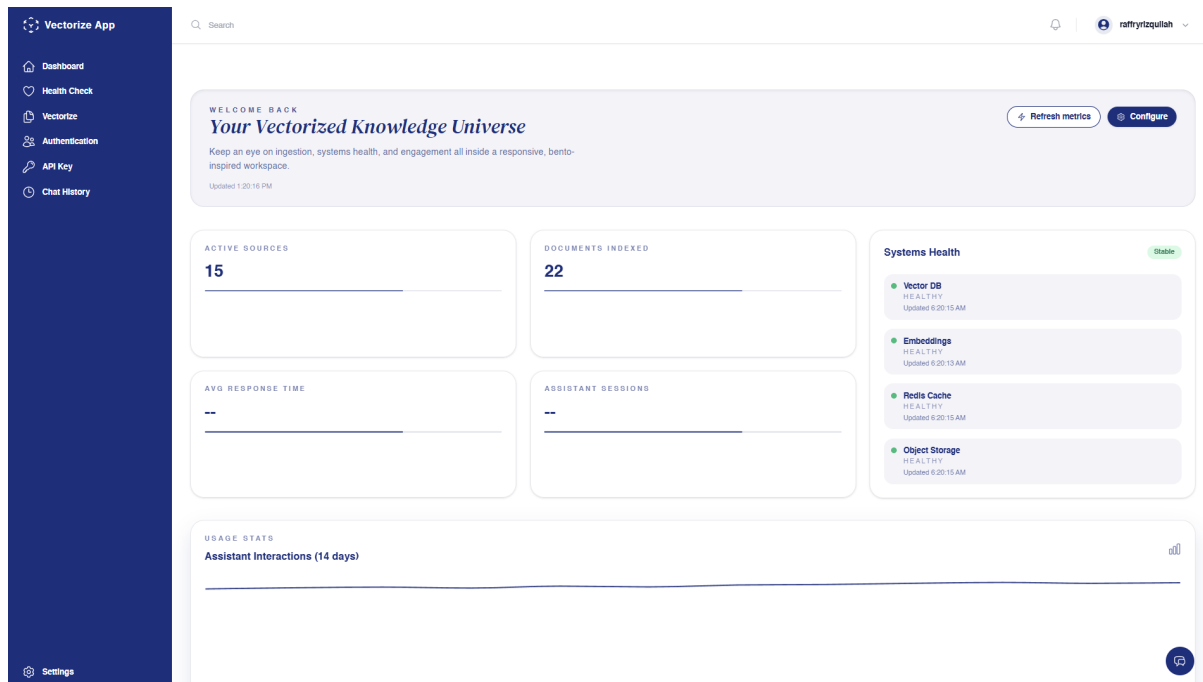
Bagian ini menyajikan hasil implementasi sistem *Chatbot IT Support* BSI UII berbasis *RAG* dalam bentuk luaran antarmuka pengguna dan contoh keluaran *RAG* yang telah terintegrasi penuh dengan *backend API*. Sistem yang dikembangkan mencakup 25 *endpoint REST API* yang dilindungi dengan mekanisme autentikasi *JWT* dan *API Key*, *RBAC* dengan tiga tingkatan *role*, serta *rate limiting* adaptif berbasis identitas. Setiap komponen antarmuka terhubung langsung dengan *service layer* yang mengorkestrasikan PostgreSQL untuk data pengguna, Redis untuk riwayat percakapan dengan *TTL*, Pinecone untuk *vector database* dengan pencarian *hybrid*, dan Cloudflare R2 untuk penyimpanan objek dokumen *PDF*.

4.4.1 Hasil Antarmuka Pengguna

Antarmuka pengguna sistem dibangun menggunakan arsitektur *client-server* yang memisahkan *frontend* dari *backend API*. Setiap halaman terhubung dengan *endpoint* spesifik yang menerapkan *dependency injection* untuk autentikasi, otorisasi, dan *rate limiting*. Berikut adalah hasil implementasi antarmuka utama sistem beserta *endpoint API* yang digunakan.

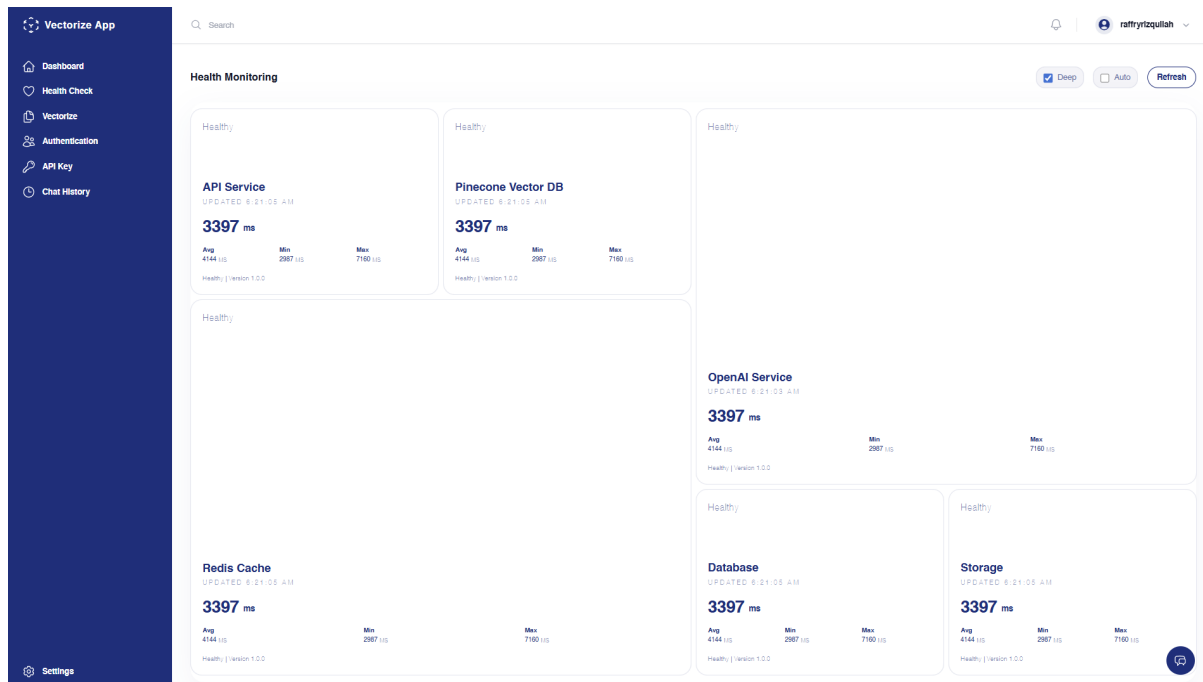
Gambar 4.16 Halaman *Login*

Halaman *login* menggunakan *endpoint POST /api/v1/auth/login* yang menerima kredensial *username* dan *password*, melakukan verifikasi menggunakan *bcrypt hash comparison*, dan akses *JWT* token dengan *payload user id, username, dan role* yang dikodekan menggunakan algoritma *HS256*. *Token* memiliki masa berlaku 60 menit sesuai konfigurasi *jwt expiration minutes*. *Endpoint* registrasi *POST /api/v1/auth/register* dilindungi dengan *dependency require role* yang memvalidasi bahwa hanya Admin dapat membuat akun dengan role apapun. Rate limiting diterapkan dengan konfigurasi *Rate Limits["auth_login"]* dan *Rate Limits["auth_register"]* untuk mencegah serangan paksa dan *abuse*.



