BAB III

PELAKSANAAN PROYEK

3.1 Kedudukan dan Koordinasi

Pelaksanaan Proyek Independen MBKM ini melibatkan tim yang terdiri dari tiga mahasiswa, yaitu Samuel Theodore Chandra, James Anderson, dan saya sendiri, Ayodhya Rifelino. Koordinasi dan struktur kerja tim disusun secara hierarkis untuk memastikan efektivitas dalam proses pengembangan proyek chatbot layanan informasi berbasis web di lingkungan Universitas Multimedia Nusantara.

Tabel 3.1 pembagian peran antar anggota

No.	Nama Anggota	Jabatan/Peran	Deskripsi Tugas Utama
1	Samuel Theodore Chandra	Koordinator/LLM Lead	Menyusun alur kerja, mengatur koordinasi tim, memilih dan menerapkan model LLM
2	James Anderson	RAG Developer	Menyusun pipeline RAG, membersihkan data handbook, menyusun vektor dan basis data
3	Ayodhya Rifelino	User Interface	Mendesain antarmuka pengguna dengan Streamlit,

			integrasi frontend dengan backend LLM & RAG

Pembimbing, baik akademik maupun lapangan, berada pada posisi tertinggi dalam struktur koordinasi. Mereka memberikan arahan, evaluasi, dan masukan secara berkala terkait kemajuan dan kualitas proyek. Komunikasi dengan pembimbing dilakukan secara rutin melalui pertemuan daring dan luring, menggunakan platform seperti Google Meet dan WhatsApp.

Samuel Theodore Chandra menjabat sebagai koordinator tim dan bertanggung jawab pada pengembangan model Large Language Model (LLM). Sebagai ketua tim, Samuel mengatur distribusi tugas antar anggota, memimpin diskusi teknis, serta menjaga agar timeline proyek berjalan sesuai rencana. Dalam aspek teknis, ia fokus memilih, mengimplementasikan, dan menguji model LLM yang digunakan dalam proyek ini, termasuk menyusun struktur prompt yang sesuai untuk kebutuhan chatbot. Samuel juga melakukan tuning terhadap model agar dapat merespons dengan konteks yang akurat dan relevan.

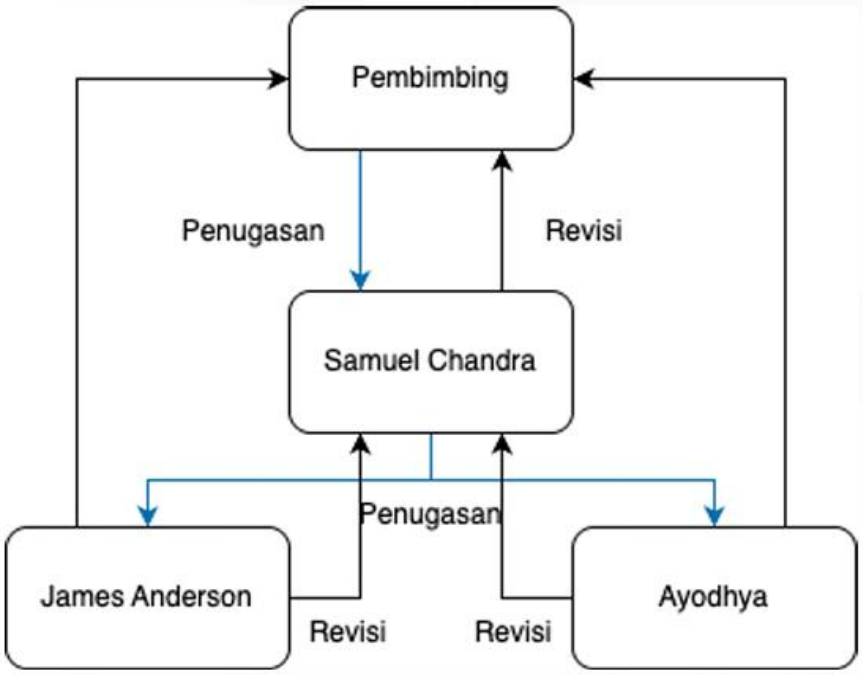
Saya, Ayodhya, bertanggung jawab dalam pengembangan antarmuka pengguna menggunakan Streamlit. Fokus utama saya adalah merancang tampilan UI yang interaktif, mengatur alur input-output pengguna, serta mengintegrasikan sistem frontend dengan backend yang terdiri dari RAG dan LLM. Selain itu, saya juga menyusun dokumentasi teknis terkait UI, melakukan debugging, dan memastikan sistem berjalan lancar tanpa gangguan saat digunakan oleh pengguna akhir.

James Anderson fokus dalam penerapan metode Retrieval-Augmented Generation (RAG). Ia bertugas mengembangkan sistem retrieval berbasis embedding

dokumen, serta mengimplementasikan algoritma pencarian yang mampu mengakses potongan data relevan dari sumber dokumen seperti handbook UMN. James juga melakukan optimasi pipeline RAG agar hasil retrieval dapat digunakan secara efisien oleh LLM.

Koordinasi antar anggota tim dilakukan secara fleksibel namun tetap terstruktur. Setiap anggota memiliki ruang untuk memberi masukan, berdiskusi mengenai hambatan teknis, serta merevisi hasil pekerjaan berdasarkan evaluasi dari pembimbing dan feedback internal. Kami menggunakan prinsip check and balance agar seluruh hasil pekerjaan dapat saling divalidasi dan dikembangkan secara kolaboratif.

Bagan alur koordinasi berikut menggambarkan hubungan kerja antara pembimbing dan anggota tim dalam pelaksanaan proyek:



Gambar 3.1 Bagan alur koordinasi

Dengan struktur pembagian tugas yang jelas dan koordinasi yang erat, proyek ini dapat berjalan sesuai rencana, serta menghasilkan sistem chatbot yang dapat memberikan layanan informasi kampus secara relevan dan efisien.

