docAI

a Document-Understanding AI Assistant

This repository contains the code for a Document-Understanding AI Assistant. The system is built using Python, FastApi, Langchain, FAISS, Sqlite, Sqlachemy, Chainlit, Huggingface Embedding Model and an OpenAI's Language Model. Docker Compose is used to easily spin up the required services in Docker containers.

Table of Contents

Project Objective

Develop an interactive AI assistant capable of understanding and answering questions about content from PDF or CSV files.

System Components

- **Frontend(UI)**: Built using Chainlit, a modern framework for building AI chat interfaces.
- **Backend(API)**: Built using FastApi, a modern framework for building web APIs.
- **AI-Framework**: Utilizes Langchain, a framework for developing applications powered by language models.
- **Vector Database**: FAISS (Facebook AI Similarity Search) is used for efficient similarity search and retrieval of document embeddings.
- **Relational Database**: Sqlite is used as the database to store chat history.
- **Embedding Model**: Huggingface Embedding Model is used to convert text data into numerical vectors.
- **Language Model**: OpenAI's Language Model is used to generate responses based on the input queries.
- **Sqlachemy**: Used for database operations.
- **Deployment**: Docker Compose is used to easily spin up the required services in Docker containers.

Setup Instructions

Requirements

- Python 3.10 and above
- Docker (optional)

- `fastapi[standard]`
- `langchain-openai`
- `langchain-groq` (optional)
- `langchain`
- `langchain-community`
- `langchain-huggingface`
- `pydantic-settings`
- `python-multipart`
- `pypdf`
- `python-dotenv`
- `faiss-cpu`
- `pydantic==2.9.2`
- `sqlalchemy`
- `chainlit`
- `requests`

Setup Instructions

1. **Clone the repository**:

```
git clone https://github.com/felixLandlord/docAI.git
cd docAI
```

2. **Create a virtual environment**:

```
python -m venv .venv
```

3. **Activate the virtual environment**:

MacOS:

```
source .venv/bin/activate
```

Windows:

```
.venv\Scripts\activate
```

Linux:

```
source .venv/bin/activate
```

4. **Install the required dependencies**:

```
pip install -r requirements.txt
```

5. **Set up environment variables**:

Create a `.env` file in the root directory of the project and add the following variables:

```
OPENAI_API_KEY = "your_openai_api_key"
HUGGINGFACE_KEY = "your_huggingface_token"

#local
DATABASE_URL =
"sqlite:///./backend/app/db/sqlite_history/chat_history.db"
FAISS_INDEX_DIR = "backend/app/db/faiss_index"

# docker (optional - already specified in docker-compose)
# DATABASE_URL="sqlite:///app/db/sqlite_history/chat_history.db"
# FAISS_INDEX_DIR="app/db/faiss_index"


# Chainlit UI

#local
API_BASE_URL = "http://localhost:8000"

# docker (optional - already specified in docker-compose)
# API_BASE_URL="http://fastapi_backend:8000"
```

you can use a free language model with the Groq API, just replace **OPENAI_KEY=your_openai_key** with **GROQ_KEY = "your_groq_key"**

6. Running the application:

Without Docker

i. Start the FastApi application:

To run in development mode:

```
fastapi dev backend/app/main.py
```

To run in production:

```
fastapi run backend/app/main.py
```

This will start the FastApi application on *http://localhost:8000/*

ii. Run the Chainlit web app:
To run in development mode:

```
chainlit run frontend/app.py --port 8001 -w
```

To run in production:

```
chainlit run frontend/app.py --port 8001
```

This will start the Chainlit web application on *http://localhost:8001/*

With Docker
Make sure the Docker daemon is running by opening the docker application

```
docker-compose up -d
```

This will start a docker container with FastApi service (port 8000) and Chainlit service (port 8001)

Modifications to use GROQ(free) instead of OPENAI

- Make sure `langchain-groq` is installed
- In the `.env` file, replace `OPENAI_KEY` with `GROQ_KEY`
- In the `backend/app/core/config.py` file, replace `openai_key` with `groq_key`
- In the `backend/app/core/config.py` file, replace `openai_llm_name` with `groq_llm_name`
- In the `backend/app/core/chains.py` file, replace `ChatOpenAI` import with `ChatGroq` import
- In the `backend/app/core/chains.py` file, replace the `OpenAI` **get_llm function** with `Groq` **get_llm function**
- If Using Docker, In the `docker-compose.yaml` file, replace `OPENAI_KEY` with `GROQ_KEY`

Usage

Document Upload

- Upload one or more PDF documents to be processed into the vector store.
- The chat UI will display a success message upon successful upload.
- If invalid documents are provided, you'll receive a prompt to upload valid PDF files.