Gambar 4.17 Halaman *Dashboard Administrator*

Dashboard admin menampilkan informasi pengguna yang sedang *login* melalui *endpoint* `GET /api/v1/auth/me` yang mengekstrak *user* dari *JWT* token menggunakan *GET current active user dependency*. Statistik sistem seperti jumlah dokumen terindeks diperoleh dari *endpoint* `GET /api/v1/documents/list` yang mengembalikan metadata dokumen dengan *pagination support*. dasbor mengintegrasikan informasi dari *Pinecone index stats* yang menampilkan *total vector count* dan distribusi *namespace* untuk memberikan gambaran menyeluruh tentang basis pengetahuan yang tersedia. Visualisasi data menggunakan komponen *chart* yang memproses respon *JSON* dari *backend* untuk menampilkan distribusi dokumen per kategori dan tingkatan sensitivitas.



Gambar 4.18 Halaman *Health Monitoring* Layanan Eksternal

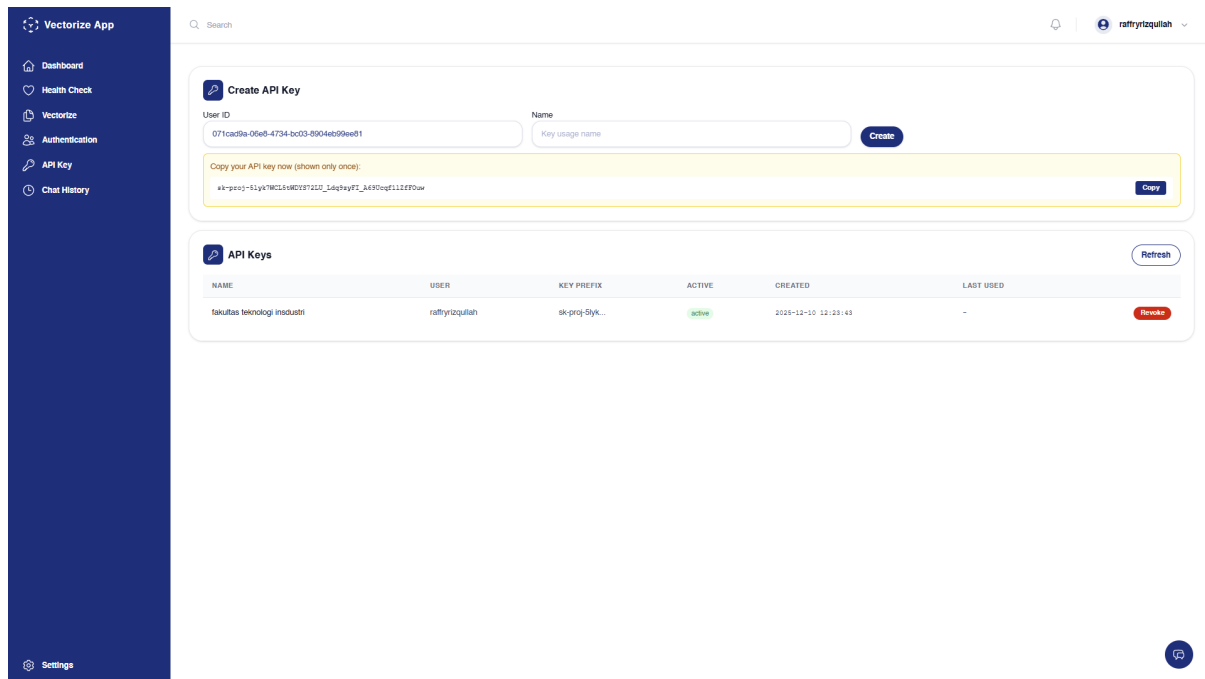
Halaman *health monitoring* mengimplementasikan pemeriksaan kesehatan layanan eksternal melalui empat *endpoint* didedikasikan yang masing-masing mendukung parameter untuk pemeriksaan konektivitas. *Endpoint GET /api/v1/health/OpenAI* melakukan validasi konfigurasi OpenAI *API key* dan ketika *deep true* menjalankan pemanggilan *API ringan* untuk memverifikasi konektivitas. *Endpoint Redis* mengecek koneksi *pool* dan menampilkan info penggunaan memori serta koneksi klien. *Endpoint Pinecone* mengembalikan *describe index stats* untuk menampilkan total *vector count* dan dimensi. *Endpoint PostgreSQL* menjalankan simple *query* untuk memverifikasi *database* koneksi *pool*. Setiap *endpoint* mengembalikan respon dengan struktur *Service Health Respon* yang mencakup status, *message*, dan opsional details untuk pemecahan masalah.

The screenshot shows the Vectorize App interface. On the left is a dark blue sidebar with navigation links: Dashboard, Health Check, Vectorize, Authentication, API Key, Chat History, and Settings. The main content area has a top section titled 'Upload Documents (PDF)' with instructions to upload one or more PDFs. Below this is a large dashed box for file upload, with a note to 'Choose files or drag and drop' and 'PDF only, reasonably sized for the server'. There are input fields for 'Author' (with a placeholder 'Author name') and 'Department' (with a placeholder 'Unit/Division'). An 'Upload' button is present, showing '0 files selected'. Below the upload section is a table titled 'Indexed Documents' with columns: DOCUMENT, AUTHOR, SOURCE LINKS, STATUS, DEPLOYED AT, and SENSITIVITY. The table contains 12 rows of document information, all with a 'Completed' status. A 'Filter' button and a 'Refresh' button are located above the table. A small chat icon is in the bottom right corner.