3.2 Tugas dan Uraian Kerja

Berikut ini adalah tabel tugas dan uraian kerja mingguan yang secara khusus dikerjakan oleh saya selama proyek MBKM berlangsung:

Tabel 3.2 Timeline Pelaksaan Projek

Minggu	Proyek	Keterangan
1	Diskusi topik dan tema penelitian	Berdiskusi dengan tim untuk mencari ide proyek yang sesuai. Kami melakukan brainstorming untuk menentukan topik dan masalah yang ingin diselesaikan. Beberapa ide dicatat untuk dievaluasi lebih lanjut.
2	Diskusi topik dan tema penelitian	Berdiskusi dengan tim untuk mencari ide proyek yang sesuai. Kami melakukan brainstorming untuk menentukan topik

		dan masalah yang ingin diselesaikan. Beberapa ide dicatat untuk dievaluasi lebih lanjut.
5	Mendesain UI Chatbot	Membuat layout awal antarmuka chatbot menggunakan Streamlit sesuai kebutuhan user dan tim.
6	Pengembangan Komponen Frontend	Menambahkan fitur interaksi input pengguna, output jawaban, serta tombol navigasi.
7	Integrasi Backend ke Frontend	Menghubungkan komponen UI dengan backend RAG dan LLM agar respons muncul secara real-time.
8	Implementasi Pengujian Tampilan	Melakukan debugging dan uji coba tampilan antarmuka chatbot

		agar bebas error dan stabil.
9	Optimalisasi UI Responsif	Menyesuaikan ukuran, warna, dan alur tampilan agar antarmuka nyaman digunakan di berbagai layar.
10	Evaluasi dan Perbaikan Antarmuka	Menerima feedback dari tim dan dosen, serta melakukan penyempurnaan tampilan sesuai saran.
11	Finalisasi Tampilan Streamlit	Menyusun ulang struktur dan layout UI untuk pengujian akhir dan dokumentasi.
12	Penulisan Jurnal Ilmiah	Penulisan Jurnal ilmiah dari hasil projek independen yang dibuat

Dalam pelaksanaan Proyek Independen MBKM ini, tanggung jawab utama saya terfokus pada pengembangan sistem antarmuka chatbot berbasis web yang menjadi penghubung antara pengguna dengan model Large Language Model (LLM) serta metode Retrieval-Augmented Generation (RAG). Meskipun proyek dilaksanakan

secara berkelompok, setiap anggota tim memiliki pembagian tugas yang berbeda-beda agar proses pengerjaan lebih efektif dan spesifik.

Pada proyek ini, peran saya tidak meliputi pengembangan model LLM secara langsung maupun proses fine-tuning model. Hal-hal terkait pemilihan model, pemrosesan embedding dokumen, serta implementasi algoritma RAG lebih banyak dikerjakan oleh rekan satu tim. Fokus utama saya adalah memastikan bahwa antarmuka pengguna (UI) yang dibangun dapat bekerja dengan optimal, baik dari sisi desain, kemudahan penggunaan, maupun integrasinya dengan backend sistem.

Tugas yang saya jalankan meliputi:

- Mendesain layout halaman chatbot berbasis Streamlit agar tampil sederhana, responsif, dan mudah dipahami pengguna.
- Membuat alur interaksi pengguna, mulai dari input pertanyaan, proses pemanggilan retriever, hingga penampilan jawaban yang diberikan oleh LLM.
- Membatasi jumlah token yang diproses sistem, agar input maupun output chatbot tidak melebihi kapasitas model.
- Membuat fitur untuk menampilkan sumber dokumen (metadata) agar pengguna dapat mengetahui dari mana asal informasi yang ditampilkan.
- Melakukan uji coba antarmuka secara berulang untuk memastikan sistem berjalan lancar tanpa error atau bug.

Selain pengembangan UI, saya juga bertanggung jawab menyusun dokumentasi teknis khusus terkait antarmuka, mulai dari diagram alur sistem, kode program, hingga penjelasan cara kerja interaksi antara pengguna, retriever, dan model LLM.

3.3 Uraian Pelaksanaan Kerja

Selama menjalani Proyek Independen MBKM, saya berperan aktif dalam merancang dan membangun sistem antarmuka chatbot berbasis web yang terhubung langsung dengan model LLM dan metode Retrieval-Augmented Generation (RAG). Fokus pekerjaan saya adalah menjembatani teknologi backend yang dikembangkan oleh rekan satu tim, dengan pengalaman pengguna (user experience) yang nyaman, efisien, dan intuitif.

Pekerjaan saya dimulai dari tahap perencanaan alur sistem interaksi pengguna. Dalam tahap ini, saya membuat rancangan visual dan fungsional mengenai bagaimana pengguna akan berinteraksi dengan chatbot: bagaimana pertanyaan dimasukkan, bagaimana hasil ditampilkan, dan bagaimana konteks jawaban ditautkan ke sumber dokumen.

Proses pengembangan antarmuka dilakukan menggunakan framework **Streamlit**, yang memungkinkan pembuatan UI web secara cepat dengan Python. Saya menyiapkan tampilan input teks, pemrosesan real-time terhadap query pengguna, serta penampilan hasil berupa jawaban dan referensi dokumen yang relevan. Selain itu, saya menambahkan fitur-fitur pendukung seperti pengingat batasan jumlah karakter dan validasi input agar sistem tetap berjalan stabil.

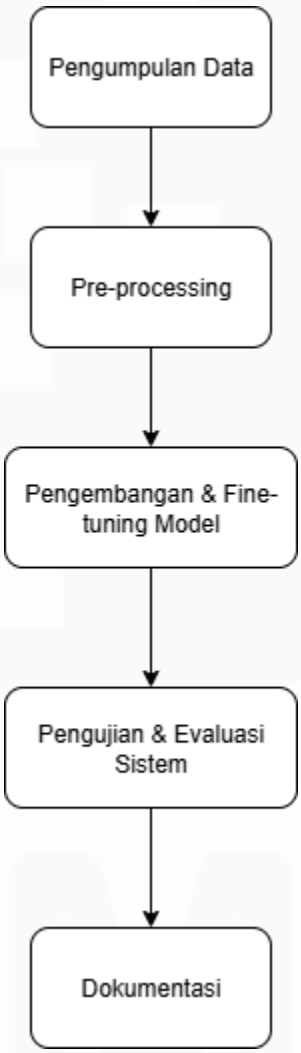
Integrasi antara antarmuka dan komponen RAG dilakukan dengan cara menyusun alur pemanggilan fungsi dari retriever dan model LLM. Saya menyesuaikan struktur prompt agar dapat menyatu secara otomatis dalam antarmuka, termasuk merancang cara menampilkan respons secara stream (mengalir) agar pengalaman pengguna lebih interaktif.

Seluruh pekerjaan ini dilengkapi dengan serangkaian pengujian. Saya melakukan simulasi berbagai jenis pertanyaan yang mungkin diajukan mahasiswa untuk memastikan bahwa sistem mampu memberikan respons yang relevan. Selain

pengujian fungsional, saya juga mengevaluasi aspek estetika dan kenyamanan UI agar sistem mudah digunakan bahkan oleh pengguna awam.

Secara keseluruhan, peran saya tidak hanya terbatas pada penulisan kode, tetapi juga mencakup penyusunan dokumentasi terkait alur kerja antarmuka, pemetaan fungsi, dan diagram integrasi antarmuka dengan sistem backend. Proses pelaksanaan dikerjakan secara iteratif dengan konsultasi rutin kepada dosen pembimbing, agar kualitas dan arah pengembangan tetap sesuai tujuan proyek.

