Medical Chatbot RAG Implementation

LLM: OpenAI
Embeddings: OpenAI Embeddings
VectorDB: FAISS db
Frontend: streamlit

project utilizes Retrieval-Augmented Generation (RAG) to create a highly efficient medical chatbot. Here's a step-by-step breakdown of the process:

Upload PDF: The first step involves uploading a PDF document containing the necessary medical information. This document serves as the primary source of data for the chatbot.

PDF Chunking: Once the PDF is uploaded, the document undergoes a chunking process. This involves splitting the PDF into smaller, manageable chunks of text to facilitate easier processing and analysis.

Embedding Generation: After chunking, each text chunk is converted into embeddings using OpenAI Embeddings. These embeddings represent the semantic meaning of the text in a numerical format, which is essential for efficient retrieval and processing.

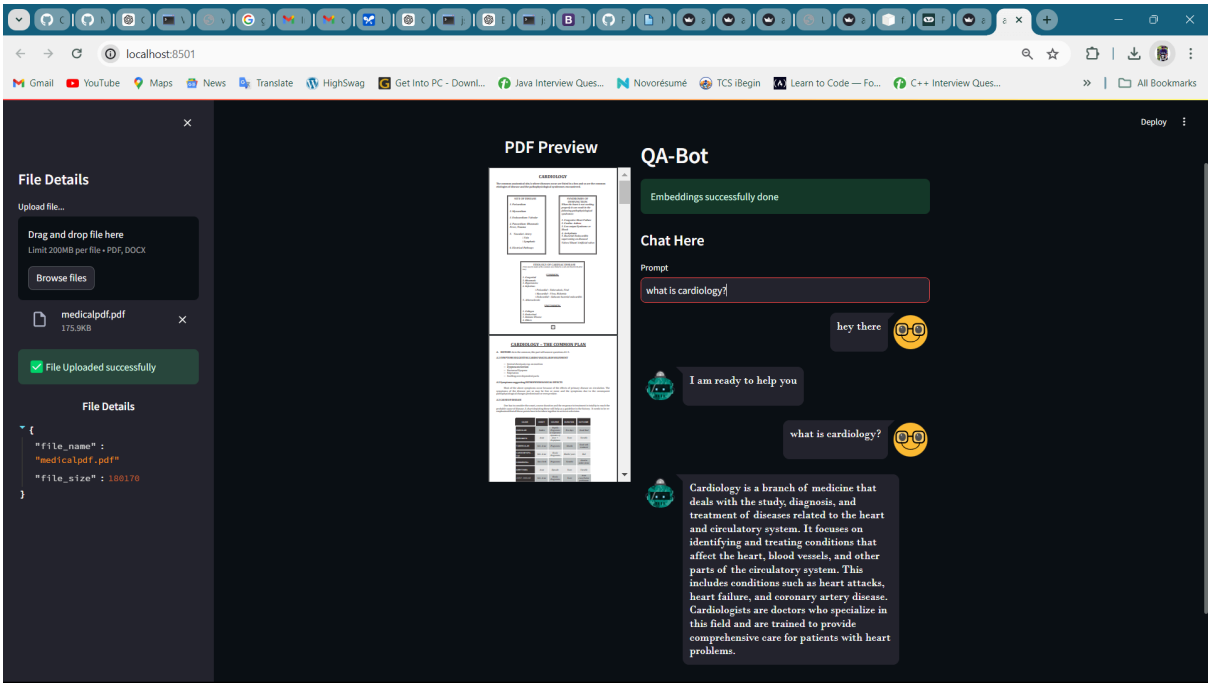
Storing Embeddings in VectorDB: The generated embeddings are stored in a FAISS (Facebook AI Similarity Search) database, a highly optimized vector database designed for similarity search and clustering of dense vectors.

RetrievalQA Chain for Query Processing: When a user asks a question, the query is first converted into a vector format. The RetrievalQA chain then retrieves the most relevant data from the FAISS vector database using similarity search.

Response Generation: The retrieved information is used to generate a precise and relevant response to the user's query, leveraging the capabilities of OpenAI’s language model (LLM).

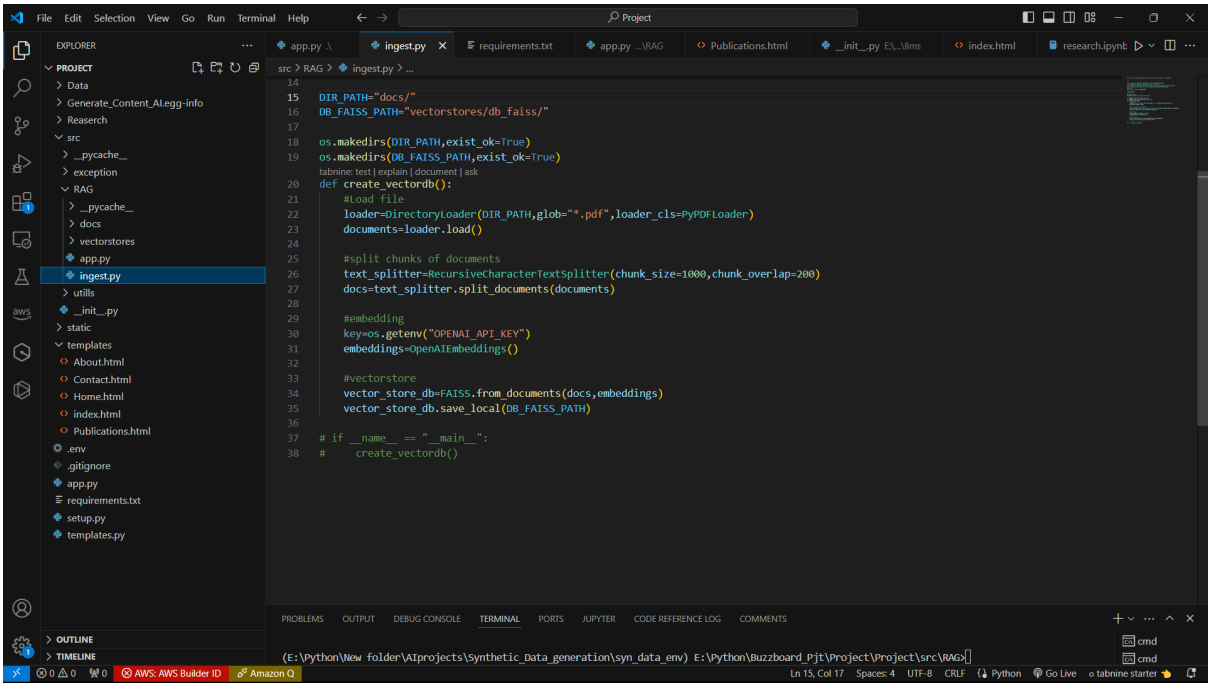
This RAG implementation ensures that our medical chatbot provides accurate, contextually relevant answers by effectively combining retrieval and generation techniques

Output:



Below Code handling the DB creation :

path **src/RAG/ingest.py**



This code handling Retrieval data from vector db

src/RAG/app.py