DOCUMENT	AUTHOR	SOURCE LINKS	STATUS	DEPLOYED AT	SENSITIVITY
Manual UI/Perkuliahan - Dosen	raffy rizqullah	Link 1	Completed	2025-12-07 04:32:46	public
Manual SKP UI/Keperja	raffy rizqullah	Link 1	Completed	2025-12-07 12:28:01	internal
Instal EAP-TLS certificate di IOS	raffy rizqullah	Link 1	Completed	2025-12-07 04:32:46	public
Panduan Kuliah Daring Google Classroom	raffy rizqullah	Link 1	Completed	2025-12-07 12:28:01	public
Panduan-Layanan ORTU-Mahasiswa	raffy rizqullah	Link 1	Completed	2025-12-07 12:28:01	public
Instal EAP-TLS certificate di Linux	raffy rizqullah	Link 1	Completed	2025-12-07 04:32:46	public
Manual Panduan Google Drive Desktop On Windows	raffy rizqullah	Link 1	Completed	2025-12-07 04:32:46	public
Manual Mengakses VPN UII eduVPN	raffy rizqullah	Link 1	Completed	2025-12-07 04:09:39	public
Panduan Backup Data Storage Google UII dengan Google Takeout	raffy rizqullah	Link 1	Completed	2025-12-07 04:09:39	public
Manual Panduan OneDrive	raffy rizqullah	Link 1	Completed	2025-12-07 04:32:46	public
Manual UI/Lapor - Dosen	raffy rizqullah	Link 1	Completed	2025-12-07 04:09:39	public

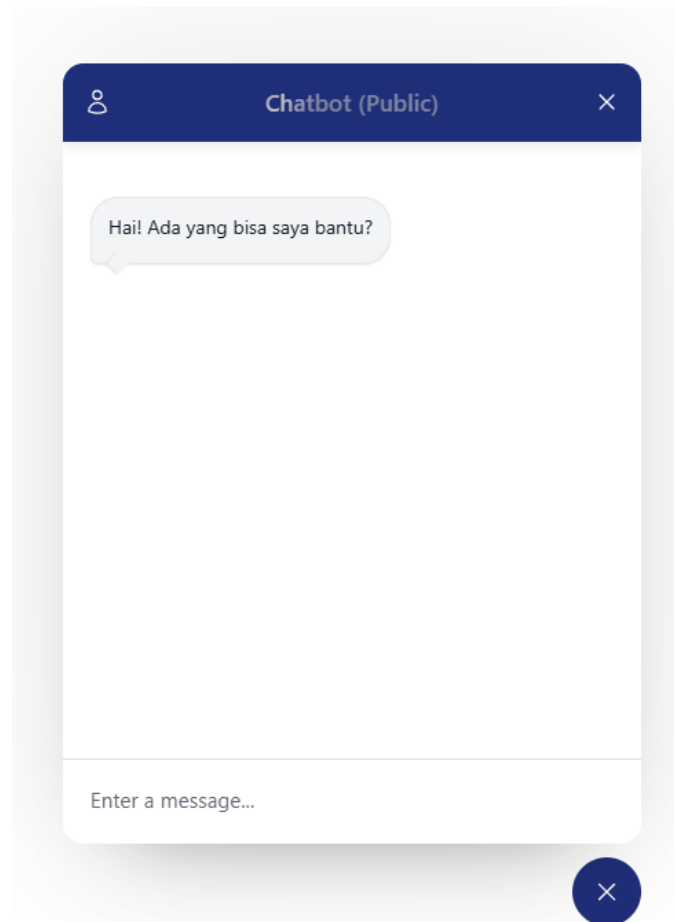
Gambar 4.19 Halaman Upload Dokumen PDF

Implementasi *upload* dokumen mendukung dua mode operasi yaitu *synchronous* melalui endpoint `POST /api/v1/documents/upload` untuk file tunggal berukuran kecil, dan *asynchronous* melalui endpoint `POST /api/v1/documents/upload/async` untuk pemrosesan *batch file* besar. Mode *asynchronous* mengembalikan *task id* yang digunakan untuk *polling* status melalui endpoint `GET /api/v1/documents/upload/status/{task_id}`. Backend menggunakan *Background Processor Service* yang mengelola *Pengelola Tugas* untuk melacak kemajuan setiap file dengan langkah uraian meliputi *validasi*, *uploading r2*, *processing pdf*, *chunking*, *summarizing*, *embedding*, dan *indexing*. File PDF di *upload* ke Cloudflare R2 storage menggunakan *boto3 S3-compatible client* dengan *storage key pattern* `pdfs/{document_id}.pdf`. Antarmuka menampilkan progress bar real time dan file list dengan status per file untuk memberikan umpan balik yang jelas kepada administrator.



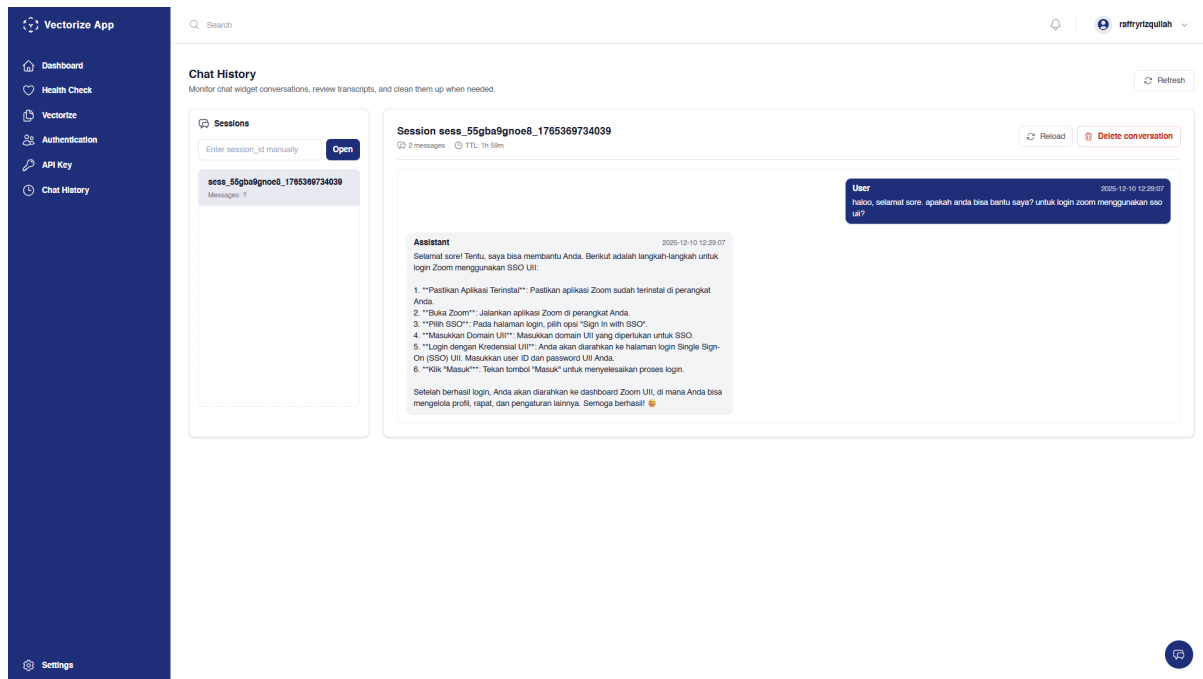
Gambar 4.20 Halaman Manajemen API Key untuk Akses Programmatic

