BAB III

PELAKSANAAN PROYEK

3.1 Kedudukan dan Koordinasi

MBKM Proyek Independen dengan judul "Pengembangan Chatbot Berbasis Large Language Model Dengan Metode Retrieval-Augmented Generation Untuk Layanan Informasi Di Universitas Multimedia Nusantara" telah dilaksanakan di Universitas Multimedia Nusantara, tepatnya di ruang Lab Big Data pada Program Studi Sistem Informasi. Dalam pelaksanaan proyek ini, penulis dibimbing oleh Dr. Irmawati, S.Kom., M.M.S.I. sebagai supervisor dan Dr. Erick Fernando, S.Kom.,M.S.I. sebagai advisor.

Dalam pelaksanaan proyek MBKM Proyek Independen ini, saya, James Andersen, berperan sebagai pengembang utama dalam implementasi teknologi Retrieval-Augmented Generation (RAG) yang menjadi inti dari sistem chatbot berbasis Large Language Model (LLM). Tanggung jawab saya mencakup berbagai tahap teknis mulai dari pengumpulan dataset internal UMN, preprocessing data, pembuatan pipeline RAG, integrasi dengan model LLM (Llama), dan pengujian terhadap respons chatbot dalam berbagai kategori pertanyaan akademik dan administratif.

Proyek ini dikerjakan secara kolaboratif oleh tiga anggota tim, yaitu James Andersen, Ayodhya Rifaliono Rifan Rangkti, dan Samuel Chandra Theodore. Masing-masing anggota memiliki tanggung jawab dan peran sebagai berikut:

1. **James Andersen**

a. Bertanggung jawab dalam aspek teknis pengembangan model **Retrieval-Augmented Generation (RAG)** yang digunakan untuk chatbot.

b. Melakukan perancangan pipeline RAG, termasuk proses embedding dokumen, pengolahan query, serta pencarian dokumen relevan melalui retriever.

c. Mengintegrasikan pipeline RAG dengan model Large Language Model (LLM) untuk menghasilkan jawaban berbasis konteks dokumen internal kampus.

d. Aktif melaporkan perkembangan teknis kepada koordinator dan pembimbing serta menindaklanjuti *feedback* dalam bentuk revisi teknis.

2. **Ayodhya Rifaliono Rifan Rangkti**

a. Bertanggung jawab atas pengembangan antarmuka pengguna (UI) chatbot berbasis **Streamlit**.

b. Mendesain tampilan UI dengan mempertimbangkan prinsip kemudahan penggunaan, estetika visual, dan UI/UX yang baik.

c. Melakukan integrasi antarmuka dengan sistem chatbot, serta mengatur tampilan input/output pengguna.

d. Melaksanakan pengujian usability dan perbaikan antarmuka berdasarkan masukan dari tim dan pembimbing.

3. **Samuel Chandra Theodore**

a. Berperan sebagai **koordinator tim** yang menjembatani komunikasi antara pembimbing lapangan dan anggota tim.

b. Menyusun perencanaan tahapan pengerjaan proyek.

c. Mengatur distribusi tugas sesuai dengan keahlian dan fokus masing-masing anggota.

d. Memeriksa hasil kerja tim secara menyeluruh sebelum diserahkan ke pembimbing untuk validasi akhir.

e. Menyampaikan arahan, revisi, dan evaluasi yang diberikan pembimbing kepada anggota tim terkait.

Dalam alur kerja proyek, seluruh anggota tim berkoordinasi secara terstruktur berdasarkan tanggung jawab masing-masing. Progres pengerjaan dan kendala teknis selalu dikomunikasikan secara terbuka, sehingga proses revisi dan

pengambilan keputusan dapat dilakukan dengan cepat dan efektif untuk menjaga kualitas hasil proyek. Penerapan alur kerja yang sistematis ini membantu menjaga kualitas proyek dan memastikan bahwa setiap anggota tim dapat menjalankan perannya secara optimal. Proses pengerjaan dilakukan secara bertahap dan saling berkesinambungan, sehingga keberhasilan setiap tahapan sangat menentukan kelancaran tahapan berikutnya. Melalui kolaborasi ini, proyek chatbot berbasis LLM-RAG dapat dikembangkan dan didokumentasikan secara utuh sesuai dengan tujuan yang dicapai.

3.2 Tugas dan Uraian Kerja

Tabel 3.1 *Timeline* Pelaksanaan Proyek

No	Minggu	Proyek	Keterangan
1	1	Pengajuan proposal dan penetapan proyek	Diskusi awal dengan supervisor lapangan. Menentukan topik terkait chatbot berbasis LLM-RAG. Penjadwalan bimbingan supervisor (Rabu & Jumat, 14:00–17:00 WIB).
2	2	Studi literatur dan inisiasi penulisan laporan	Kajian pustaka mengenai LLM dan RAG. Penyusunan awal struktur laporan MBKM serta penulisan BAB I.
3	3	Pengumpulan dataset internal kampus	Pengambilan data dari handbook MyUMN dan website UMN. Persiapan untuk fine-tuning dan proses indexing dokumen.
4	4	Preprocessing dataset dan tokenisasi dokumen	Dilakukan pembersihan teks, tokenisasi, dan pemecahan menjadi chunk teks untuk kebutuhan retriever.
5	5	Implementasi model embedding dan ChromaDB	Dilakukan Sentence Transformers untuk embedding dan menyimpan hasil ke dalam Vector Store menggunakan ChromaDB.
6	6	Penerapan pipeline Retrieval-Augmented Generation (RAG)	Penggabungan proses retrieval dan generation

			dalam satu pipeline berbasis LLM.
7	7	Pengujian fungsi RAG dan validasi jawaban	Pengujian awal fungsi “rag_query()” dengan input dummy untuk mengevaluasi konteks jawaban.
8	8	Desain UI chatbot berbasis Streamlit	Pengembangan tampilan awal antarmuka chatbot (form input, output, gaya visual).
9	9	Pembuatan visual, CSS styling, dan dokumentasi kode	Penyesuaian tampilan Streamlit agar lebih user-friendly. Penambahan dokumentasi di tiap fungsi.
10	10	Evaluasi jawaban berdasarkan kategori pertanyaan UMN	Pengujian berdasarkan kategori seperti akademik, layanan kampus, beasiswa, registrasi, dll.
11	11	Penyusunan BAB II dan BAB III laporan MBKM	Dokumentasi proses teknis ke dalam laporan proyek MBKM. Diselaraskan dengan hasil implementasi.
12	12	Validasi respons LLM terhadap mahasiswa	Pengujian model untuk menjawab input dengan gaya bahasa umum untuk digunakan oleh pengguna.
13	13	Penyusunan BAB IV dan finalisasi struktur laporan	Penulisan hasil dan pembahasan serta menyempurnakan struktur laporan.
14	14	Review internal tim dan feedback	Melakukan pengecekan lintas anggota dan diskusi perbaikan konten atau alur teknis dalam chatbot.
15	15	Penguatan fungsi logging dan output referensi dokumen	Menambahkan sumber jawaban dan histori interaksi pada antarmuka Streamlit.
16	16	Pengujian final terhadap Proyek Chatbot	Uji ketahanan penggunaan, input masif, dan validasi akhir output chatbot berbasis dokumen UMN.
17	17	Pemeriksaan awal dari pembimbing dan revisi laporan	Pengecekan isi laporan oleh supervisor dan advisor. Diberikan masukan terkait struktur dan teknis.