3.3.1 Proses Pelaksanaan



Gambar 3.2 Alur Pelaksaan Projek

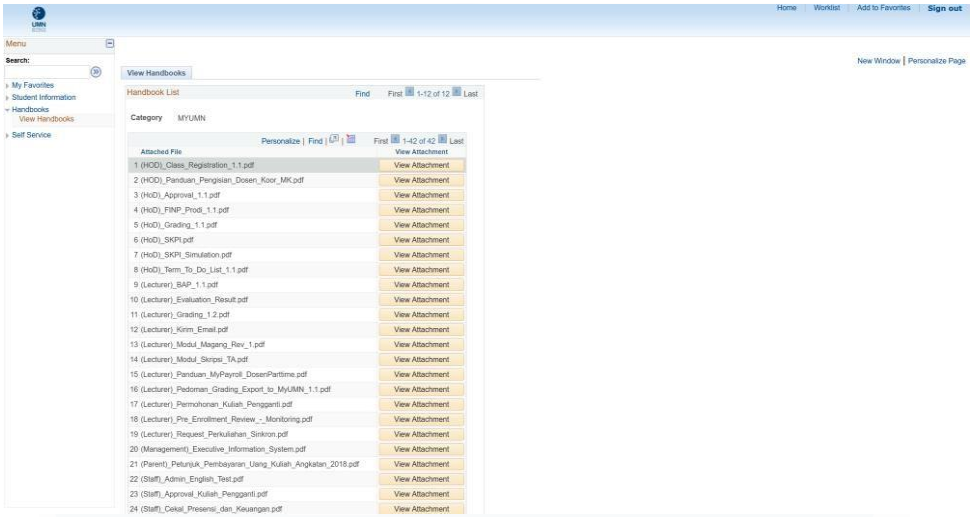
Proyek ini dijalankan secara sistematis dan terstruktur melalui sejumlah tahap yang saling terintegrasi, dengan tujuan menghasilkan sebuah chatbot berbasis teknologi Large Language Model (LLM) yang mampu memberikan layanan informasi secara

cepat, akurat, dan relevan kepada civitas akademika Universitas Multimedia Nusantara (UMN). Setiap tahapan dirancang agar mendukung keseluruhan workflow proyek, mulai dari pengumpulan data hingga implementasi sistem, serta diuji secara komprehensif untuk memastikan kualitas dan kestabilan produk akhir.

8. Pengumpulan Data

Tahap pertama difokuskan pada pengumpulan data yang menjadi sumber pengetahuan chatbot. Data ini dikumpulkan dari dokumen-dokumen resmi kampus, seperti Handbook MyUMN, serta konten dari website resmi UMN. Proses ini mencakup identifikasi dokumen yang relevan, digitalisasi dokumen dalam bentuk PDF, serta ekstraksi teks untuk diolah lebih lanjut. Tahapan ini menjadi sangat penting karena kualitas data akan memengaruhi ketepatan informasi yang diberikan chatbot kepada pengguna.

Gambar 3.3 Website Universitas Multimedia Nusantara

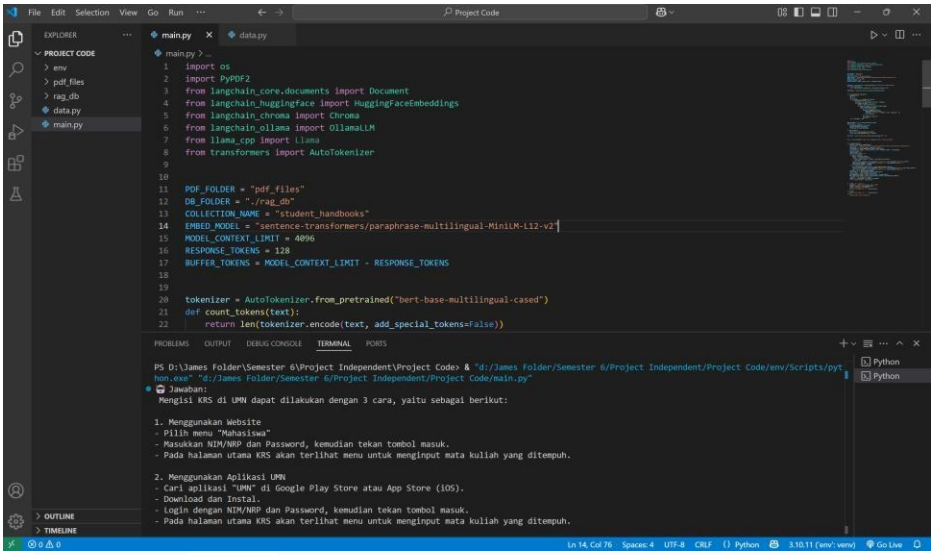


Gambar 3.4 Handbook MyUMN

9. Pre-processing Data

Tahap berikutnya adalah pre-processing, yaitu proses membersihkan dan menyiapkan data agar siap diolah oleh model. Dokumen PDF diekstraksi menjadi teks menggunakan Python karena Python memiliki pustaka (library) yang sangat kuat dalam membaca file PDF, seperti PyPDF2. Data kemudian dibersihkan dari karakter-karakter tidak penting, spasi berlebih, dan format-format yang tidak diperlukan.

Setelah teks bersih, dilakukan proses konversi menjadi vektor (embeddings). Untuk tahap ini, model MiniLM dipilih sebagai alat embedding karena lebih ringan dan cepat dibandingkan model lain seperti BERT. MiniLM memiliki ukuran parameter yang lebih kecil sehingga efisien digunakan di perangkat dengan sumber daya terbatas. Selain itu, MiniLM sudah mendukung bahasa Indonesia, sehingga lebih sesuai dengan konteks dokumen kampus.



Gambar 3.5 Koding Pengembangan Model

10. Integrasi Sistem

Langkah ketiga adalah integrasi sistem, yaitu menggabungkan berbagai komponen agar chatbot dapat bekerja secara menyeluruh. Model LLaMA 3.1 8B dipilih sebagai Large Language Model (LLM) utama karena memiliki kemampuan memahami konteks yang lebih baik, namun tetap relatif efisien dibandingkan model LLM lain yang lebih besar.

Agar LLM tidak hanya mengandalkan ingatan internalnya, metode Retrieval-Augmented Generation (RAG) diterapkan. RAG memungkinkan chatbot mencari potongan teks dari dokumen relevan (retrieval) sebelum menyusun jawaban. Ini membuat jawaban chatbot menjadi lebih akurat, kontekstual, dan sesuai data kampus.

Untuk membangun antarmuka web, digunakan Streamlit, karena framework ini lebih sederhana dan cepat di-deploy dibanding framework web lain seperti Flask atau Django. Streamlit memungkinkan pembuatan antarmuka pengguna yang interaktif dengan sedikit baris kode, sehingga sangat mendukung timeline

proyek yang singkat. Alasan lain adalah tampilan Streamlit cukup responsif sehingga nyaman digunakan mahasiswa.