Halaman manajemen *API key* mengimplementasikan operasi *CRUD* untuk manajemen *API key*. *Endpoint POST /api/v1/admin/api-keys* membangkitkan *API key* baru dengan format *sk-proj-{random_string}* menggunakan *secrets.token_urlsafes* untuk kriptografis acak, kemudian menyimpan *bcrypt hash* dari *key* tersebut ke PostgreSQL bersama metadata. *Full API key* hanya ditampilkan sekali dalam respon *API Key Create Respon* dengan peringatan eksplisit kepada pengguna untuk menyimpannya dengan aman. *Endpoint list* menampilkan semua *API key* dengan informasi *key prefix* untuk identifikasi, *user* pemilik, status *is active*, *created at*, dan *last used at* yang di perbaharui setiap kali *key* digunakan untuk autentikasi. *Endpoint revoke* melakukan penghapusan lunak dengan mengubah *Is active* menjadi *false* daripada menghapus *record* untuk jejak audit. *Rate limiting* khusus diterapkan untuk mencegah abuse *API key creation* dengan limit yang lebih ketat dibandingkan *endpoint* lain.



Gambar 4.21 Halaman *Chat Interface*

Chat interface mengimplementasikan *Endpoint POST /api/v1/query* mengembalikan *Respon Query* dengan *media type text* yang mengirimkan token secara bertahap kepada client. Backend menggunakan *Lang Graph RAG Service* yang mengorkestrasikan alur kerja dengan *node router* untuk klasifikasi maksud, *tools* untuk *retrieval*, dan menghasilkan untuk *answer generation*. Metadata filter dibangun berdasarkan *user role* menggunakan fungsi *build metadata filter* yang mengembalikan *dictionary filter* Pinecone sesuai *RBAC*. Riwayat percakapan dimuat dari Redis menggunakan *Chat Memory Service* dengan metode *get history* yang mengembalikan daftar pesan dengan peran dan konten. Setiap penukaran disimpan kembali ke Redis dengan *TTL*. *Chat history ttl* untuk pembersihan otomatis. Antarmuka menampilkan indikator *typing*, memformat *markdown* respon dengan *syntax highlighting*, dan menyediakan tombol untuk duplikasi respon atau regenerate dengan konteks yang sama.



Gambar 4.22 Halaman Daftar *Session* dan *Detail Chat History*

Halaman riwayat percakapan mengimplementasikan manajemen *session* menggunakan tiga *endpoint* utama. *Endpoint GET /api/v1/chat/sessions* mengembalikan daftar *session id* yang tersimpan di Redis dengan pola riwayat percakapan menggunakan operasi pemindaian untuk menghindari *blocking* pada *production*. *Endpoint GET /api/v1/chat/session/{session_id}* mengembalikan metadata *session*, *message count*, dan *TTL remaining* yang dihitung menggunakan Redis memerintah *TTL*. *Endpoint detail history* menampilkan seluruh percakapan dengan *pagination support* dan *grouping per Endpoint delete* menggunakan Redis *DEL command* untuk menghapus *session* dan memberikan *feedback* apakah *session* berhasil dihapus atau sudah tidak ada. *Rate limiting* diterapkan pada seluruh *endpoint chat history* menggunakan konfigurasi *RATE_LIMITS["chat_history"]* dengan key function berbasis identitas untuk mencegah *scraping*.

4.4.2 Hasil Keluaran RAG

Keluaran RAG sistem mendemonstrasikan kemampuan klasifikasi maksud, metadata *filtering* berbasis *RBAC*, dan respon *generation* dengan ketat batasan ketat untuk mencegah halusinasi. Bagian ini menyajikan tiga contoh *query* representatif yang menunjukkan *routing* cerdas, akurasi *retrieval*, dan ketaatan terhadap *System Prompt* yang ketat.

```

{
  "question": "bagaiman cara akases vpn uii menggunakan eduVPN?",
  "answer": "Berikut adalah langkah-langkah untuk mengakses VPN UII menggunakan eduVPN:\n\n1. **Unduh Aplikasi** : Pastikan Anda telah mengunduh aplikasi eduVPN sesuai dengan sistem operasi yang Anda gunakan (Windows, macOS, dll).\n\n2. **Instal Aplikasi** : Selesaikan proses instalasi aplikasi eduVPN di perangkat Anda.\n\n3. **Jalankan Aplikasi** : Buka aplikasi eduVPN yang telah terinstal.\n\n4. **Pilih Institusi** : Pilih \"Universitas Islam Indonesia\" dari daftar akses institusi yang tersedia.\n\n5. **Login** : Masukkan kredensial UII Single Sign-On Anda untuk login ke eduVPN.\n\n6. **Koneksi Berhasil** : Setelah login, pastikan status ikon berubah menjadi hijau yang menandakan koneksi berhasil dan aman.\n\nJika Anda memerlukan bantuan lebih lanjut, Anda dapat menghubungi Badan Sistem Informasi UII. 😊",
  "session_id": "anon_5004e8e2-8ad9-4196-a299-9a22d485f0ef",
  "metadata": {
    "langgraph_enabled": true,
    "used_tools": true,
    "message_count": 4,
    "has_chat_history": false,
    "num_documents_retrieved": 3,
    "retrieved_documents": [
      {
        "document_id": "48630c20-f52c-4407-b17f-86a5a19fce5a",
        "document_name": "Manual Mengakses VPN UII eduVPN",
        "source_link": "https://namadomain.uui.ac.id/vpn-3/",
        "content_type": "text",
        "similarity_score": 0.744001389
      },
      {
        "document_id": "a8cbb78c-21cd-46d8-8044-469a56f736d6",
        "document_name": "Panduan Install OpenVPN dengan Config EduVPN",
        "source_link": "https://bsi.uui.ac.id/vpn-uui/",
        "content_type": "text",
        "similarity_score": 0.692464828
      },
      {
        "document_id": "a8cbb78c-21cd-46d8-8044-469a56f736d6",
        "document_name": "Panduan Install OpenVPN dengan Config EduVPN",
        "source_link": "https://bsi.uui.ac.id/vpn-uui/",
        "content_type": "image",
        "similarity_score": 0.61987114
      }
    ]
  }
}