18	18	Perbaikan dan kelengkapan administratif MBKM	Revisi laporan, kelengkapan dokumen MBKM (LoA, kartu, bukti pendaftaran lomba).
19	19	Revisi berdasarkan feedback dosen pembimbing dan pemeriksaan Turnitin	Finalisasi format laporan, pengecekan plagiarisme, serta penyelarasan abstrak dan tugas kerja sesuai catatan reviewer.

Pada tabel 3.1 diatas, tabel tersebut menggambarkan rangkaian pelaksanaan proyek dalam pengembangan chatbot berbasis Large Language Model (LLM) dengan metode Retrieval Augmented Generation (RAG) selama 19 minggu. Selama pelaksanaan program MBKM Proyek Independen, penulis bersama dua rekan tim melaksanakan serangkaian tahapan pengembangan chatbot berbasis Large Language Model (LLM) dengan metode Retrieval-Augmented Generation (RAG). Proyek ini dirancang untuk membantu optimalisasi layanan informasi akademik di Universitas Multimedia Nusantara (UMN). Kegiatan berlangsung selama 19 minggu terbagi dalam fase perencanaan, pengembangan sistem, pengujian, evaluasi, hingga penyusunan laporan akhir.

Pada minggu pertama, dilakukan penyusunan proposal proyek serta penetapan topik dan ruang lingkup proyek. Penulis dan tim menentukan fokus pengembangan chatbot berbasis LLM dan RAG. Selain itu, dilakukan pengaturan jadwal bimbingan rutin bersama supervisor lapangan. Memasuki minggu kedua, dimulai kajian literatur yang mencakup pemahaman teoritis mengenai LLM, metode RAG, serta teknologi pendukung lainnya. Bersamaan dengan itu, penulis mulai menyusun kerangka awal laporan MBKM, khususnya bagian Bab I.

Pada minggu ketiga, penulis bertugas mengumpulkan data dan dokumen internal dari lingkungan UMN, termasuk handbook MyUMN dan informasi dari website resmi. Dokumen ini dijadikan dasar dalam penyusunan basis data chatbot. Minggu keempat digunakan untuk melakukan pembersihan dan pemrosesan data. Kegiatan ini mencakup normalisasi teks, tokenisasi, serta pemecahan teks menjadi potongan kecil untuk keperluan indexing oleh sistem retriever.

Selanjutnya, pada minggu kelima, dilakukan penerapan model embedding menggunakan Sentence Transformers yang hasilnya disimpan dalam ChromaDB sebagai vector store yang mendukung fungsi retrieval. Minggu keenam merupakan tahap penerapan pipeline RAG yang menggabungkan kemampuan retrieval dan generation berbasis dokumen. Pipeline ini menjadi inti dari chatbot yang dikembangkan.

Minggu ketujuh fokus pada uji coba fungsi rag_query() dengan masukan sederhana guna mengevaluasi keakuratan dan konteks jawaban yang dihasilkan chatbot. Minggu kedelapan difokuskan pada pengembangan antarmuka pengguna berbasis Streamlit. Penulis dan tim merancang tampilan chatbot agar ramah pengguna, mulai dari textbox input hingga area output jawaban.

Pada minggu kesembilan, dilakukan penyempurnaan visual antarmuka, penambahan elemen desain, serta dokumentasi fungsi program agar memudahkan dalam proses debugging dan pengembangan lanjutan. Minggu kesepuluh digunakan untuk melakukan pengujian jawaban chatbot berdasarkan kategori informasi umum di lingkungan UMN seperti informasi akademik, registrasi, layanan beasiswa, dan fasilitas kampus.

Pada minggu kesebelas, penulis mulai menyusun Bab II dan Bab III laporan proyek. Seluruh dokumentasi teknis dituangkan secara sistematis untuk mencerminkan proses implementasi yang telah dilakukan. Minggu kedua belas diisi dengan pengujian terhadap input dari pengguna yang menggunakan gaya bahasa yang umum untuk mahasiswa, termasuk pertanyaan yang bersifat kasual maupun tidak baku.

Selanjutnya, pada minggu ketiga belas, disusun Bab IV dan dilakukan finalisasi struktur keseluruhan laporan agar sesuai dengan pedoman penyusunan laporan MBKM. Minggu keempat belas digunakan untuk melakukan peer-review internal dalam tim dengan berdiskusi antar anggota tim terhadap konten teknis chatbot dan laporan yang telah disusun sebelumnya.

Minggu kelima belas, penulis melakukan penguatan sistem dengan menambahkan fitur logging untuk merekam riwayat interaksi pengguna serta menyertakan referensi sumber jawaban dari dokumen. Pada minggu keenam belas, dilakukan pengujian akhir chatbot untuk mengukur stabilitas sistem dalam menghadapi input dengan frekuensi tinggi serta memastikan keakuratan respons.

Minggu ketujuh belas digunakan untuk pemeriksaan laporan oleh pembimbing berupa masukan yang diberikan berkaitan dengan struktur dan substansi teknis laporan. Minggu kedelapan belas difokuskan pada perbaikan dokumen laporan, serta pelengkapan berbagai dokumen administratif MBKM seperti LoA, bukti pendaftaran lomba, dan dokumen verifikasi.

Terakhir, pada minggu kesembilan belas, penulis menyelesaikan revisi laporan berdasarkan catatan dari dosen pembimbing dan hasil pemeriksaan turnitin.

3.3 Uraian Pelaksanaan Kerja

Secara keseluruhan pelaksanaan kerja, tanggung jawab saya dalam proyek ini mencakup pengembangan chatbot melalui serangkaian langkah-langkah pengembangan teknis. Proyek ini mencakup perencanaan, riset, pengembangan teknis, dan kolaborasi. Fokus utamanya adalah menciptakan solusi chatbot berbasis LLM yang inovatif melalui pendekatan metodologis seperti RAG yang didukung oleh analisis data yang komprehensif dan studi literatur. Proses pengerjaan dilakukan secara terstruktur dengan memadukan kreativitas, keahlian teknis, dan koordinasi tim untuk mencapai target proyek.

3.3.1 Proses Pelaksanaan

Pelaksanaan proyek MBKM Proyek Independen ini dilakukan secara bertahap dan sistematis yang berfokus pada pengembangan solusi teknologi berbasis kecerdasan buatan. Proyek ini terbagi ke dalam beberapa tahapan dengan penjelasan rinci sebagai berikut:

3.3.1.1 Identifikasi Permasalahan

Universitas Multimedia Nusantara (UMN) telah menyediakan layanan asisten virtual bernama Vara yang terintegrasi ke dalam aplikasi Union. Vara merupakan

chatbot otomatis yang dirancang untuk membantu mahasiswa dalam mengakses berbagai informasi akademik dan administratif. Meskipun demikian, fitur yang dimiliki oleh Vara masih terbatas pada daftar topik yang telah ditentukan sebelumnya. Hal ini menyebabkan keterbatasan dalam menjawab pertanyaan pengguna yang bersifat lebih bebas dan kontekstual.

Keterbatasan tersebut mencakup kemampuan dalam memahami bahasa alami dari mahasiswa secara fleksibel serta dalam melakukan pencarian informasi berbasis dokumen internal kampus secara komprehensif. Penggunaan menu yang terbatas dan kurangnya kemampuan adaptif terhadap variasi bahasa pengguna mengurangi efektivitas chatbot dalam memberikan pelayanan informasi yang optimal.

Berdasarkan kondisi tersebut, proyek MBKM Proyek Independen ini bertujuan untuk mengembangkan chatbot berbasis Large Language Model (LLM) dengan pendekatan Retrieval-Augmented Generation (RAG). Tujuannya adalah untuk menciptakan sistem yang mampu menjawab pertanyaan secara langsung menggunakan bahasa alami dan mengakses informasi dari berbagai sumber dokumen internal UMN secara efisien. Solusi ini diharapkan dapat meningkatkan kualitas layanan informasi kampus dengan memberikan jawaban yang lebih luas, relevan, dan kontekstual kepada pengguna.