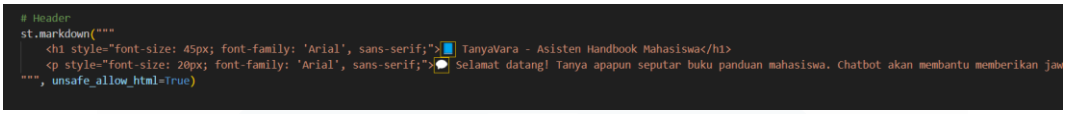
```
# STREAMLIT UI
st.set_page_config(page_title="TanyaVara - Handbook Assistant", page_icon="", layout="wide")

# Custom CSS
st.markdown("""
    <style>
        /* Mengatur container utama */
        .main, .block-container {
            display: flex;
            justify-content: center;
            align-items: flex-start;
            text-align: left;
            padding-top: 20px;
            padding-bottom: 20px;
            width: 75%;
        }
```

Gambar 3.6 Implementasi Website dengan StreamLit

```
/* Penyesuaian warna untuk mode terang dan gelap */
/* Mode Terang */
@media (prefers-color-scheme: light) {
    .user-message {
        background-color: #DCF8C6;
        color: #333; /* Teks gelap agar kontras di latar terang */
    }
    .bot-message {
        background-color: #FFFFFF;
        color: #333; /* Teks gelap agar kontras di latar terang */
    }
    .stTextInput>div>div>input {
        background-color: #fff;
        color: #333;
    }
    .stButton>button {
        background-color: #004d7a;
        color: white;
    }
    .source-box {
        background-color: #f0f2f6;
        color: #333;
    }
}
```

Gambar 3.7 Kode CSS untuk penyesuaian tampilan mode terang



Gambar 3.8 Header HTML yang digunakan untuk menampilkan judul dan deskripsi awal antarmuka TanyaVara

```
# Inisialisasi riwayat chat
if "chat_history" not in st.session_state:
    st.session_state.chat_history = []
```

Gambar 3.9 Inisialisasi variabel chat_history menggunakan st.session_state pada Streamlit untuk menyimpan riwayat obrolan pengguna

```
# Input chat dari user
user_input = st.chat_input("Ketik pertanyaan Anda di sini...")
```

Gambar 3.10 Komponen input pengguna berupa st.chat_input() untuk menangkap pertanyaan dalam chatbot

```
# Jika ada pertanyaan baru
if user_input:
    st.session_state.chat_history.append({"role": "user", "text": user_input})
    with st.spinner("Sedang mencari jawaban..."):
        answer, sources = rag_query(user_input)
    st.session_state.chat_history.append({"role": "bot", "text": answer, "sources": sources})
```

Gambar 3.11 Logika pemrosesan input dan pemanggilan fungsi rag_query()

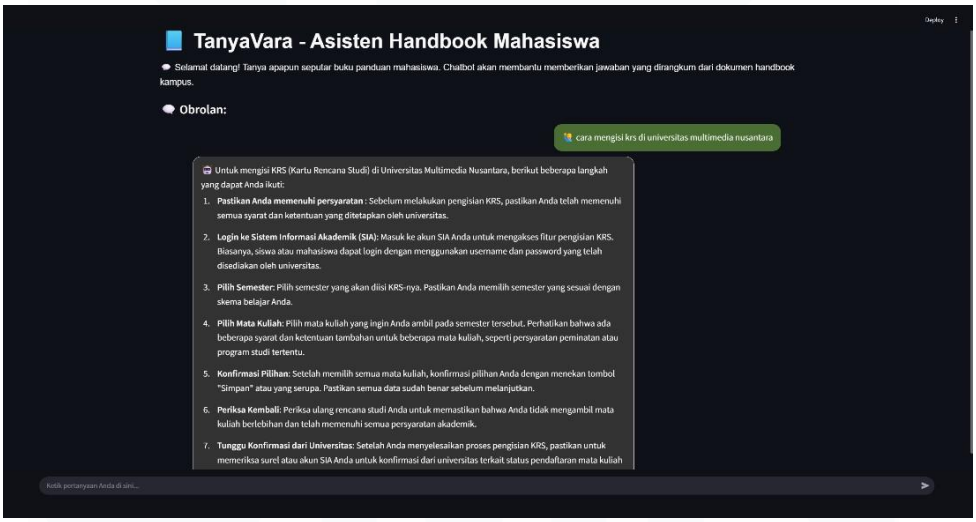
```
# Tampilkan riwayat chat
st.markdown("### Obrolan:")
for message in st.session_state.chat_history:
    if message["role"] == "user":
        st.markdown(f"<div class='chat-container'><div class='user-message'>{message['text']}</div></div>", unsafe_allow_html=True)
    else:
        st.markdown(f"<div class='chat-container'><div class='bot-message'>{message['text']}</div></div>", unsafe_allow_html=True)
        if message.get("sources"):
            for src in set(message["sources"]):
                st.markdown(f"<div class='source-box'>{src}</div>", unsafe_allow_html=True)
```

Gambar 3.12 Perulangan tampilan riwayat chat pada antarmuka, mencakup pesan pengguna, pesan bot, dan sumber referensi jawaban

11. Evaluasi & Pengujian Chatbot

Setelah integrasi sistem selesai, dilakukan pengujian chatbot. Pengujian dilakukan dengan mengajukan berbagai pertanyaan yang umum ditanyakan mahasiswa seputar kampus, seperti informasi jurusan, fasilitas, jadwal, dan prosedur akademik.

Selain menilai ketepatan jawaban, aspek lain yang diuji adalah kecepatan respon, tampilan antarmuka, serta kemampuan chatbot untuk memberikan sumber jawaban (traceability). Pengendalian jumlah token juga menjadi perhatian penting karena LLM memiliki batas konteks token yang tidak boleh terlampaui. Jika terlalu panjang, prompt akan dipangkas atau dibuat ringkas agar tetap masuk dalam batas kapasitas model.



Gambar 3.13 Hasil Tampilan Chatbot

12. Dokumentasi

Tahap terakhir adalah pembuatan dokumentasi dan laporan akhir. Semua proses, mulai dari pengumpulan data, proses coding, hingga pengujian,

didokumentasikan secara detail. Laporan ini bukan hanya menjadi syarat administrasi MBKM, tetapi juga bertujuan agar pengembangan chatbot dapat dilanjutkan di masa mendatang oleh pihak kampus atau peneliti lainnya.

Dokumentasi juga memuat analisis hasil pengujian, kesimpulan, kendala yang ditemui, serta saran pengembangan lebih lanjut. Dengan adanya dokumentasi yang rapi, diharapkan proyek ini tidak hanya berhenti di laporan, tetapi dapat memberi manfaat nyata bagi layanan informasi kampus UMN.