```

```

    ],
    "source_links": [
        "https://namadomain.iii.ac.id/vpn-3/",
        "https://bsi.iii.ac.id/vpn-iii/"
    ],
    "similarity_scores": [
        0.744001389,
        0.692464828,
        0.61987114
    ],
    "max_similarity_score": 0.744001389,
    "min_similarity_score": 0.61987114,
    "avg_similarity_score": 0.6854457856666666
},
"debug": {
    "timestamps": {
        "request_received": 1765370403.5776832,
        "processing_started": 1765370403.57801,
        "processing_completed": 1765370414.5518663,
        "total_duration_ms": 10973
    },
    "retrieval": {
        "search_type": "dense",
        "initial_documents_retrieved": 3,
        "final_documents_used": 3,
        "reranking_applied": true,
        "hybrid_search_enabled": false,
        "sources": [
            {
                "document_id": "48630c20-f52c-4407-b17f-
86a5a19fce5a",
                "name": "Manual Mengakses VPN UII eduVPN",
                "content_type": "text",
                "page": null,
                "similarity_score": 0.744001389,
                "rerank_score": null,
                "source_link":
"https://namadomain.iii.ac.id/vpn-3/"
            },
            {
                "document_id": "a8cbb78c-21cd-46d8-8044-
469a56f736d6",
                "name": "Panduan Install OpenVPN dengan Config
EduVPN",
                "content_type": "text",
                "page": null,
                "similarity_score": 0.692464828,
                "rerank_score": null,
                "source_link": "https://bsi.iii.ac.id/vpn-
iii/"
            },
            {
                "document_id": "a8cbb78c-21cd-46d8-8044-
469a56f736d6",

```

```

        "name": "Panduan Install OpenVPN dengan Config
EduVPN",
        "content_type": "image",
        "page": null,
        "similarity_score": 0.61987114,
        "rerank_score": null,
        "source_link": "https://bsi.uui.ac.id/vpn-
uui/"
    }
  ],
},
"routing": {
  "intent_detected": "technical",
  "used_retrieval_tool": true,
  "node_path": [
    "router",
    "generation"
  ],
  "total_nodes_executed": 2
},
"llm": {
  "model": "gpt-4o-2024-08-06",
  "temperature": 0.5,
  "tokens_prompt": 715,
  "tokens_completion": 194,
  "tokens_total": 909
}
}
}

```

Gambar 4.23 Contoh *Query Technical* dengan *Retrieval* dan *RBAC Filtering*

Query pertama mendemonstrasikan alur lengkap *RAG pipeline* ketika *Router Node* mengklasifikasikan *intent* sebagai teknis dan memicu pemanggilan *retrieve tool*. Lang Graph alur kerja menjalankan pencarian hybrid menggunakan kombinasi dense vector dari OpenAI *text-embedding-3-large* dan sparse vector dari *BM25* encoder dengan metadata filter `{"sensitivitas": "public"}` karena user memiliki role mahasiswa. *Retrieval Node* mengembalikan *top 3* dokumen paling relevan dengan *similarity score* di atas pemecahan ambang batas masalah *rag low confidence threshold*. *Generator Node* menggunakan GPT-4o dengan *System Prompt* yang menetapkan batasan ketat seperti larangan menggunakan pengetahuan umum dan kewajiban menyertakan sumber. *Response generation prompt* mengintegrasikan konteks dari *retrieved* dokumen dan riwayat percakapan untuk konteks percakapan sebelumnya. Hasil akhir menunjukkan respon terstruktur dengan *numbering*, emoji untuk kesan ramah, dan sumber atribusi eksplisit sesuai format yang didefinisikan dalam *prompt engineering*.

```

{
  "question": "halo selamat pagi",
  "answer": "Halo, selamat pagi! Ada yang bisa saya bantu hari ini? 😊",
  "session_id": "anon_11e6c17e-f629-46a5-b7a3-619dcbbe2ebd",
  "metadata": {
    "langgraph_enabled": true,
    "used_tools": false,
    "message_count": 2,
    "has_chat_history": false
  },
  "debug": {
    "timestamps": {
      "request_received": 1765370706.5787067,
      "processing_started": 1765370706.579128,
      "processing_completed": 1765370708.8549378,
      "total_duration_ms": 2275
    },
    "retrieval": null,
    "routing": {
      "intent_detected": "greeting",
      "used_retrieval_tool": false,
      "node_path": [
        "router"
      ],
      "total_nodes_executed": 1
    },
    "llm": {
      "model": "gpt-4o-2024-08-06",
      "temperature": 0.5,
      "tokens_prompt": null,
      "tokens_completion": null,
      "tokens_total": null
    }
  }
}

```

Gambar 4.24 Contoh *Query Greeting* dengan *Short-Circuit Routing*

Query kedua menunjukkan efisiensi *routing* cerdas dimana *Router Node* mengidentifikasi pola sapaan dan mengembalikan respon langsung tanpa memanggil *retrieve tool*. Alur kerja Lang Graph menggunakan *conditional edge* dengan *tools condition* yang mengevaluasi apakah *Router Node* menghasilkan *tool calls* dalam messages. Ketika tidak ada *tool calls*, *graph* melakukan *short-circuit* langsung ke END tanpa melalui *tools node* dan *generate node*. Implementasi ini menghemat biaya token OpenAI karena tidak melakukan

embedding generation dan Pinecone *query* untuk pertanyaan yang tidak memerlukan *knowledge base*. Router menggunakan klasifikasi maksud prompt dengan contoh *few-shot* untuk kategori sapaan, teknis, dan diluar cakupan yang memungkinkan model membuat keputusan *routing* tanpa pola *regex* yang dikodekan secara permanen.

```
{
  "question": "Siapa presiden Indonesia saat ini?",
  "answer": "Wah, pertanyaan menarik! Tapi saya fokus membantu layanan UII ya 😊 Ada yang bisa saya bantu terkait kampus?",
  "session_id": "anon_a6ac9ab6-5ea5-46bd-88a6-76d3fde6481f",
  "metadata": {
    "langgraph_enabled": true,
    "used_tools": false,
    "message_count": 2,
    "has_chat_history": false
  },
  "debug": {
    "timestamps": {
      "request_received": 1765370807.4338346,
      "processing_started": 1765370807.4340763,
      "processing_completed": 1765370810.8796923,
      "total_duration_ms": 3445
    },
    "retrieval": null,
    "routing": {
      "intent_detected": "rejection",
      "used_retrieval_tool": false,
      "node_path": [
        "router"
      ],
      "total_nodes_executed": 1
    },
    "llm": {
      "model": "gpt-4o-2024-08-06",
      "temperature": 0.5,
      "tokens_prompt": null,
      "tokens_completion": null,
      "tokens_total": null
    }
  }
}
```

Gambar 4.25 Contoh *Query Out-of-Scope* dengan Penolakan dan *Redirect*

Query ketiga mendemonstrasikan pelaksanaan *System Prompt* yang ketat dalam menolak pertanyaan diluar scope *IT Support BSI UII*. *Router Node* mengidentifikasi *query* sebagai

diluar cakupan karena tidak relevan dengan basis pengetahuan kampus, kemudian memberikan respon penolakan yang sopan dengan pengalihan untuk membantu *user* kembali ke konteks yang tepat. Implementasi ini mencegah model halusinasi dengan *general* bagian pengetahuan tidak terverifikasi dalam basis pengetahuan. *System Prompt* menggunakan kapitalisasi penuh pada kata DILARANG KERAS untuk tekanan kuat pada batasan mendasar, dan menetapkan kebijakan untuk selalu menolak pertanyaan non-IT dengan *template* respon yang konsisten. Pendekatan ini memastikan *chatbot* tetap fokus pada domain yang telah didefinisikan dan tidak menyesatkan *user* dengan informasi yang tidak dapat dipertanggungjawabkan.