3.3.1.2 Studi Literatur

Dalam pelaksanaan proyek pengembangan chatbot berbasis Large Language Model (LLM) dengan metode Retrieval-Augmented Generation (RAG), dilakukan studi literatur untuk memahami perkembangan, tantangan, aplikasi chatbot dalam pendidikan serta bidang lainnya, dan pendekatan teknis yang relevan.

1. Teknologi dan Konsep Smart Campus

Konsep smart campus menjadi latar belakang penting dalam proyek ini karena menggabungkan pemanfaatan teknologi informasi untuk meningkatkan efisiensi dan layanan kampus. Zhang et al. menyatakan bahwa pengembangan sistem berbasis kebutuhan pengguna dalam smart campus sangat bergantung pada layanan digital yang cepat dan responsif [1]. Hal ini didukung pula oleh gagasan arsitektur

empat pilar smart campus menurut Baba et al., yang menempatkan sistem informasi berbasis AI sebagai komponen penting dalam transformasi digital institusi pendidikan [2].

2. Large Language Model (LLM) dan Potensi dalam Sistem Chatbot

LLM adalah model berbasis *deep learning* dengan kapasitas besar untuk memahami dan menghasilkan bahasa alami. Beberapa studi menunjukkan bahwa LLM dapat digunakan sebagai mesin pemrosesan teks untuk berbagai keperluan, termasuk layanan pendidikan dan kesehatan [3][21][22]. Model seperti GPT, BERT, hingga Gemini terbukti mampu merespons pertanyaan pengguna dengan konteks yang kompleks [9][20][31]. Namun, pelatihan ulang (*fine-tuning*) model berukuran besar masih menjadi tantangan, baik dari sisi komputasi maupun efisiensi [8][11][27].

Riset oleh Tinn et al. dan Xia et al. menunjukkan berbagai pendekatan dalam melakukan fine-tuning yang efisien untuk meningkatkan akurasi respons model di domain tertentu [11][12]. LLM juga telah dimanfaatkan untuk memberikan *feedback* edukatif kepada mahasiswa [15][16][30].

3. Konsep Retrieval-Augmented Generation (RAG)

Metode Retrieval-Augmented Generation (RAG) merupakan pendekatan baru yang menggabungkan proses *retrieval* informasi dari basis data eksternal dengan kemampuan *generation* dari LLM. Miao et al. menjelaskan bahwa RAG dapat meningkatkan keakuratan jawaban karena berfokus pada sumber dokumen aktual [3]. Siriwardhana et al. mengembangkan model RAG untuk open-domain QA dan menemukan bahwa domain adaptation adalah komponen penting agar output tetap relevan [18].

Beberapa teknik lanjutan pada RAG juga telah dieksplorasi, seperti self-knowledge guided retrieval, in-context retrieval, hingga RAG berbasis arsitektur enterprise [19][23][24]. Studi oleh Triwicaksana dan Oktavia membuktikan bahwa RAG dapat diterapkan secara efektif dalam konteks kampus, sebagai media pembelajaran dan pusat informasi digital [25].

4. Tantangan dan Etika dalam Penggunaan LLM

Studi oleh Zamfirescu-Pereira et al. dan Goyal et al. mengungkapkan bahwa pengguna non-teknis sering kali kesulitan dalam membuat prompt yang efektif, sehingga desain sistem perlu mempertimbangkan usability [4][5]. Dalam aspek etika, penting untuk mempertimbangkan prinsip keadilan, transparansi, dan bias model, sebagaimana dibahas oleh Bang et al. [14]. Hal ini juga mencakup pertanyaan seputar akurasi jawaban LLM di bidang sensitif seperti kedokteran dan hukum [26][31].

5. Aplikasi LLM di Berbagai Bidang

LLM telah berhasil diterapkan dalam berbagai domain mulai dari kesehatan, manufaktur, hingga industri konstruksi [21][26][12][10]. Chatbot berbasis LLM juga telah digunakan sebagai sistem respons akademik institusional, serta sebagai alat bantu dalam pembelajaran [13][15][16].

RAG dan LLM juga digunakan dalam pendeteksian hoaks, sistem rekomendasi, hingga generasi teks ilmiah [28][14][17]. Jeong bahkan membangun arsitektur layanan generatif yang dapat diimplementasikan di tingkat organisasi untuk optimalisasi data internal [24].

Berdasarkan literatur yang telah dianalisis, LLM dan RAG terbukti menjadi pendekatan yang kuat untuk mengembangkan sistem chatbot kampus yang kontekstual, responsif, dan terintegrasi dengan basis pengetahuan institusi. Pendekatan ini sejalan dengan arah pengembangan smart campus yang menekankan pada otomasi layanan dan pengalaman pengguna. Kombinasi fine-tuning efisien, infrastruktur RAG, dan desain antarmuka yang ramah pengguna menjadi fondasi teknis utama dalam proyek MBKM ini.

3.3.1.3 Penyusunan Desain Teknis

Desain teknis melibatkan implementasi model LLaMA 8B yang diintegrasikan dengan database vektor menggunakan ChromaDB. Data yang telah dikumpulkan diproses melalui metode embedding dengan SentenceTransformers, lalu dibangun

pipeline RAG untuk menjawab pertanyaan pengguna secara otomatis. Antarmuka pengguna dirancang menggunakan Streamlit dengan struktur chat interaktif.

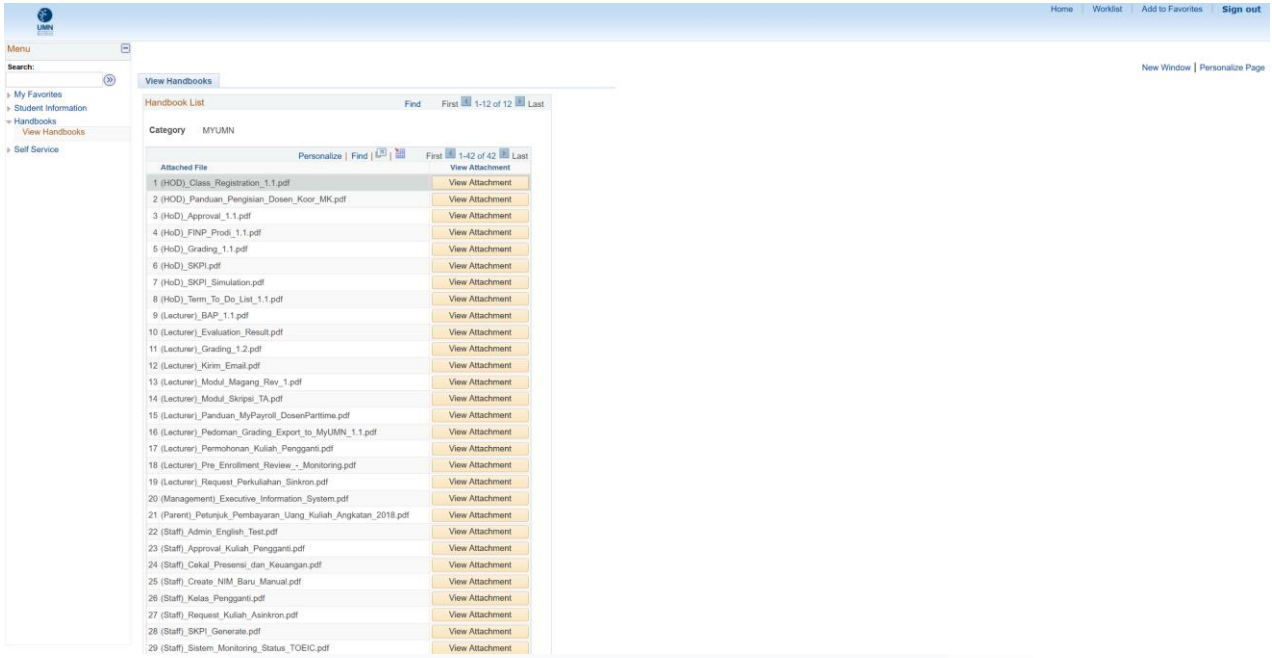
3.3.1.4 Pengembangan Produk/Jasa Layanan

Produk yang dikembangkan adalah chatbot TanyaVara, sistem layanan informasi berbasis LLM-RAG yang dirancang untuk menjawab pertanyaan pengguna secara kontekstual dengan mengacu pada dokumen internal UMN. Sistem ini tidak hanya mengandalkan pengetahuan dari model LLM, tetapi juga memperkuat akurasinya melalui proses retrieval dokumen relevan.