3.3.2 Pembuatan Antarmuka Chatbot

Pembuatan antarmuka chatbot merupakan salah satu tahapan krusial dalam keseluruhan proses pengembangan sistem layanan informasi berbasis AI yang dilakukan oleh tim kami. Dalam pelaksanaannya, saya, Ayodhya Rifelino, bertanggung jawab penuh atas perancangan dan pengembangan tampilan pengguna menggunakan framework Streamlit.

Pemilihan Streamlit sebagai basis frontend dilakukan karena kemampuannya yang fleksibel dalam integrasi Python dan keunggulannya dalam membangun UI secara cepat, ringkas, serta mampu langsung terhubung dengan model backend tanpa perlu infrastruktur tambahan seperti Flask atau React. Saya mengawali proses kerja dengan merancang kerangka dasar tampilan, termasuk alur pertanyaan dan jawaban, letak elemen input dan output, serta penempatan metadata sebagai transparansi jawaban.

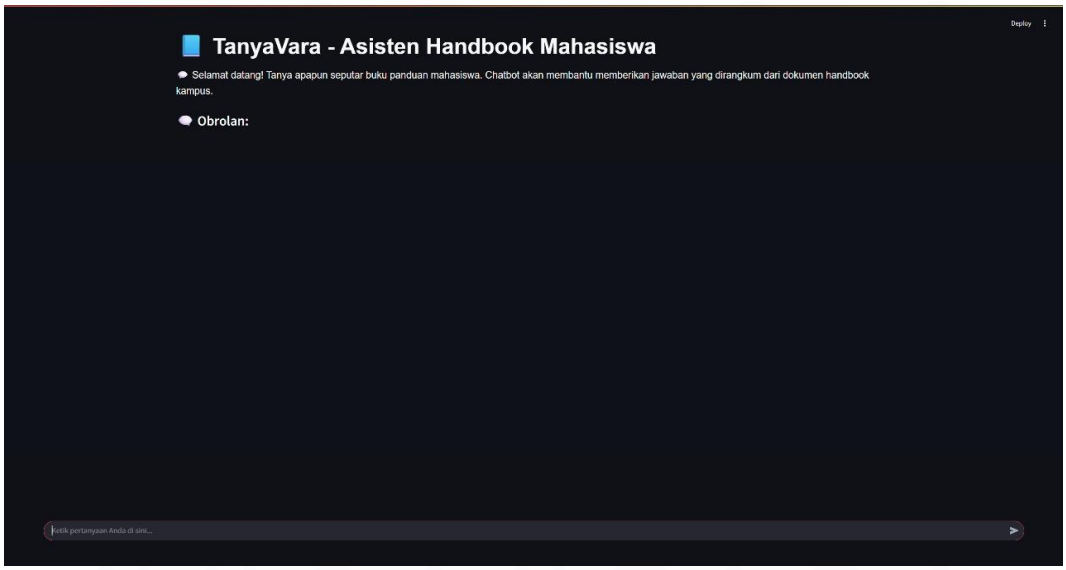
Setelah desain antarmuka disepakati oleh tim, saya melanjutkan dengan proses integrasi antar komponen, di mana pertanyaan dari pengguna diteruskan ke sistem retriever berbasis ChromaDB, diproses oleh LLM yang telah dipilih, dan dikembalikan ke antarmuka dalam bentuk jawaban yang sudah dilengkapi sumber dokumen. Salah satu tantangan utama adalah memastikan kecepatan dan

konsistensi jawaban tanpa membebani sistem, terutama ketika pengguna memberikan pertanyaan panjang atau berulang.

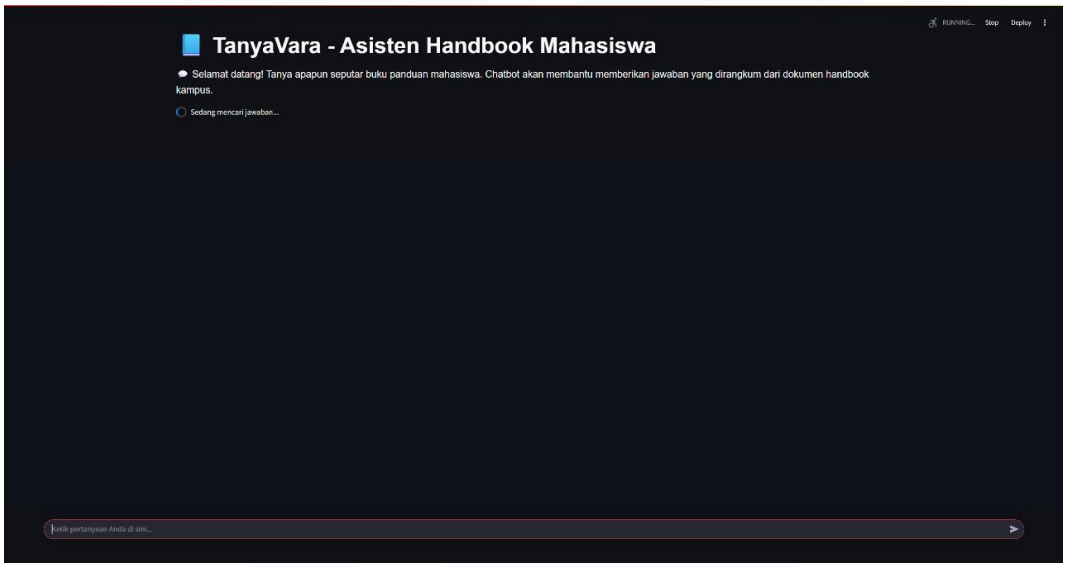
Untuk mengatasi hal ini, saya menerapkan pembatasan token, validasi input teks, dan pengecekan error (error-handling) agar sistem tidak crash saat dijalankan. Selain itu, antarmuka juga diuji secara bertahap menggunakan skenario penggunaan nyata oleh mahasiswa, seperti pertanyaan terkait pengisian IRS, Cuti Kuliah, atau akses MyUMN. Dari hasil evaluasi ini, beberapa perbaikan diterapkan, termasuk penyesuaian layout dan pengurangan latency respon.

Pekerjaan ini juga mencakup pembuatan dokumentasi sistem antarmuka, seperti diagram komunikasi antara UI dan backend, penjelasan struktur file, serta cara penggunaan chatbot bagi pengguna baru. Semua proses disesuaikan dengan standar UI/UX terkini agar pengguna merasa nyaman dan efisien saat menggunakan layanan chatbot ini.