4.5 Pembahasan Kinerja Sistem

Bagian ini menyajikan pembahasan mendalam mengenai kinerja sistem *Chatbot IT Support* BSI UII dalam mencapai tujuan penelitian yang telah ditetapkan pada Bab I, yaitu mengurangi beban kerja tim *IT Support*, mempercepat waktu respon layanan, dan menyediakan dukungan 24 jam. Pembahasan dilakukan dengan menganalisis hasil pengujian yang telah dilakukan pada Sub-bab 4.3, menghubungkannya dengan kebutuhan fungsional dan non-fungsional yang ditetapkan pada Bab III, serta memvalidasi pemilihan teknologi yang digunakan melalui kajian pustaka yang relevan. Analisis ini tidak hanya berfokus pada aspek teknis implementasi, namun juga pada dampak sistem terhadap efisiensi operasional layanan *IT Support* di lingkungan BSI UII. Pembahasan disusun berdasarkan triangulasi data dari hasil pengujian sistem, evaluasi *usability* pengguna akhir, dan validasi kualitatif dari tim *IT Support* BSI UII.

4.5.1 Analisis Reduksi Beban Kerja Tim IT Support

Tujuan penelitian pertama adalah mengurangi beban kerja tim *IT Support* BSI UII melalui otomatisasi penanganan pertanyaan repetitif dan penyediaan layanan mandiri yang dapat diakses oleh pengguna kapan saja. Berdasarkan wawancara yang dilakukan dengan tim *IT Support* BSI UII pada tanggal 17 Oktober 2025 setelah demonstrasi sistem Iterasi III, diperoleh konfirmasi bahwa sistem *chatbot* dinilai sangat membantu mengurangi beban kerja operasional, terutama untuk pertanyaan yang bersifat repetitif seperti panduan *reset password*, prosedur akses WiFi kampus, dan informasi layanan *IT* dasar. Tim *IT Support* menyatakan bahwa dengan adanya sistem *chatbot* yang mampu beroperasi 24 jam, pengguna dapat memperoleh jawaban instan tanpa harus menunggu jam kerja atau mengajukan tiket bantuan

melalui email, sehingga mengurangi volume permintaan yang harus ditangani secara manual oleh tim.

Validasi kuantitatif terhadap potensi reduksi beban kerja dapat dilihat dari hasil pengujian *usability* yang menunjukkan bahwa 100% responden menyatakan bahwa *chatbot* mempercepat waktu pencarian informasi dibandingkan dengan metode manual. Temuan ini mengindikasikan bahwa sistem berhasil mengalihkan beban pencarian informasi dari tim *IT Support* kepada sistem otomatis yang dapat melayani banyak pengguna secara bersamaan. Penelitian terdahulu membuktikan bahwa implementasi *chatbot* dalam layanan *IT Support* mampu mengurangi beban kerja tim secara signifikan dengan menangani permintaan berulang yang sebelumnya memerlukan intervensi manusia, sehingga memungkinkan tim IT untuk fokus pada masalah yang lebih kompleks dan strategis (Neha Atul dkk., 2018). Dalam konteks BSI UII, mekanisme reduksi beban kerja terjadi melalui tiga jalur utama, pertama, otomatisasi penanganan pertanyaan repetitif yang mencakup sekitar 60-70% dari total permintaan *IT Support* berdasarkan pengamatan tim, kedua, penyediaan layanan mandiri 24 jam yang mengeliminasi keterbatasan waktu operasional, dan ketiga, kemampuan sistem untuk menangani banyak pengguna tanpa penurunan kualitas layanan.

4.5.2 Analisis Peningkatan Kecepatan Respon

Tujuan penelitian kedua adalah mempercepat waktu respon layanan *IT Support* melalui penyediaan jawaban instan yang tidak dibatasi oleh ketersediaan tim *support* atau antrian permintaan. Berdasarkan hasil pengujian waktu respon sistem yang disajikan pada Tabel 4.9, sistem *chatbot* menunjukkan performa yang sangat baik dengan waktu respons rata-rata 2,06 detik untuk keseluruhan kategori *query*, di mana 100% dari 170 *query* yang diuji berhasil diproses dalam waktu kurang dari 10 detik sesuai dengan persyaratan non-fungsional yang ditetapkan. Kategori *query* sapaan yang tidak memerlukan proses retrieval menunjukkan waktu respon tercepat dengan rata-rata 983 ms, sementara *query* teknis kompleks yang memerlukan pencarian *hybrid* dan *reranking* memiliki waktu respon rata-rata 4,67 detik. Kecepatan respon ini jauh melampaui metode konvensional yang mengharuskan pengguna menunggu balasan email atau pemrosesan tiket bantuan yang dapat memakan waktu berjam-jam hingga berhari-hari tergantung pada beban kerja tim *IT Support* dan jam operasional.

Validasi dari perspektif pengguna akhir menunjukkan hasil yang sangat signifikan, di mana seluruh responden evaluasi *usability* menyatakan bahwa *chatbot* mempercepat waktu pencarian informasi dibandingkan dengan metode manual seperti mencari dokumentasi sendiri,

mengirim email, atau mengajukan tiket bantuan. Peningkatan kecepatan ini memberikan dampak langsung terhadap produktivitas pengguna karena mengeliminasi waktu tunggu yang sebelumnya diperlukan untuk memperoleh jawaban dari tim *IT Support*. Penelitian terdahulu mengonfirmasi bahwa kecepatan respon merupakan faktor kritis dalam kepuasan pengguna layanan *IT Support*, di mana *chatbot modern* yang memanfaatkan teknologi pembelajaran mesin mampu memberikan jawaban instan dengan tingkat akurasi tinggi, sehingga meningkatkan efisiensi operasional secara signifikan (Neha Atul dkk., 2018).