3.3.1.5 Pengumpulan Dataset Internal Kampus

Data dikumpulkan dari berbagai sumber internal Universitas Multimedia Nusantara, seperti handbook akademik MyUMN, halaman situs resmi, dan dokumen terkait prosedur akademik. Data dikumpulkan pada minggu ke-3 hingga ke-4. Seluruh dokumen berbasis PDF dilakukan koversi ke dalam bentuk teks menggunakan *library* PyMuPDF. Teks tersebut kemudian dibersihkan dari karakter non-alfabet dan dibagi menjadi beberapa bagian (chunk) sepanjang 500 token untuk persiapan indexing dalam sistem retriever. Dataset ini digunakan untuk training dan sebagai sumber jawaban chatbot.

Gambar 3.1 Website Universitas Multimedia Nusantara



Gambar 3.2 Handbook MyUMN

3.3.1.6 Implementasi dan Pengembangan Sistem

Setelah dataset berhasil dikumpulkan dan dibersihkan, langkah berikutnya adalah mengembangkan model dengan memanfaatkan pendekatan Retrieval- Augmented Generation (RAG) yang digabungkan dengan Large Language Model (LLM). Dalam proyek ini, model LLM yang dipilih adalah LLaMA 8B yang kemudian disesuaikan menggunakan teknik RAG. Pendekatan RAG memungkinkan model untuk mengambil informasi relevan dari kumpulan data eksternal, sehingga model tidak hanya bergantung pada data pelatihan awalnya saja, melainkan juga dapat mengakses konteks spesifik yang ada dalam dataset yang telah disiapkan. Dengan demikian, model dilatih agar mampu memahami dan memberikan respons yang lebih akurat dan kontekstual terkait informasi seputar UMN secara khusus.

```
NAME            ID              SIZE      MODIFIED
llama3.1:8b     46e0c10c039e    4.9 GB    2 months ago
```

Gambar 3.3 Model LLM

```python
import streamlit as st
import os
import re
import fitz  # PyMuPDF
from langchain_core.documents import Document
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_chroma import Chroma
from langchain_ollama import OllamaLLM
from transformers import AutoTokenizer
import shutil
```

Gambar 3.4 Import Library

Proyek ini dimulai dengan mengimpor berbagai library penting yang mendukung seluruh proses seperti yang ditampilkan pada gambar 3.4, mulai dari pembuatan antarmuka pengguna berbasis web menggunakan Streamlit, pengelolaan file dan folder dengan modul os serta shutil, hingga pemrosesan dokumen PDF dengan PyMuPDF (fitz). Selain itu, proyek ini memanfaatkan library LangChain untuk mengelola dokumen dan database vektor, HuggingFace

```python
# --- Konfigurasi ---
PDF_FOLDER = "pdf_files"
DB_FOLDER = "./rag_db"
COLLECTION_NAME = "student_handbooks"
EMBED_MODEL = "sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2"
MODEL_CONTEXT_LIMIT = 4096
RESPONSE_TOKENS = 128
BUFFER_TOKENS = MODEL_CONTEXT_LIMIT - RESPONSE_TOKENS
```

untuk model embedding, serta Transformers untuk tokenisasi teks. Model LLM diakses melalui Ollama yang memungkinkan dalam melakukan interaksi dengan model.

Gambar 3.5 Konfigurasi variabel

Pada bagian konfigurasi yang ditampilkan gambar 3.5, beberapa parameter didefinisikan seperti lokasi folder untuk file PDF, direktori database vektor, nama koleksi dokumen, dan model embedding yang digunakan (model multilingual dari Sentence Transformers). Batas jumlah token untuk konteks dan respons juga diatur agar sesuai dengan kapasitas model LLM yang digunakan.

```
# Tokenizer
tokenizer = AutoTokenizer.from_pretrained("bert-base-multilingual-cased")
def count_tokens(text):
    return len(tokenizer.encode(text, add_special_tokens=False))
```

Gambar 3.6 Tokenizer

Selanjutnya, gambar 3.6 menampilkan kode melakukan inisialisasi tokenizer dari HuggingFace (bert-base-multilingual-cased) yang berfungsi untuk menghitung jumlah token dalam setiap teks. Fungsi ini sangat penting untuk memastikan bahwa input ke model LLM tidak melebihi batas token yang diizinkan, sehingga proses inferensi dapat berjalan lancar tanpa error.

```
# Embeddings
embeddings = HuggingFaceEmbeddings(model_name=EMBED_MODEL)
```

Gambar 3.7 Embeddings

Bagian berikutnya adalah proses embedding seperti yang ditampilkan pada gambar 3.7, di mana model embedding dari HuggingFace digunakan untuk mengubah potongan teks menjadi vektor numerik. Vektor-vektor ini kemudian

disimpan dalam database vektor (ChromaDB), sehingga memungkinkan pencarian dokumen relevan secara efisien berdasarkan kemiripan semantik.

```python
# Fungsi PDF
def test_pdf_open(file_path):
    try:
        doc = fitz.open(file_path)
        print(f"✅ Berhasil membuka PDF: {file_path}")
        return True
    except Exception as e:
        print(f"⚠ Gagal membuka PDF: {e}")
        return False

def extract_text_with_fitz(file_path):
    text = ""
    if test_pdf_open(file_path):
        doc = fitz.open(file_path)
        for page in doc:
            text += page.get_text("text") + "\n"
    return text.strip()

def clean_text(text):
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\n+', '\n', text)
    return text.strip()

def split_text_into_chunks(text, max_tokens=512):
    words = text.split()
    chunks = []
    while words:
        chunk = " ".join(words[:max_tokens])
        chunks.append(chunk)
        words = words[max_tokens:]
    return chunks
```

Gambar 3.8 Pengolahan dokumen PDF

Dalam melakukan pengolahan dokumen PDF yang ditampilkan pada gambar 3.8 diatas, terdapat beberapa fungsi kunci yang dimana fungsi “test_pdf_open” digunakan untuk memastikan file PDF dapat dibuka tanpa error. Fungsi “extract_text_with_fitz” bertugas mengekstrak seluruh teks dari setiap halaman PDF. Setelah teks diekstrak, fungsi “clean_text” akan membersihkan teks dari spasi berlebih dan baris kosong, sehingga menghasilkan data yang lebih rapi dan siap diolah. Selanjutnya, fungsi “split_text_into_chunks” akan memecah teks

panjang menjadi potongan-potongan kecil dengan batas token tertentu, agar sesuai dengan kapasitas model embedding dan LLM.

```
def load_documents_from_pdfs():
    documents = []
    ids = []
    doc_id = 0
    for filename in os.listdir(PDF_FOLDER):
        if filename.endswith(".pdf"):
            file_path = os.path.join(PDF_FOLDER, filename)
            if test_pdf_open(file_path):
                text = extract_text_with_fitz(file_path)
                cleaned_text = clean_text(text)
                if cleaned_text and len(cleaned_text.split()) > 5:
                    for chunk in split_text_into_chunks(cleaned_text):
                        documents.append(Document(
                            page_content=chunk,
                            metadata={"filename": filename},
                            id=str(doc_id)
                        ))
                        ids.append(str(doc_id))
                        doc_id += 1
    return documents, ids
```