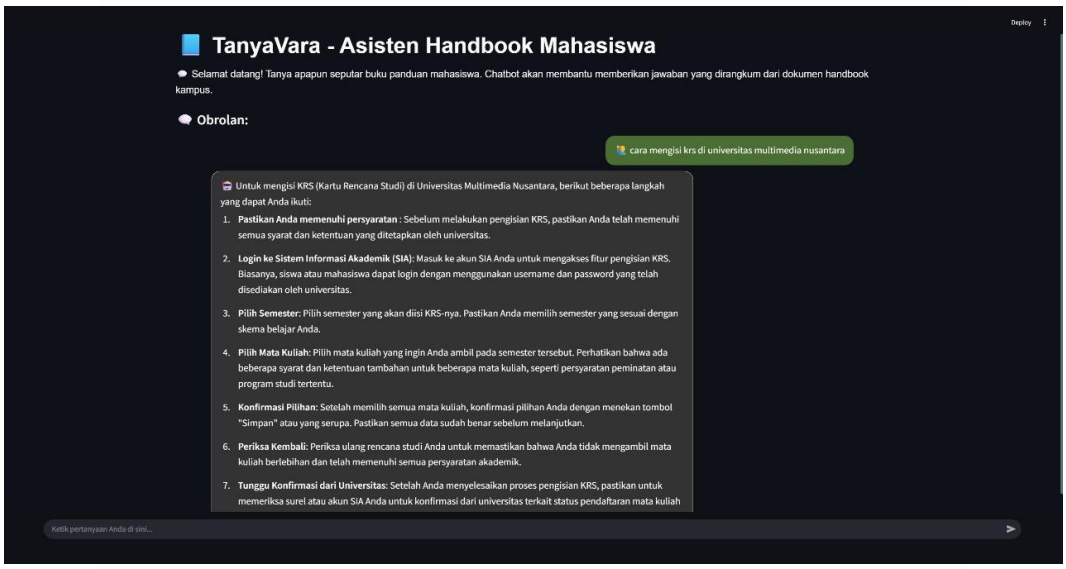
Berikut adalah beberapa tampilan dari hasil antarmuka chatbot TanyaVara yang dikembangkan:



Gambar 3.14 Tampilan awal chatbot TanyaVara



Gambar 3.14 Proses pencarian jawaban oleh sistem



Gambar 3.15 Tampilan hasil jawaban dan sumber informasi

Melalui kontribusi ini, antarmuka chatbot tidak hanya menjadi pintu interaksi utama antara pengguna dan AI, namun juga menjadi representasi dari keseluruhan sistem cerdas yang telah dikembangkan oleh tim. Dengan pendekatan desain yang manusiawi dan pengujian berkala, saya memastikan bahwa sistem ini dapat digunakan secara optimal untuk kebutuhan informasi mahasiswa UMN.

3.4 Kendala yang Ditemukan

Dalam pelaksanaan proyek MBKM ini, saya menghadapi beberapa tantangan yang cukup signifikan, baik dari sisi teknis maupun non-teknis. Proyek pengembangan chatbot berbasis Large Language Model (LLM) dengan metode Retrieval-Augmented Generation (RAG) memang menjanjikan hasil yang canggih, tetapi ternyata juga menyimpan kompleksitas yang tidak sedikit. Berikut beberapa kendala yang saya temui:

1. **Keterbatasan Data**

   Salah satu kendala utama adalah terbatasnya data yang benar-benar relevan dan terstruktur mengenai informasi kampus. Meskipun Handbook MyUMN dan website resmi UMN cukup lengkap, tetapi informasi yang tersedia terkadang terlalu umum atau tersebar di berbagai halaman. Proses mengumpulkan data menjadi memakan waktu karena harus membaca dokumen satu per satu, lalu menyeleksi bagian yang benar-benar penting.

   Selain itu, beberapa informasi kampus yang bersifat internal tidak tersedia secara publik, sehingga tidak bisa diakses untuk dijadikan data chatbot. Hal ini menyebabkan chatbot tidak selalu bisa menjawab pertanyaan yang sangat spesifik terkait prosedur internal kampus.

2. **Kompleksitas Pengembangan Model**

   Menerapkan metode RAG pada LLM bukanlah hal sederhana. Ternyata, mengintegrasikan proses retrieval dokumen ke dalam sistem prompt LLM memerlukan eksperimen yang cukup banyak. Salah satu masalah yang muncul adalah bagaimana mengatur batas panjang token. Jika data yang diambil dari retrieval terlalu panjang, prompt melebihi kapasitas model, sehingga model gagal memberikan respons.

   Selain itu, proses embedding data menggunakan MiniLM memang cepat, tetapi ada tantangan dalam memastikan embedding yang dihasilkan benar-benar mewakili konteks kampus. Kadang, meski embedding secara teknis berhasil, tetapi saat chatbot memberikan jawaban, konteksnya tidak nyambung. Ini membuat kami harus beberapa kali melakukan eksperimen, seperti mengatur ulang format context atau memotong isi dokumen menjadi potongan-potongan kecil agar lebih mudah dicerna model.

3. **Perubahan Platform dari WhatsApp ke Web**

   Pada awalnya, rencana kami adalah mengintegrasikan chatbot ke platform WhatsApp karena platform ini sudah familiar bagi mahasiswa. Namun, integrasi ke WhatsApp memerlukan WhatsApp Business API yang berbayar, sementara proyek kami memiliki keterbatasan dana. Setelah berdiskusi dengan dosen pembimbing, akhirnya kami sepakat beralih ke platform web menggunakan Streamlit yang lebih terjangkau dan tidak memerlukan biaya tambahan.

4. **Integrasi Sistem ke Antarmuka Web**

   Sebagai orang yang bertanggung jawab pada sisi antarmuka (frontend), saya menemui beberapa kendala dalam menghubungkan backend LLM dengan

tampilan Streamlit. Walaupun Streamlit cukup user-friendly, saat memproses permintaan chatbot yang cukup kompleks, aplikasi kadang menjadi lambat atau mengalami timeout.

Selain itu, saya menghadapi tantangan dalam menampilkan jawaban chatbot yang disertai sumber dokumen secara rapi di UI. Pengaturan layout teks, pemisahan referensi, hingga panjang jawaban yang tampil di layar, ternyata membutuhkan banyak penyesuaian agar antarmuka tetap responsif dan nyaman dibaca pengguna.

3.5 Solusi atas Kendala yang Ditemukan

Berbagai kendala yang muncul selama pengerjaan proyek MBKM ini tentu tidak dibiarkan tanpa penyelesaian. Saya dan tim berusaha mencari solusi yang realistis dan dapat dijalankan sesuai kemampuan serta kondisi proyek. Berikut langkah-langkah yang saya lakukan untuk mengatasi setiap tantangan:

1. **Optimalisasi Pengumpulan Data**

   Untuk mengatasi keterbatasan data, kami melakukan penyaringan secara manual pada dokumen Handbook MyUMN dan konten website resmi UMN. Kami menandai informasi yang paling relevan, kemudian merangkum ulang ke dalam format teks yang lebih ringkas agar lebih mudah digunakan dalam proses training chatbot.