Querying

- After successful document upload, you can start sending queries
- The system will process your query and provide relevant responses based on the uploaded documents

Chat Management

- To start fresh, click the "New Chat" button in the top-right corner
- This will clear all previous conversations and uploads

Steps Summary

1. Upload PDF document(s)
2. Wait for success confirmation
3. Start asking questions about your documents
4. Use "New Chat" to reset everything

Note: Make sure to only upload PDF documents as other file formats are not supported as of this moment.

Design Choice

The design of docAI is centered around modularity, scalability, and ease of use. Each component was selected to ensure that the system can efficiently handle document processing and querying with good accuracies and speeds.
The design choices aim to balance performance, modularity, and ease of use while allowing for future extensibility and integration with other tools or models.

- Frontend (Chainlit) and Backend (FastAPI) separation allows for independent development and scaling of each service. Chainlit, used for the frontend, is designed to provide a modern, dynamic user experience tailored for AI Chat applications, while FastAPI is optimized for fast, high-performance backend API creation.

- By using Langchain, which integrates various language model operations, it allows flexibility in switching between or incorporating multiple language models and embeddings without significant code changes.

- FAISS (Facebook AI Similarity Search) is an optimized library for similarity search that is allows for quickly retrieving relevant text passages from large document embeddings. This is particularly useful for searching across multiple documents with high-speed response times.

- The Huggingface Model is used to generate embeddings that transform text into vectors, enabling efficient similarity search via FAISS. The sentence-transformers/all-mpnet-base-v2 provides high quality embeddings with dimension of 768. OpenAI’s Language Model for high-quality responses. Optionally, the Groq API can be used as a free alternative to OpenAI’s model, offering flexibility in model selection based on cost, response quality, and latency needs.

- SQLite is a lightweight relational database allows for storing chat history and user interactions, enabling future analysis or user experience improvements without complex infrastructure. SQLAlchemy further simplifies the ORM (Object-Relational Mapping) processes and improves development speed by abstracting SQL commands.

- Docker Compose simplifies the deployment and scalability of the system by packaging the entire stack (FastAPI, Chainlit, databases, etc.) into isolated containers. This also ensures that the environment remains consistent across different setups.
- The system provides an intuitive interface through Chainlit for uploading documents and querying them, with real-time feedback and chat management features like "New Chat" to reset conversations. This design enhances user interaction and usability, making it accessible to both technical and non-technical users.

API Key Guide

OPENAI API KEY

Step 1: Go to OpenAI's website.

Step 2: Log in to your OpenAI account. If you don’t have one, sign up for a new account.

Step 3: After logging in, click on the settings icon in the top-right corner and go to API Keys.

Step 4: Click on Create new secret key. A new API key will be generated.

Step 5: Copy and securely save this API key. (You will not be able to see it again after leaving this page.)

HUGGINGFACE TOKEN

Step 1: Go to HuggingFace's website.

Step 2: Log in to your account, or sign up if you don’t have one.

Step 3: Once logged in, click on your profile icon in the top-right corner and go to Settings.

Step 4: In the Settings menu, click on Access Tokens.

Step 5: Click on New Token to generate a new API key. SELECT token type "Read".

Step 6: Copy the token and save it securely.

OPENAI API KEY

Step 1: Go to Groq’s website. Look for a link or section about developers or API access. Groq's API availability may vary, so check if registration is needed.

Step 2: Sign up for a developer account if required, or log in if you already have an account.

Step 3: After logging in, navigate to your account settings or API access area.

Step 4: Find the option to Create API Key and click it. A new API key will be generated.

Step 5: Copy and securely save the key for your records.

Screenshots


dashboard1.png

dashboard2.png

dashboard3.png