Gambar 3.9 Pemrosesan dokumen PDF

Fungsi “load_documents_from_pdfs” yang ditampilkan pada gambar 3.9 diatas, kemudian memproses seluruh file PDF yang ada di folder, mengekstrak, membersihkan, dan memecah teksnya, lalu menyimpannya sebagai objek dokumen

lengkap dengan metadata seperti nama file dan ID dokumen. Proses ini memastikan setiap bagian dokumen dapat dicari dan diakses melalui database vektor.

```
# Inisialisasi ChromaDB
if os.path.exists(DB_FOLDER):
    try:
        shutil.rmtree(DB_FOLDER)
        print("🔄 Database dihapus untuk mencegah lock error.")
    except Exception as e:
        print(f"⚠ Gagal menghapus database: {e}")

add_documents = not os.path.exists(DB_FOLDER)
vector_store = Chroma(
    collection_name=COLLECTION_NAME,
    persist_directory=DB_FOLDER,
    embedding_function=embeddings
)
if add_documents:
    docs, ids = load_documents_from_pdfs()
    print(f"📁 Total dokumen yang dimasukkan ke ChromaDB: {len(docs)}")

retriever = vector_store.as_retriever(search_kwargs={"k": 5})
```

Gambar 3.10 Inisialisasi ChromaDB

Pada gambar 3.10 diatas, sebelum dokumen baru dimasukkan ke ChromaDB, database lama dihapus terlebih dahulu untuk mencegah error akibat file yang terkunci. Setelah itu, dokumen hasil ekstraksi dimasukkan ke dalam koleksi ChromaDB. Database vektor ini kemudian diinisialisasi sebagai retriever yang dapat mencari hingga lima dokumen paling relevan untuk setiap pertanyaan yang diajukan pengguna.

```
# LLM
llm = OllamaLLM(model="llama3.1:8b", base_url="http://localhost:11434")
```

Gambar 3.11 Inisialisasi Model LLM

Model LLM pada gambar 3.11, berjalan secara lokal melalui Ollama yang diinisialisasi untuk menghasilkan jawaban berdasarkan konteks yang diberikan.

```
#  RAG Query
def rag_query(query):
    results = retriever.invoke(query)
    print(f"🔎 {len(results)} dokumen ditemukan")

    base_prompt = "Jawablah pertanyaan berikut berdasarkan konteks yang diberikan.\n\nKonteks:\n"
    query_part = f"\n\nPertanyaan: {query}\nJawaban:"
    fixed_tokens = count_tokens(base_prompt + query_part)
    max_tokens_for_context = MODEL_CONTEXT_LIMIT - RESPONSE_TOKENS - fixed_tokens

    context_chunks = []
    used_sources = []
    total_context_tokens = 0
    for doc in results:
        chunk = doc.page_content
        tokens = count_tokens(chunk)
        if total_context_tokens + tokens > max_tokens_for_context:
            continue
        context_chunks.append(f"[{doc.metadata.get('filename', 'Tidak ada metadata')}]\n{chunk}")
        used_sources.append(doc.metadata.get("filename", "Tidak ada metadata"))
        total_context_tokens += tokens

    if not context_chunks and results:
        doc = min(results, key=lambda d: count_tokens(d.page_content))
        context_chunks.append(f"[{doc.metadata.get('filename', 'Tidak ada metadata')}]\n{doc.page_content}")
        used_sources.append(doc.metadata.get("filename", "Tidak ada metadata"))

    context = "\n\n".join(context_chunks)
    full_prompt = f"{base_prompt}{context}{query_part}"
    response = llm.stream(full_prompt)
    answer = "".join(chunk for chunk in response)

    return answer.strip(), used_sources
```

Gambar 3.12 Inisialisasi RAG Query

Pada gambar 3.12 diatas, fungsi utama dalam menjalankan proses RAG adalah “rag_query”. Fungsi ini menerima pertanyaan pengguna, mencari dokumen-dokumen paling relevan dari database vektor, lalu membangun prompt yang berisi instruksi, konteks hasil pencarian, dan pertanyaan pengguna. Prompt ini kemudian dikirim ke model LLM untuk menghasilkan jawaban yang relevan dan terfokus pada konteks yang telah disediakan. Fungsi ini juga memastikan total token yang

digunakan tidak melebihi batas yang telah ditentukan sehingga respons tetap optimal.

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

Gambar 3.13 Konfigurasi StreamLit

Penggunaan Streamlit merupakan fondasi dari antarmuka chatbot TanyaVara yang dirancang untuk membantu mahasiswa dalam mencari informasi dari buku panduan kampus. Pada gambar 3.13 diatas, digunakan untuk mengatur konfigurasi dasar halaman web aplikasi. Dengan “page_title”, judul tab browser akan menampilkan "TanyaVara - Handbook Assistant", sementara “page_icon” menampilkan emoji buku yang merepresentasikan tema edukasi. Pengaturan layout="wide" memastikan bahwa seluruh elemen aplikasi dapat memanfaatkan lebar layar secara maksimal, sehingga tampilan terasa lebih lega dan nyaman digunakan di berbagai perangkat.

```
# 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%;
        }

        /* Chat container style */
        .chat-container {
            display: flex;
            flex-direction: column;
            gap: 12px;
            width: 90%;
            margin: auto;
        }

        /* Gaya untuk pesan pengguna */
        .user-message {
            padding: 14px 18px;
            border-radius: 18px;
            margin-bottom: 12px;
            max-width: 75%;
            align-self: flex-end;
            font-size: 20px;
            word-wrap: break-word;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }

        /* Gaya untuk pesan bot */
        .bot-message {
            padding: 14px 18px;
            border-radius: 18px;
            margin-bottom: 12px;
            max-width: 75%;
            align-self: flex-start;
            font-size: 20px;
            word-wrap: break-word;
            border: 1px solid #ddd;
            box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
        }
```

Gambar 3.14 Kostumisasi dan Gaya Chatbot

```css
/* Sumber dokumen */
.source-box {
    background-color: #f0f2f6;
    padding: 0.75em;
    border-left: 5px solid #004d7a;
    margin-top: 0.5em;
    font-size: 16px;
    margin-left: 25px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
}

/* Styling untuk input form chat */
.stTextInput>div>div>input {
    background-color: #333;
    color: #fff;
    font-size: 24px;
    padding: 30px 35px;
    border-radius: 15px;
    width: 500px;
    max-width: 90%;
    margin: auto;
    box-sizing: border-box;
}

.stButton>button {
    background-color: #004d7a;
    color: white;
    font-size: 16px;
    padding: 12px;
    border-radius: 12px;
    width: 100%;
}

.stButton>button:hover {
    background-color: #0066a1;
}

/* Menambahkan padding untuk responsif di perangkat mobile */
@media (max-width: 768px) {
    .chat-container {
        width: 100%;
    }

    .user-message, .bot-message {
        font-size: 18px;
    }

    .stTextInput>div>div>input {
        font-size: 20px;  /* Ukuran font lebih besar di mobile */
        width: 80%;  /* Membatasi lebar di mobile */
    }

    .stButton>button {
        font-size: 18px;
    }
}
```

