# 4. IMPLEMENTASI SISTEM

Pada bab ini akan dibahas mengenai implementasi sistem sesuai dengan analisis dan desain sistem yang telah dibahas dalam bab sebelumnya. Implementasi sistem meliputi proses Implementasi perangkat lunak yang digunakan, inisialisasi chatbot, ingesting file, load dokumen, chat engine, respon, dan input output chat.

Tabel 4.1 Segmen Program

| Segmen Program | Flowchart | Nama Segmen | Keterangan Program |
|---|---|---|---|
| 4.4 | 3.14 | Ingesting File | Fungsi untuk mengingest file yang diinput |
| 4.6 | 3.6 | Load File | Load file yang sudah di ingest |
| 4.8 | 3.10 | Respon Chatbot | Kode untuk memberikan respon terhadap pertanyaan pengguna |

## 4.1. Implementasi Perangkat Lunak yang Digunakan

Pembuatan *chatbot* konsultasi rohani kristen berupa website. Website ini menggunakan bahasa pemrograman *Python* versi 3.10.12 dengan bantuan library streamlit yang akan digunakan sebagai antarmuka untuk *chatbot* ini. Program ini juga dijalankan di dalam virtual environment sehingga dapat lebih mudah dalam menginstal library-library python yang diperlukan dalam pengerjaan chatbot ini.

C*hatbot* ini menggunakan Qdrant untuk menyimpan data yang telah di ingest ke dalam qdrant agar memudahkan dalam menyimpan dan melakukan pencarian vector dalam ruang berdimensi tinggi.

Qdrant menggunakan url dan api key untuk store data yang di ingest sehingga dalam data di implementasikan agar bisa digunakan dengan lebih mudah.

Segmen Program 4. 1 Inisialisasi Secrets Streamlit

```
[qdrant]
url = https://d1dcc1d9-6aa5-40eb-938e-6d98354d9e5a.us-east4-0.gcp.cloud.qdrant.io:6333
api_key ="cXUWqd8Hfawr35dfRp7qZumGuqRloNkuSzZ5cwNSBJKe57Hd6mSSA"
```

**4.2. Inisialisasi Chatbot**

Pada proses ini adalah proses untuk inisialisasi chatbot dengan menginisialisasi models, embedding, dan prompt yang akan digunakan oleh chatbot.

### 4.2.1. Inisialisasi Models dan Embedding

Pada proses ini dilakukan inisialisasi chatbot dimana chatbot ini menggunakan Ollama dan model open source yaitu adalah openchat dan inisialisasi embedding model yang menggunakan intfloat/multilingual-e5-large. embedding model ini digunakan karena sudah mendukung bahasa Indonesia. Base url merupakan tempat ollama berada sehingga pengguna tidak perlu melakukan ollama serve ketika menjalankan program dan ollama dapat running tanpa dipanggil.

Segmen Program 4.2 Inisialisasi Model dan Embedding

```
Settings.llm = Ollama(model='openchat:latest',
base_url="http://172.17.0.1:11434")
Settings.embed_model =
FastEmbedEmbedding(model_name="intfloat/multilingual-e5-large",
cache_dir="../cache")
```

Embedding model ini berguna untuk mengubah text menjadi representasi numerik yang bertujuan untuk menangkap makna teks untuk meningkatkan efisiensi penelusuran dan mencari teks yang serupa dibandingkan dengan teks mentah yang tidak diproses menjadi representasi numerik dengan menggunakan embedding model. Cache adalah tempat dimana model dapat menyimpan dan mengambil cache data seperti embedding yang sudah di download agar bisa digunakan nanti ketika diperlukan tanpa memproses data yang sama untuk menghemat waktu dan resource yang akan digunakan untuk menjalankannya Kembali.

Contoh representasi numerik:

```
tensor([[ 0.0075, -0.0123,  0.0189,  0.0056, -0.0032,  0.0214, -
0.0117,  0.0048,  0.0126,  0.0159, ...]])
```

### 4.2.2. Inisialisasi Prompt

Pada proses ini dilakukan inisialisasi prompt dimana prompt bertujuan untuk menginstruksikan model tersebut tentang apa saja yang harus dilakukan, apa saja yang bisa dilakukan, apa saja yang tidak boleh dilakukan kepada pengguna sesuai dengan apa yang ditulis di dalam prompt tersebut.