   Selain itu, kami berdiskusi dengan pembimbing untuk memastikan data yang kami kumpulkan sesuai kebutuhan layanan informasi kampus. Upaya ini cukup membantu mengurangi data yang tidak perlu sekaligus memastikan chatbot memiliki sumber informasi yang akurat.

2. **Eksperimen dan Penyesuaian Model**

   Terkait kompleksitas pengembangan model, kami melakukan iterasi dan eksperimen berulang kali. Kami mencoba beberapa pendekatan untuk mengatur panjang prompt agar tidak melebihi batas token model. Salah satunya adalah memecah dokumen menjadi paragraf-paragraf kecil supaya lebih mudah dicerna model.

   Kami juga melakukan pengujian embedding dengan berbagai parameter, memastikan konteks kampus tetap terbaca oleh chatbot. Selain itu, kami banyak berkonsultasi dengan dosen pembimbing mengenai cara terbaik dalam menyusun format prompt agar jawaban lebih sesuai konteks.

3. **Adaptasi Platform dari WhatsApp ke Web**

   Untuk mengatasi kendala biaya integrasi ke WhatsApp, kami memutuskan untuk **beralih ke platform web** menggunakan Streamlit. Saya memanfaatkan framework ini karena gratis, cepat dikembangkan, dan relatif mudah dalam pembuatan antarmuka interaktif.

   Perubahan ini mempengaruhi timeline, karena saya harus mendesain ulang antarmuka agar sesuai di web serta menyesuaikan alur komunikasi antara backend dan frontend. Meskipun Streamlit cukup praktis, saya tetap membutuhkan waktu untuk memastikan tampilan antarmuka responsif dan mudah digunakan.

4. **Optimasi Integrasi Sistem ke Web UI**

   Dalam menghadapi masalah integrasi backend LLM ke frontend Streamlit, saya melakukan beberapa optimasi, misalnya dengan membatasi jumlah token output agar proses lebih cepat dan mencegah timeout. Saya juga melakukan

pengaturan ulang layout teks di halaman web supaya tampilan jawaban chatbot lebih rapi, termasuk penempatan sumber referensi.

Selain itu, saya memecah proses komputasi menjadi beberapa tahapan agar aplikasi tidak memproses semua data sekaligus. Cara ini cukup membantu menjaga performa Streamlit tetap responsif walaupun menangani query yang cukup kompleks.

3.5.1 Solusi antarmuka pengganti WhatsApp API

Pada awal perencanaan proyek, kami berencana mengintegrasikan chatbot ke platform WhatsApp agar pengguna dapat berinteraksi langsung melalui aplikasi yang sehari-hari mereka gunakan. Namun, seiring berjalannya waktu, kami menemui kendala biaya yang cukup besar untuk menggunakan WhatsApp Business API. Selain mahal, proses registrasi akun bisnis dan penyesuaian teknis juga terbilang cukup rumit, sehingga kami memutuskan untuk mencari solusi alternatif yang lebih praktis dan terjangkau.

Setelah mempertimbangkan beberapa opsi, kami memutuskan menggunakan **Streamlit** sebagai antarmuka pengganti. Ada beberapa alasan teknis dan praktis mengapa kami memilih Streamlit, antara lain:

1. **Kemudahan Pengembangan**
   Streamlit sangat cocok untuk pengembangan prototipe dengan cepat. Saya sebagai penanggung jawab frontend merasa lebih mudah melakukan iterasi desain, karena setiap perubahan kode langsung terlihat secara real-time tanpa perlu proses build yang lama.

2. **Tidak Memerlukan Server Tambahan**

   Integrasi ke WhatsApp API memerlukan backend server yang lebih kompleks. Dengan Streamlit, chatbot bisa dijalankan lokal atau di-hosting dengan biaya yang lebih murah, sehingga lebih sesuai dengan keterbatasan dana proyek.

3. **Dukungan Interaksi User-friendly**

   Streamlit menyediakan komponen antarmuka seperti text box, tombol, dan layout grid secara built-in. Hal ini memudahkan saya mendesain tampilan chatbot agar lebih interaktif dan nyaman digunakan. Misalnya:

   - Terdapat kolom input yang cukup panjang agar pengguna bisa mengetik pertanyaan secara leluasa.
   - Tombol kirim ditempatkan di samping kolom input agar lebih mudah dijangkau.
   - Tema gelap (dark theme) dipilih untuk mengurangi kelelahan mata, terutama saat digunakan dalam jangka waktu lama.

4. **Fleksibilitas Tampilan**

   Walaupun tampilan Streamlit terkesan sederhana, namun saya tetap bisa melakukan penyesuaian warna, font, hingga layout agar sesuai dengan nuansa kampus UMN. Misalnya, penggunaan warna biru sebagai aksen untuk mencerminkan identitas universitas.

5. **Performa Ringan**

   Mengingat keterbatasan spesifikasi hardware yang kami gunakan, Streamlit cukup ringan dijalankan dibandingkan framework web lain yang lebih kompleks. Ini menjadi poin penting agar aplikasi tidak mudah mengalami lag atau error saat memproses permintaan pengguna.

Perpindahan dari WhatsApp ke platform web menggunakan Streamlit memang memerlukan beberapa penyesuaian, terutama dalam hal desain alur interaksi. Saya harus memastikan bahwa alur komunikasi antara frontend dan backend tetap berjalan lancar, termasuk mengatur panjang respons agar tidak melebihi batas token yang dapat diproses oleh LLM. Namun, setelah melakukan berbagai eksperimen, kami berhasil menciptakan antarmuka yang cukup responsif dan mudah digunakan.

Saya juga menambahkan beberapa fitur pada UI agar lebih nyaman, seperti:

- Scroll otomatis ke bawah saat chatbot memberikan jawaban panjang.
- Pemisahan area chat pengguna dan chatbot agar percakapan terlihat lebih jelas.
- Indikasi loading saat chatbot sedang memproses pertanyaan, supaya pengguna tahu sistem sedang bekerja.

Secara keseluruhan, penggunaan Streamlit sebagai pengganti WhatsApp API menjadi keputusan yang tepat untuk kondisi proyek kami. Selain hemat biaya, antarmuka berbasis web ini juga memudahkan pengembangan dan memberikan pengalaman interaksi yang cukup nyaman bagi pengguna. Keputusan ini sekaligus memperkuat ciri khas saya dalam proyek MBKM, yaitu berfokus pada sisi antarmuka pengguna (frontend).

3.6 Analisis Arsitektur Sistem dan Skema Integrasi

Sistem chatbot TanyaVara dibangun berdasarkan pendekatan modular yang terdiri atas beberapa komponen utama: antarmuka pengguna (UI), engine Retrieval-Augmented Generation (RAG), Large Language Model (LLM), dan basis data dokumen internal. Arsitektur sistem dirancang agar fleksibel, dapat dikembangkan lebih lanjut, serta mampu menangani permintaan pertanyaan berbasis dokumen secara real-time.