Gambar 3.15 Kostumisasi Tampilan Chatbot

```
        /* 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;
            }
        }

        /* Mode Gelap */
        @media (prefers-color-scheme: dark) {
            .user-message {
                background-color: #4A6E33; /* Warna lebih gelap untuk user di mode gelap */
                color: #fff; /* Teks terang untuk kontras */
            }
            .bot-message {
                background-color: #333333;
                color: #fff; /* Teks terang agar kontras di latar gelap */
            }
            .stTextInput>div>div>input {
                background-color: #444;
                color: #fff;
            }
            .stButton>button {
                background-color: #004d7a;
                color: white;
            }
            .source-box {
                background-color: #333;
                color: #fff;
            }
        }
    </style>
""", unsafe_allow_html=True)
```

Gambar 3.16 Menyesuaikan Tampilan Chat Mode Gelap dan Terang

Pada tampilan gambar 3.14, 3.15, dan 3.16 di atas, ditambahkan CSS ke dalam aplikasi Streamlit. CSS ini mengatur hampir seluruh aspek visual aplikasi, mulai dari tata letak utama (.main, .block-container) yang diposisikan di tengah dan diberi padding agar tidak terlalu rapat dengan tepi layar, hingga pengaturan gaya untuk setiap elemen chat. Pesan pengguna (.user-message) dan pesan chatbot (.bot-message) dibuat berbeda agar mudah dibedakan secara visual, baik dari segi warna, posisi, maupun efek bayangan. Sumber jawaban yang diberikan chatbot ditampilkan dalam kotak khusus (.source-box) yang menonjol dengan garis tebal di sisi kiri, sehingga pengguna dapat langsung mengetahui referensi jawaban yang diberikan.

Selain itu, CSS ini juga memperhatikan aspek responsivitas. Melalui media query, ukuran font dan lebar elemen akan menyesuaikan secara otomatis ketika aplikasi diakses melalui perangkat mobile, sehingga tetap nyaman digunakan bagi pengguna seluler. Terakhir, CSS dibuat adaptif terhadap preferensi mode terang atau gelap untuk pengguna, guna menjaga visibilitas dan kenyamanan membaca di berbagai kondisi cahaya.

```
# Header
st.markdown("""
    <h1 style="font-size: 45px; font-family: 'Arial', sans-serif;">📘 TanyaVara - Asisten Handbook Mahasiswa</h1>
    <p style="font-size: 20px; font-family: 'Arial', sans-serif;">💬 Selamat datang! Tanya apapun seputar buku panduan mahasiswa. Chatbot akan membantu memberikan jawaban yang dirangkum dari dokumen handbook kampus.</p>
""", unsafe_allow_html=True)
```

Gambar 3.17 Tampilan Header Aplikasi

Kode pada gambar 3.17 diatas, menampilkan header dengan judul besar dan deskripsi singkat. Judul dan paragraf deskripsi dibuat menggunakan HTML agar tampilan lebih fleksibel dan menarik. Bagian ini memberikan kesan profesional dan langsung menjelaskan fungsi utama kepada pengguna baru.

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

Gambar 3.18 Inisialisasi Riwayat Chat

Pada gambar 3.18 diatas, inisialisasi riwayat chat tersebut berfungsi untuk menginisialisasi riwayat chat. Dengan memanfaatkan “st.session_state”, aplikasi

dapat menyimpan seluruh percakapan antara pengguna dan bot selama sesi berlangsung. Jika belum ada riwayat, maka akan dibuat list kosong sebagai penampung pesan.

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

Gambar 3.19 Tampilan Input Chat User

Pada gambar 3.19 input chat User diatas, kode tersebut menampilkan input box di bagian bawah aplikasi, tempat pengguna dapat mengetik pertanyaan mereka dengan placeholder "Ketik pertanyaan Anda di sini..." untuk membantu memberikan petunjuk kepada pengguna mengenai fungsi kolom tersebut.

```
# 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 3.20 Kondisi Pengiriman Pertanyaan

Pada gambar 3.20 diatas, ketika pengguna mengirimkan pertanyaan, kode ini akan menambahkannya ke dalam riwayat chat dengan peran "user". Selanjutnya, aplikasi menampilkan animasi loading dengan pesan "Sedang mencari jawaban..." selama proses pencarian jawaban berlangsung. Fungsi “rag_query” dipanggil untuk mencari jawaban yang relevan dari dokumen handbook, dan hasilnya berupa jawaban serta sumber dokumen yang disimpan kembali ke dalam riwayat chat sebagai pesan dari chatbot.

```
# 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 3.21 Tampilan Riwayat Chat

Pada bagian gambar 3.21 diatas, kode tersebut bertugas menampilkan seluruh riwayat percakapan di layar. Judul "Obrolan" memperjelas bahwa bagian ini adalah ruang diskusi antara pengguna dan bot. Setiap pesan pengguna dan chatbot ditampilkan dengan gaya yang telah diatur oleh CSS, sehingga mudah dibedakan. Jika chatbot memberikan jawaban yang disertai sumber, maka sumber tersebut akan muncul dalam kotak khusus di bawah pesan chatbot, memperkuat transparansi dan kepercayaan terhadap jawaban yang diberikan.

Secara keseluruhan, kode ini memperlihatkan perhatian terhadap pengalaman pengguna, baik dari sisi tampilan visual yang modern dan responsif, maupun dari sisi fungsionalitas yang memudahkan interaksi dan pelacakan sumber informasi. Dengan struktur ini, chatbot “TanyaVara” tidak hanya membantu mahasiswa mendapatkan jawaban dari handbook, tetapi juga memberikan pengalaman digital yang profesional dan menyenangkan, sesuai dengan penulisan yang baik dan penjelasan yang alami serta mudah dipahami.

3.3.1.7 Pengujian dan Evaluasi Chatbot

Pertanyaan Mudah	(Ada di RAG)	4.1
Pertanyaan Biasa	(Biasa ditanya mahasiswa)	2.3
Pertanyaan Sulit	(Dicampur dari berbagai index)	2
Rata-Rata		2.8

Gambar 3.22 Hasil Evaluasi

Dalam mengevaluasi ketepatan model LLM yang digunakan seperti gambar 3.22 diatas, pertanyaan-pertanyaan dibagi menjadi tiga kategori berdasarkan tingkat kesulitannya: sederhana, standar, dan tangguh. Setiap jawaban yang dihasilkan oleh LLM dinilai kebenarannya menggunakan skala dari 1 sampai 5. Secara

keseluruhan, hasil evaluasi menunjukkan bahwa rata-rata skor kebenaran model adalah 2,8, yang mengindikasikan bahwa ketepatan LLM masih tergolong moderat dan belum sepenuhnya dapat diandalkan untuk menjawab berbagai jenis pertanyaan secara konsisten.

Untuk pertanyaan yang tergolong sederhana, yang biasanya sudah tercakup dalam sistem Retrieval Augmented Generation (RAG), tingkat kebenaran meningkat signifikan hingga mencapai skor rata-rata 4,1. Hal ini menunjukkan bahwa ketika LLM memiliki akses langsung ke informasi latar belakang yang relevan dan tersimpan dalam database RAG, model dapat memberikan respons yang sangat akurat. Dengan kata lain, RAG mampu meningkatkan performa LLM dengan menyediakan informasi kontekstual dari kumpulan dokumen yang besar dan relevan.