Segmen Program 4.3 Inisialisasi Prompt

```
Settings.system_prompt = """ Anda adalah asisten digital ahli
dalam konsultasi rohani KRISTEN yang berperan menjadi penasihat
rohani KRISTEN yang fasih berbahasa indonesia dan memiliki
pengetahuan yang luas dalam dunia konsultasi rohani KRISTEN dan
hanya akan menjawab user berdasarkan data yang valid dengan
menggunakan bahasa indonesia. Tugas utama anda adalah melakukan
percakapan dengan user, mendengarkan keluh kesah mereka, atau
menjawab pertanyaan dari user dengan memberikan pandanganan yang
didasarkan pada prinsip-prinsip iman KRISTEN. Sebagai penasihat
rohani KRISTEN, hal pertama yang paling penting ketika user
berbagi cerita atau perasaan kepada anda, anda akan melanjutkan
percakapan dengan memberikan beberapa pertanyaan atas jawaban
user sampai anda memahami konteks keluhan user secara rinci,
termasuk alur cerita, penyebab, akibat, penanganan saat ini dan
aspek lainnya yang menjadi bagian dari cerita mereka. Anda akan
melakukan ini dengan cara memberikan pertanyaan, mendengarkan
jawaban mereka, dan bertanya lebih lanjut hingga Anda merasa
memahami situasinya secara lengkap untuk dapat memberikan
bantuan yang sesuai. Dalam percakapan, Anda akan bersikap empati
tanpa menghakimi. Jika pengguna melakukan kesalahan, Anda akan
mencoba mempengaruhi mereka secara halus. Jika Anda tidak
mengetahui jawabannya, atau bahkan setelah menanyakan lebih
lanjut Anda masih tidak tahu jawabannya, Anda akan mengakui
ketidaktahuan Anda dan tetap memberikan dukungan dan empati
kepada pengguna. Anda akan menggunakan bahasa yang ramah dan
sesuai dengan pengguna. Jawaban, solusi, dan saran yang Anda
tawarkan akan diprioritaskan dalam bahasa yang mudah dimengerti
```

```
dan didasarkan pada unsur Alkitab, rohani, Kristen, dan sains
dalam pengetahuan Anda. Penting untuk menghindari memberikan
asumsi atau solusi yang prematur kepada pengguna jika konteks
mereka tidak cukup rinci dan komprehensif. Anda akan selalu
bertindak dengan ramah dan penuh pengertian terhadap user. Anda
akan mendengarkan dengan sabar dan menjawab dengan menggunakan
bahasa yang sopan, memperlakukan semua user dengan hormat dan
setara tanpa adanya perbedaan apapun. Anda akan melakukan
percakapan seperti yang dijelaskan di atas sebelum memberikan
solusi yang Anda anggap pantas."""
```

Prompt digunakan untuk memerintah model yang digunakan dari chatbot untuk apa saja yang perlu dilakukan untuk berinteraksi dengan pengguna. Prompt diatas berisi perintah untuk model agar bertindak sebagai penasihat rohani kristen yang fasih berbahasa Indonesia. Prompt juga memerintah untuk menjawab pengguna berdasarkan data yang valid. Selain itu tugas utama yang diberikan adalah untuk melakukan percakapan dengan pengguna dan menjawab pertanyaan yang diberikan oleh pengguna.

Prompt dibuat berdasarkan pertimbangan-pertimbangan seperti untuk menjadi pembimbing konselor rohani bagi pengguna. Prompt juga menekankan agar menjawab pertanyaan pengguna dengan menggunakan bahasa Indonesia karena ketika tidak menggunakannya maka chatbot bisa menjawab menggunakan bahasa inggris.

### 4.3. Ingesting File

Proses ini adalah proses untuk mengupload file yang akan digunakan oleh chatbot sebagai sumber data untuk menjawab pertanyaan-pertanyaan yang akan diajukan oleh user kepada chatbot dan chabot tersebut akan mencari jawaban yang sesuai dengan konteks pertanyaan tersebut dengan data yang sudah di ingest.

Segmen Program 4.4 Upload File

```
def upload_files(files, path):
    files_path = []
    for file in files:
        try:
            save_path = os.path.join(path, file.name)
            if os.path.exists(save_path):
                st.toast("{} already exists!".format(file.name),
icon="⚠")
            else:
                with open(save_path, "wb") as f:
```

```
                    f.write(file.getvalue())
                    files_path.append(file.name)
                    indexing_data(path, file.name)
        except Exception as e:
            st.error(f"Error saving file: {e}")
            return None
    files_path = ", ".join(files_path)
    if files_path:
        print("Before")
        st.success("Successfully uploaded
{}".format(files_path))
```

Pada proses ini file akan diupload ke dalam folder yang sudah ditentukan ketika menggunakan upload_files. File tersebut akan disimpan pada folder yang ditentukan yang kemudian nanti akan digunakan ketika chatbot dijalankan untuk menjawab pertanyaan yang ditanyakan oleh pengguna.

Segmen Program 4.5 Qdrant

```
def create_collection(documents, collection_name):
    if not st.session_state.chatbot:
        client = QdrantClient(url=st.secrets["qdrant"]["url"],
api_key=st.secrets["qdrant"]["api_key"])
    else:
        chatbot = st.session_state.chatbot
        client = chatbot.client
    vector_store = QdrantVectorStore(client=client,
collection_name=collection_name)
    storage_context =
StorageContext.from_defaults(vector_store=vector_store)
    index = VectorStoreIndex.from_documents(documents,
storage_context=storage_context)
    return index
```

Proses ini adalah proses untuk mengupload file ke dalam qdrant QdrantClient digunakan untuk menyambungkan koneksi ke layanan Qdrant. Untuk mengambil url yang ada di dalam tempat rahasia stremalit menggunakan st.secrets untuk mengakses tempat rahasia tersebut dengan mencari kata kuncinya yaitu qdrant dan mengambil url yang terdapat dalam tempat rahasia tersebut dan menyimpannya ke dalam url. Untuk mengambil api key juga digunakan cara

yang sama yaitu menggunakan st.secrets untuk mengakses tempat rahasia streamlit tempat dimana api key tersebut disimpan dan mencari kata kunci yaitu qdrant dan mengambil api key yang tersimpan di dalam tempat rahasia tersebut yaitu *.streamlit*.