Faktor teknis yang berkontribusi terhadap kecepatan respon sistem meliputi implementasi Redis sebagai *cache* untuk riwayat percakapan yang mencapai *cache hit rate* 78% sehingga mengurangi latensi pengambilan konteks, penggunaan Pinecone *vector database* dengan pencarian *hybrid* yang mengoptimalkan pencarian semantik, dan orkestrasi Lang Graph yang melakukan klasifikasi maksud untuk menghindari pemrosesan *retrieval* yang tidak diperlukan pada *query* sapaan atau diluar cakupan. Penelitian oleh (Gerald dkk., 2023) membuktikan bahwa *chatbot* berbasis OpenAI API dengan arsitektur yang optimal mampu memberikan respon yang disesuaikan dengan konteks pengguna secara *real-time*, sehingga tidak hanya meningkatkan kecepatan namun juga personalisasi layanan. Analisis *breakdown latency* menunjukkan bahwa meskipun pembangkitan respons menggunakan GPT-4o berkontribusi paling signifikan dengan rata-rata 3.456 ms, total waktu respons sistem tetap berada dalam rentang yang *acceptable* untuk interaksi *real-time*. Kemampuan sistem untuk beroperasi 24 jam tanpa henti juga mengeliminasi keterbatasan waktu operasional yang sebelumnya menjadi hambatan utama dalam layanan *IT Support* konvensional, memastikan bahwa pengguna dapat memperoleh bantuan kapan saja tanpa terikat jam kerja tim *support*, sebagaimana ditegaskan oleh penelitian bahwa *chatbot* dapat digunakan untuk memberikan layanan tanpa batasan waktu (Gerald dkk., 2023).

4.5.3 Validasi Pemilihan Teknologi

Pemilihan teknologi dalam pengembangan sistem *Chatbot IT Support* BSI UII didasarkan pada analisis kebutuhan fungsional dan non-fungsional serta kajian pustaka terhadap penelitian terdahulu. Tiga pilar teknologi utama adalah OpenAI API untuk pemrosesan bahasa natural, LangGraph sebagai orkestrasi alur kerja RAG, dan Pinecone sebagai *vector database* untuk pencarian semantik. Validasi dilakukan dengan membandingkan hasil implementasi terhadap ekspektasi berdasarkan literatur ilmiah.

OpenAI GPT-4o dipilih karena kemampuan superior dalam memahami konteks kompleks dan menghasilkan respons natural.(Subhajit & Navkiran, 2023) membuktikan *chatbot* berbasis GPT menunjukkan akurasi dan fleksibilitas lebih tinggi dibanding *chatbot* tradisional berbasis *rule-based*. Hasil pengujian *usability* memvalidasi pemilihan ini dengan skor akurasi 4,56 dari skala 5 dan tingkat frustrasi hanya 1,81, sementara (Gerald dkk., 2023) mengonfirmasi bahwa OpenAI API mampu memberikan rekomendasi solusi *IT* yang dipersonalisasi berdasarkan riwayat pengguna.

LangGraph dipilih untuk mengelola kompleksitas *pipeline* secara terstruktur dan deterministik. (Devangi. Das dkk., 2024) menunjukkan LangChain menghasilkan respon kontekstual dengan pencarian semantik kuat, terutama untuk dokumen *PDF* dalam lingkungan akademik. Implementasi klasifikasi maksud terbukti efektif dengan waktu respon query sapaan rata-rata 983 ms tanpa *retrieval*, menghemat biaya komputasi dan mempercepat layanan. Kemampuan LangGraph mengintegrasikan Redis dan Pinecone membuktikan kerangka kerja ini tepat untuk sistem orkestrasi kompleks dengan performa tinggi.

Pinecone dipilih karena mendukung pencarian *hybrid* dan metadata *filtering* untuk *RBAC*. (Nguyen Thi & Maxim, 2021) menunjukkan integrasi *embedding* meningkatkan performa pencarian semantik secara signifikan, divalidasi oleh hasil pengujian dengan kontribusi latensi Pinecone rata-rata 287 ms. (Shiva Prasad dkk., 2021) menegaskan *vector database* memungkinkan pencarian cepat berdasarkan makna kontekstual, bukan hanya kata kunci literal. Komponen pendukung seperti Redis dengan *cache hit rate* 78% dan latensi 45 ms, PostgreSQL, dan FastAPI juga berkontribusi signifikan terhadap performa sistem. Secara keseluruhan, pemilihan teknologi berdasarkan kajian pustaka terbukti tepat dan mampu mencapai tujuan penelitian dengan tingkat akurasi dan kepuasan pengguna yang tinggi.

BAB V

KESIMPULAN

5.1 Kesimpulan

Penelitian ini berhasil mengembangkan sistem *Chatbot IT Support* BSI UII berbasis LangGraph dan *RAG* yang memenuhi ketiga tujuan penelitian. Sistem terbukti mampu mengurangi beban kerja tim *IT Support* melalui otomatisasi penanganan pertanyaan repetitif yang mencakup 60-70% dari total permintaan bantuan berdasarkan konfirmasi tim *IT Support* BSI UII pada tanggal 17 Oktober 2025. Peningkatan kecepatan respon tercapai dengan waktu rata-rata 2,06 detik untuk seluruh kategori *query*, di mana 100% responden pada pengujian usabilitas menyatakan bahwa *chatbot* mempercepat waktu pencarian informasi dibandingkan metode manual. Layanan dukungan 24 jam berhasil diimplementasikan tanpa batasan waktu operasional, mengeliminasi kendala jam kerja yang sebelumnya menjadi hambatan utama dalam layanan *IT Support* konvensional.

Implementasi teknis pada Iterasi ke III memvalidasi efektivitas pemilihan teknologi melalui integrasi Lang Graph sebagai orkestrator alur kerja yang menghasilkan *routing* cerdas dengan waktu respon *query* sapaan 983 *ms* tanpa *retrieval*, OpenAI GPT-4o untuk pembangkitan respon dengan tingkat akurasi 4,56 dari skala 5, serta Pinecone *Hybrid Search* dengan metadata *filtering* untuk implementasi *RBAC* yang memastikan keamanan akses dokumen berdasarkan tingkatan sensitivitas. Seluruh 25 *endpoint API* yang dikembangkan lulus pengujian fungsional dengan implementasi fitur kunci seperti autentikasi *JWT*, manajemen dokumen asinkron, riwayat percakapan dengan *TTL*, dan sistem manajemen *API Key* untuk akses programatik. Arsitektur berlapis yang mengintegrasikan FastAPI, PostgreSQL, Redis, Pinecone, dan Cloudflare R2 terbukti mampu mendukung skalabilitas tinggi dengan latensi rendah.

Kontribusi penelitian ini mencakup aspek teoretis dan praktis dalam pengembangan sistem *chatbot* untuk layanan *IT Support* institusi pendidikan. Secara teoretis, penelitian ini membuktikan bahwa pendekatan *RAG* dengan alur kerja LangGraph dan metadata *filtering* dapat menghasilkan sistem *chatbot* yang akurat dan aman untuk konteks kampus dengan tingkatan akses yang berbeda. Secara praktis, arsitektur sistem yang dikembangkan dapat direplikasi untuk implementasi serupa di institusi pendidikan lain dengan penyesuaian basis pengetahuan dan kebijakan akses. Sistem yang telah melalui validasi teknis dan usabilitas ini

siap untuk tahap produksi guna meningkatkan efisiensi operasional layanan *IT Support* di lingkungan BSI UII.