Namun, tantangan muncul ketika LLM dihadapkan pada pertanyaan yang lebih kompleks, terutama yang berkaitan dengan aturan atau situasi administratif kampus yang memerlukan pemahaman mendalam terhadap konteks tersebut. Skor kebenaran untuk pertanyaan rumit ini turun drastis menjadi sekitar 1,5, jauh lebih rendah dibandingkan dengan skor rata-rata untuk pertanyaan sederhana. Penurunan ini mengindikasikan bahwa LLM cenderung membuat kesalahan atau memberikan informasi yang kurang akurat karena keterbatasan referensi atau data domain spesifik yang mendukung jawaban tersebut.

Salah satu penyebab utama ketepatan yang rendah pada pertanyaan rumit adalah cakupan data dalam sistem RAG yang terbatas, yang selama evaluasi hanya mengandalkan buku pegangan siswa. Meskipun buku pegangan tersebut memuat informasi dasar, dokumen ini kurang lengkap dan kurang rinci untuk menjawab pertanyaan yang membutuhkan analisis aturan kampus yang kompleks atau logika administratif yang canggih. Oleh karena itu, untuk meningkatkan kinerja LLM, sangat penting untuk memperluas cakupan data RAG dengan memasukkan dokumen pendukung lain, seperti pedoman administrasi internal, FAQ resmi kampus, serta dokumen kelembagaan lainnya yang lebih komprehensif.

Untuk mengatasi masalah presisi pada pertanyaan yang kompleks, strategi utama adalah memperluas jangkauan informasi yang tersedia dalam kerangka kerja RAG. Hal ini meliputi pengumpulan berbagai dokumen resmi, seperti aturan sekolah secara lengkap, peraturan staf, notulen rapat, dan surat-surat resmi dari unit-unit kampus. Dengan menyediakan sumber informasi yang lebih lengkap dan terperinci, sistem akan memiliki basis data yang lebih kuat untuk memberikan jawaban yang tepat dan relevan terhadap pertanyaan yang sebelumnya sulit dijawab dengan benar. Selain itu, pengembangan Metadata atau sistem klasifikasi dokumen yang efektif juga sangat membantu dalam meningkatkan kemampuan Search Engine untuk memberikan konteks yang sesuai bagi setiap pertanyaan.

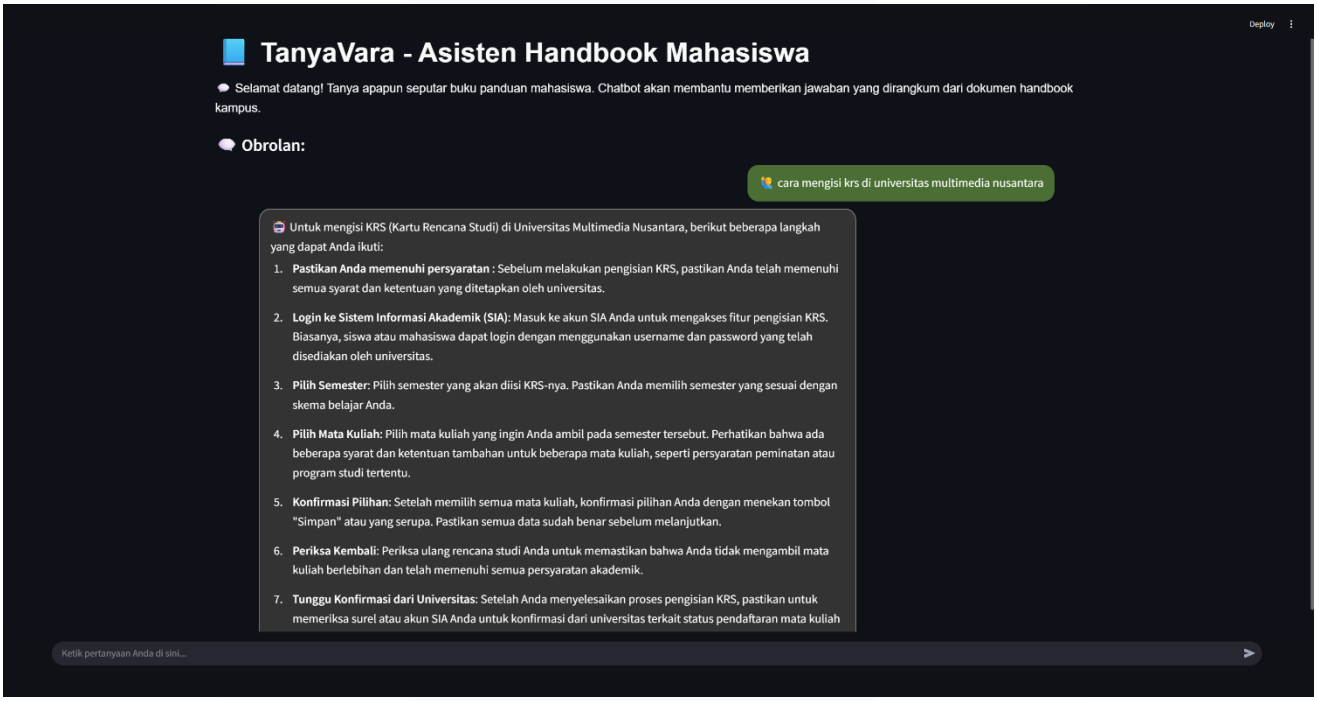
Selain peningkatan data, solusi teknis juga perlu dipertimbangkan. Penggunaan model LLM yang lebih besar dan lebih canggih dengan jumlah parameter yang lebih banyak dapat meningkatkan kemampuan pemahaman dan pemrosesan logis terhadap pertanyaan rumit. Penyesuaian model dengan data internal kampus (fine- tuning) juga menjadi langkah strategis untuk mengadaptasi model agar lebih sesuai dengan domain spesifik. Dalam situasi di mana pendekatan standar tidak memadai, opsi rekonstruksi atau pelatihan khusus menggunakan data kampus dapat dilakukan. Meskipun metode ini memerlukan sumber daya yang lebih besar, hasilnya akan lebih akurat dan dapat diandalkan, terutama aplikasi seperti chatbot yang harus memberikan jawaban yang tepat dan kontekstual.

Secara keseluruhan, pengujian dilakukan dengan membagi pertanyaan ke dalam tiga tingkat kesulitan: sederhana, standar, dan kompleks. Hasil evaluasi menunjukkan bahwa chatbot memiliki skor akurasi tertinggi pada pertanyaan sederhana (rata-rata 4,1 dari 5), sementara pada pertanyaan kompleks akurasinya menurun (sekitar 1,5 dari 5). Hal ini menunjukkan bahwa performa chatbot sangat tergantung pada cakupan dan kualitas dokumen dalam database RAG.