### 4.4. Load Documents

Segmen Program 4.6 Load Dokumen

```
client = QdrantClient(url=st.secrets["qdrant"]["url"], api_key=st.secrets["qdrant"]["api_key"])
vector_store = QdrantVectorStore(client=client, collection_name="Documents")
index = VectorStoreIndex.from_vector_store(vector_store=vector_store)
```

Pada proses ini dilakukan proses untuk load dokumen yang akan digunakan sebagai sumber data untuk chatbot konsultasi rohani kristen dengan mengakses tempat rahasia .streamlit kemudian menggunakan url dan api key tersebut untuk membuat koneksi dengan qdrant dengan QdrantClient. Kemudian melakukan inisialisasi vector store untuk menyimpan data vector dan menggunakan VectorStoreIndex untuk membuat index dari vector store yang berada di atasnya untuk membuat indeks untuk mempercepat pencarian002E

### 4.5. Chat Engine

Segmen Program 4.7 Chat Engine

```
def create_chat_engine(self, index):
        self.chat_store = SimpleChatStore()
        memory =
ChatMemoryBuffer.from_defaults(chat_store=self.chat_store,
token_limit=3000)
        return
index.as_chat_engine(chat_mode="condense_plus_context",
memory=memory,
                                    llm=self.Settings.llm,
                                    context_prompt=(
                                        "Petrabot adalah sebuah
chatbot, dapat berinteraksi dengan normal, menjawab dengan
menggunakan bahasa indonesia, dan "
                                        "berpengetahuan dalam
Konsultasi Rohani Kristen. Informasi konteks terdapat dibawah."
```

```
                                        "---------------------
\n"
                                        "{context_str}\n"
                                        "---------------------
\n"
                                        "Mengingat konteks dan
bukan pengetahuan sebelumnya, jawablah pertanyaan sesingkat
mungkin \n"
                                        "Instruksi: Gunakan
riwayat obrolan sebelumnya, "
                                        "atau konteks diatas,
untuk berinteraksi dan membantu pengguna."
                                        "Jika ada pertanyaan
yang tidak ada di dalam konteks arahkan pengguna untuk
menghubungi Pusat Kerohanian"
                                        "Universitas Kristen
Petra dengan nomor (031)-298-3197 atau email
pusroh@petra.ac.id."), verbose=True)
```

Fungsi create_chat_engine adalah fungsi yang digunakan untuk membuat mesin obrolan dimana terdapat SimpleChatStore untuk menyimpan Riwayat obrolan pada sesi ketika engine tersebut dijalankan. Context prompt adalah pesan yang digunakan untuk memberi instruksi kepada chatbot ketika sedang berinteraksi dengan pengguna. Context prompt berisi tentang fitur dan fungsi chatbot, serta memberikan tata cara yang diinginkan ketika chatbot tersebut berinteraksi dengan pengguna dan respon seperti apa yang harus diberikan kepada pengguna.

**4.6 Respon**

Segmen Program 4.8 Respon Chatbot

```
def stream_response_generator(self, prompt):
        response = self.chat_engine.stream_chat(prompt)
        for token in response.response_gen:
            yield token + ""
            time.sleep(0.025)
        self.chat_store.persist(persist_path="chat_store.json")
```

Kode ini berfungsi untuk menghasilkan respon dari chat engine sesuai dengan prompt yang pengguna masukkan, setiap respon disampaikan satu per satu melalui yield yang memastiken semua respon adalah string dengan operasi +””. Dengan menggunakan generator setiap respon akan dikenakan jeda waktu sebelum token selanjutnya dihasilkan yang akan di kirimkan ke pengguna dengan laju yang terkontrol.

### 4.7 Input Output Chat

Segmen Program 4.9 Input Output Chat

```
prompt = st.chat_input('Question')

if prompt:
    st.chat_message('user').markdown(prompt)
    st.session_state.messages.append({'role': 'user', 'content':
prompt})
    response = chatbot.stream_response_generator(prompt)
    st.chat_message('assistant').markdown(response)
    st.session_state.messages.append({'role': 'assistant',
'content': response})
```

Kode ini adalah kode untuk memberikan input pertanyaan kepada pengguna kepada chatbot konsultasi Rohani Kristen. Kode akan mengirimkan input dari pengguna kepada fungsi penghasil respon, fungsi tersebut akan mengirimkan input ke chat engine dan kemudian menerima respon dari chat engine tersebut. Respon ini kemudian dikembalikan kepada pengguna secara bertahap dengan jeda waktu sesuai dengan generator yang sudah di setel dengan jeda waktu sebesar 25 milidetik. Hal ini memungkinkan chatbot tersebut mengirimkan respon kepada pengguna secara bertahap dan memberikan respon yang lebih alami dan terkontrol.