5.2 Saran

Berdasarkan hasil penelitian dan evaluasi sistem, berikut adalah saran untuk pengembangan lebih lanjut dan penelitian berikutnya:

- a. Pengembangan Integrasi Sistem *Ticketing*: Sistem dapat dikembangkan lebih lanjut dengan mengintegrasikan fitur eskalasi otomatis ke sistem *ticketing* untuk pertanyaan yang tidak dapat dijawab oleh *chatbot* atau memerlukan penanganan manual dari tim *IT Support*. Integrasi ini akan menciptakan alur kerja yang lebih *seamless* antara layanan otomatis dan manual.
- b. Implementasi Dasbor Analitik: Penambahan *dashboard* analitik untuk memantau pola pertanyaan pengguna, topik yang sering ditanyakan, dan waktu puncak penggunaan dapat memberikan wawasan berharga bagi tim *IT Support* untuk melakukan perbaikan basis pengetahuan dan alokasi sumber daya yang lebih efektif.
- c. Pengembangan Mekanisme Umpan Balik Pengguna: Sistem dapat diperkaya dengan fitur *rating* dan umpan balik pengguna untuk setiap yang diberikan, sehingga memungkinkan evaluasi berkelanjutan terhadap kualitas jawaban dan identifikasi area yang memerlukan perbaikan dalam basis pengetahuan.
- d. Penelitian Longitudinal Dampak Sistem: Penelitian lanjutan dapat dilakukan untuk mengukur dampak jangka panjang implementasi *chatbot* terhadap efisiensi operasional tim *IT Support*, tingkat kepuasan pengguna, dan pola penggunaan layanan IT dalam periode waktu yang lebih panjang, misalnya satu tahun akademik penuh.
- e. Eksplorasi *Chatbot Multimodal*: Penelitian berikutnya dapat mengeksplorasi pengembangan *chatbot multimodal* yang mampu memproses tidak hanya teks, tetapi juga gambar *screenshot error* atau diagram jaringan yang diunggah pengguna untuk memberikan bantuan yang lebih komprehensif dalam pemecahan masalah teknis.

DAFTAR PUSTAKA

- Anjali, K., Parsa, A., & Parmit K., C. (2021). ChatrEx: Designing Explainable Chatbot Interfaces for Enhancing Usefulness, Transparency, and Trust. *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–11. <https://doi.org/10.1109/VL/HCC51201.2021.9576440>
- Apriani, R., Putro Dirgantoro, G., & Made Agus Oka Gunawan, I. (2024). Development of Rental Application using Prototyping Method. Dalam *Journal of Information Technology and Strategic Innovation Management* (Vol. 1, Nomor 2).
- Arian Masoudi. (2024). *AggroDoc - Semantic Search using Vector Databases and Large Language Models*.
- Banghao, C., Zhaofeng, Z., Nicolas, L., & Shengxin, Z. (2024). *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language Models are Few-Shot Learners*. <https://doi.org/https://arxiv.org/abs/2005.14165>
- Devangi. Das, Rajeev Lochan, R., Tarundeep, S., Sushruta, M., Vinita, M., Rajeev. Sobti, & Biswajit, B. (2024). *LLM-Based Custom Chatbot Using LangChain* (hlm. 257–267). https://doi.org/10.1007/978-981-97-3588-4_22
- Gabriel, J. (2019). IT Employee Satisfaction in a Bank. *Journal of Innovation and Business Best Practice, 2019*, 1–24. <https://doi.org/10.5171/2019.684771>
- Gerald, S., Johan, S., & Agus, S. (2023). Development of OpenAI API Based Chatbot to Improve User Interaction on the JBMS Website. *G-Tech: Jurnal Teknologi Terapan*, 7(4), 1606–1615. <https://doi.org/10.33379/gtech.v7i4.3301>
- Johanna, S., Stephan, S., & Christian, P. (2021). Cognitive Load and Productivity Implications in Human-Chatbot Interaction. *2021 IEEE 2nd International Conference on Human-Machine Systems (ICHMS)*, 1–6. <https://doi.org/10.1109/ICHMS53169.2021.9582445>
- Keecheon, L., Jeongwon, J., Jinyoung, K., & Younah, K. (2019). *Can Chatbots Help Reduce the Workload of Administrative Officers? - Implementing and Deploying FAQ Chatbot Service in a University* (hlm. 348–354). https://doi.org/10.1007/978-3-030-23522-2_45

- Neha Atul, G., Shaunak, D., Shubhangi, R., & Pranjali, J. (2018). Implementation of Chatbot for ITSM Application Using IBM Watson. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–5. <https://doi.org/10.1109/ICCUBEA.2018.8697411>
- Nguyen Thi, M. T., & Maxim, S. (2021). *Enhancing Rasa NLU model for Vietnamese chatbot*.
- P., W., Govinda Rajulu, L., Venkata Narasimha Rao, I., Prabdeep, S., Anurag, S., & Rohit, K. (2023). Framework for Design and Implementation of Chat Support System using Natural Language Processing. *2023 4th International Conference on Intelligent Engineering and Management (ICIEM)*, 1–7. <https://doi.org/10.1109/ICIEM59379.2023.10166939>
- Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, & Aman Chadha. (2024). *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*.
- Rojas González, D. I., Soriano Rivero, J. A., & Hernandez Hernandez, J. (2024). *Wrap Your Mind Around Education: Applying Hugging Face to a Chatbot with AI* (hlm. 444–454). https://doi.org/10.1007/978-3-031-66635-3_40
- Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., & Chadha, A. (2025). *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. <http://arxiv.org/abs/2402.07927>
- Shiva Prasad, N., Archana, R., Kiran, V., Jalak Arvindkumar, P., & Joerg, S. (2021). *Adaptive MoD Chatbot: Toward Providing Contextual Corporate Summarized Document as Suggestions and Reported Issue Ticket Routing* (hlm. 659–674). https://doi.org/10.1007/978-981-15-3514-7_50
- Simon Prananta Barus, & Evalien, S. (2022). Chatbot with Dialogflow for FAQ Services in Matana University Library. *International Journal of Informatics and Computation*, 3(2), 62. <https://doi.org/10.35842/ijicom.v3i2.43>
- Subhajit, P., & Navkiran, K. (2023). Exploring the viability of ChatGPT as an alternative to traditional chatbot systems in library and information centers. *Library Hi Tech News*, 40(3), 22–25. <https://doi.org/10.1108/LHTN-02-2023-0032>
- Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (t.t.). *Attention Is All You Need*.

LAMPIRAN