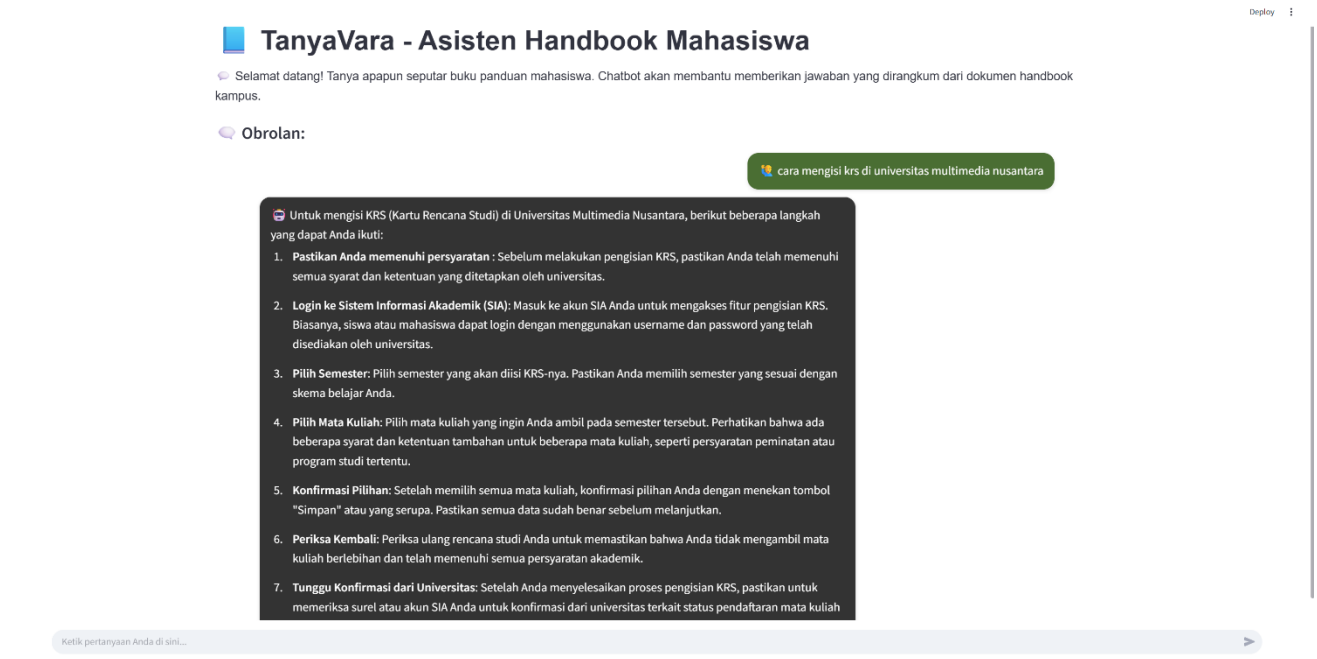
3.3.1.8 Validasi dan Perbaikan

Penulis melakukan validasi lebih lanjut terhadap jawaban yang dihasilkan dengan melakukan pencocokan terhadap dokumen sumber. Jika ditemukan jawaban yang

ambigu, maka dilakukan pengecekan ulang terhadap hasil retrieval dan relevansi dokumen.



Gambar 3.23 Tampilan Website (Mode Gelap)



Gambar 3.24 Tampilan Website (Mode Terang)

Pada gambar 3.23 dan 3.24 diatas, riwayat percakapan antara pengguna dan chatbot dikelola menggunakan session state Streamlit, sehingga setiap pertanyaan dan jawaban dapat ditampilkan kembali secara berurutan. Pengguna dapat mengetik pertanyaan melalui input chat, dan chatbot akan menampilkan jawaban beserta sumber dokumen yang digunakan sebagai referensi. Setiap pesan pengguna dan bot ditampilkan dengan gaya visual yang berbeda, sehingga mudah dibedakan. Sumber dokumen yang digunakan untuk menjawab pertanyaan juga ditampilkan secara jelas, sehingga pengguna dapat menelusuri asal informasi yang diberikan.

3.3.1.9 Dokumentasi dan Penyusunan Laporan

Dilakukan penyusunan laporan kegiatan lengkap berupa perbaikan berbasis hasil uji coba yang dilakukan bersamaan dengan penyusunan dokumentasi teknis yang telah dilampirkan serta pelaporan yang mencakup progres proyek, dan analisis data, hasil evaluasi sebagai pertanggungjawaban akhir. Seluruh proses pekerjaan terdokumentasi dalam bentuk laporan MBKM sejak minggu ke-2 pelaksanaan. Setiap tahap dikonsultasikan kepada dosen pembimbing secara berkala. Selain itu, laporan juga direvisi berdasarkan *feedback* pembimbing untuk menyesuaikan dengan penulisan akademik dan standar evaluasi MBKM di Universitas Multimedia Nusantara.

Secara keseluruhan, kode ini mengintegrasikan proses ekstraksi dan pencarian dokumen, pemrosesan bahasa alami, serta tampilan antarmuka yang interaktif, sehingga menghasilkan sebuah chatbot yang mampu menjawab pertanyaan berbasis dokumen secara cerdas, transparan, dan mudah digunakan oleh pengguna

3.4 Kendala yang Ditemukan

Selama menjalankan penelitian independen ini, saya menemui sejumlah kendala yang cukup signifikan dan memengaruhi kelancaran proses penelitian. Berikut beberapa hambatan yang saya alami:

1. **Keterbatasan Data**

Keterbatasan data menjadi tantangan utama. Mengumpulkan dataset yang benar-benar relevan dan lengkap dari sumber internal kampus memerlukan waktu dan usaha ekstra. Proses ini tidak hanya melibatkan pencarian data, tetapi juga pembersihan dan penyesuaian agar data tersebut sesuai dengan kebutuhan model. Data yang tidak terstruktur atau kurang lengkap dapat menghambat kemampuan model dalam memahami konteks secara tepat.

2. Kompleksitas Pengembangan Model

Kompleksitas pengembangan model juga menjadi hambatan tersendiri. Penerapan metode Retrieval-Augmented Generation (RAG) pada Large Language Model (LLM) tidak selalu berjalan mulus dan memerlukan beberapa kali penyesuaian serta eksperimen agar model dapat menghasilkan jawaban yang akurat dan sesuai konteks. Tantangan ini mencerminkan kebutuhan untuk mengintegrasikan model LLM dengan basis data eksternal secara efektif yang secara teknis cukup kompleks dan membutuhkan trial and error sehingga model dapat menghasilkan jawaban yang akurat dan sesuai konteks.

3. Manajemen Waktu

Membagi waktu antara pengerjaan proyek, konsultasi dengan pembimbing, dan aktivitas lain dapat menjadi aspek yang sulit diatur dan sering kali menimbulkan tekanan, terutama ketika tenggat waktu berdekatan. Hal ini menuntut disiplin dan perencanaan yang matang agar semua tugas dapat diselesaikan tepat waktu tanpa mengorbankan kualitas.

4. Keterbatasan Referensi

Keterbatasan referensi juga menjadi kendala yang signifikan. Mencari literatur dan penelitian terdahulu yang benar-benar relevan dan dapat menjadi landasan teori yang kuat tidak selalu mudah. Proses pencarian ini membutuhkan upaya ekstra untuk menemukan sumber yang tepat dan terpercaya, terutama yang berkaitan dengan teknologi terkini seperti RAG dan LLM, sehingga dasar teori dan metodologi yang digunakan dapat valid dan kuat.

3.5 Solusi atas Kendala yang Ditemukan

Berdasarkan kendala yang telah diuraikan sebelumnya, berikut beberapa solusi yang saya terapkan untuk mengatasi hambatan tersebut:

1. Optimalisasi Pengumpulan Data

Dilakukan pencarian sumber data dengan melakukan cross-check antara handbook dari MyUMN dan situs resmi UMN.

2. Eksperimen dan Iterasi Model

Dalam pengembangan model RAG, beberapa kali dilakukan percobaan dan tuning parameter untuk meningkatkan performa, serta memanfaatkan feedback dari sesi konsultasi dengan pembimbing guna memperbaiki hasil model.

3. Manajemen Waktu yang Terstruktur

Pembuatan jadwal kerja mingguan yang terperinci dan rutin melakukan evaluasi progres bersama pembimbing, sehingga dapat mengatur waktu dengan lebih efektif dan memenuhi tenggat waktu yang telah ditetapkan.

4. Pendalaman Literatur secara Selektif

Dalam mencari referensi, berfokus pada jurnal dan paper yang paling relevan serta menggunakan database akademik terpercaya untuk memperkuat dasar teori dan metodologi proyek.