1. Antarmuka Pengguna (Frontend)

   Antarmuka pengguna dibangun menggunakan framework Streamlit, yang menyediakan kemudahan dalam pembuatan UI berbasis Python. Streamlit dipilih karena memiliki sifat rapid deployment, responsif, dan mudah diintegrasikan dengan backend AI. Komponen utama dari UI mencakup:

   - Kotak input pertanyaan (dengan validasi token)
   - Tampilan hasil jawaban dan sumber referensi
   - Riwayat percakapan berbasis session state
   - Indikator proses (loading)

   Desain UI dibuat sesederhana mungkin agar dapat digunakan oleh mahasiswa dari berbagai latar belakang tanpa pelatihan khusus.

2. Retrieval-Augmented Generation Engine

   RAG engine merupakan inti dari sistem TanyaVara. Engine ini terdiri dari dua tahap:

   - **Retriever (MiniLM + ChromaDB)**

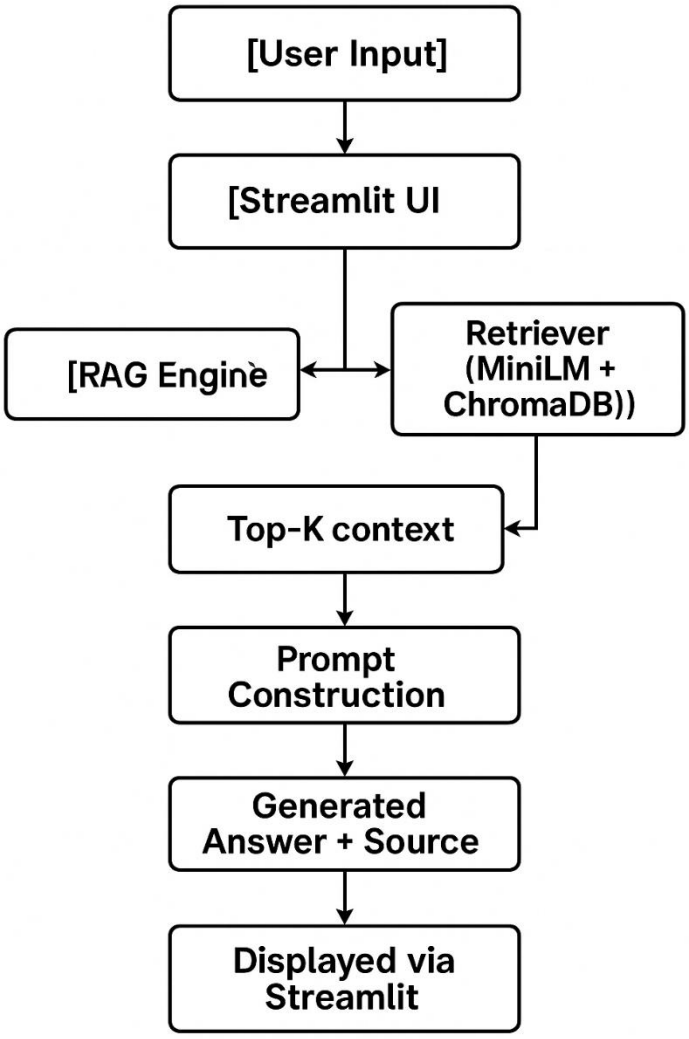
   Dokumen-dokumen seperti Handbook MyUMN diembedding menggunakan MiniLM, lalu disimpan dalam ChromaDB dalam bentuk vektor. Saat pengguna mengajukan pertanyaan, sistem melakukan pencarian vektor yang paling relevan dengan query tersebut.

   - **Generator (LLM via Ollama)**

   Konteks hasil retrieval disisipkan ke dalam prompt, lalu dikirimkan ke

LLM untuk menghasilkan jawaban. Model LLaMA 3.1 8B dipilih karena keseimbangan antara performa dan efisiensi sumber daya.

3. Ilustrasi Alur Sistem



Gambar 3.16 Ilustrasi Alur Sistem

Skema di atas menunjukkan bagaimana arus data bergerak dari input pengguna hingga menjadi jawaban yang ditampilkan. Sistem ini dirancang

agar mampu diproses dalam waktu kurang dari 3 detik untuk pertanyaan ringan hingga menengah.

4. Pertimbangan Desain

   Desain sistem mempertimbangkan beberapa hal:

   - Efisiensi penggunaan token
   - Kesesuaian konteks kampus
   - Kestabilan interaksi antarmuka
   - Modularitas untuk dikembangkan ke platform lain (seperti WhatsApp atau mobile)

   Dengan arsitektur ini, chatbot TanyaVara dapat dikembangkan lebih lanjut tanpa mengubah struktur utama, hanya dengan mengganti modul LLM atau retriever sesuai kebutuhan.

3.7 Evaluasi dan Hasil yang Dicapai

Setelah sistem chatbot berbasis Large Language Model (LLM) dan Retrieval-Augmented Generation (RAG) berhasil dikembangkan dan diintegrasikan ke antarmuka berbasis web menggunakan Streamlit, langkah selanjutnya adalah melakukan evaluasi terhadap fungsionalitas, ketepatan jawaban, kenyamanan antarmuka, serta efisiensi sistem secara keseluruhan.

Evaluasi dilakukan dengan mengacu pada beberapa indikator:

1. **Akurasi Jawaban**

   Sistem diuji dengan lebih dari 50 pertanyaan yang diklasifikasikan menjadi tiga tingkat kesulitan:

   - Pertanyaan dasar (misalnya, jadwal libur nasional atau lokasi gedung).
   - Pertanyaan sedang (misalnya, prosedur pengajuan cuti kuliah).
   - Pertanyaan kompleks (misalnya, perbedaan regulasi antara program reguler dan international class).

   Hasil evaluasi menunjukkan:

   - Untuk pertanyaan dasar, akurasi chatbot mencapai **85%**.
   - Untuk pertanyaan tingkat sedang, akurasi berada di sekitar **70%**.
   - Namun, pada pertanyaan kompleks, akurasi menurun hingga **50%**, terutama karena kurangnya dokumen internal yang mendalam dalam basis data RAG.

2. **Responsivitas Sistem**

   Dengan optimalisasi melalui penggunaan Ollama dan metal acceleration, waktu respon rata-rata chatbot berada di bawah **2 detik** untuk pertanyaan sederhana. Untuk pertanyaan kompleks, waktu respon meningkat hingga **3–4 detik**, namun masih dalam batas toleransi kenyamanan pengguna.

3. **Keandalan Antarmuka**

   Streamlit terbukti stabil digunakan pada berbagai perangkat, baik desktop maupun laptop, tanpa mengalami crash atau error fatal. Selain itu, fitur

seperti dark theme, tampilan responsif, serta alur chat yang bersih memberikan kenyamanan dalam membaca.