Mini LLM-Powered Question Answering System Using RAG

Problem Statement

In healthcare, legal, and compliance-heavy industries, professionals must extract specific information from large, dense PDF documents like diagnostic codes, clinical reports, and regulatory guidelines. Manual searching is inefficient and error-prone. This project addresses this by creating a lightweight, retrieval-augmented generation (RAG) based question-answering system using large language models (LLMs).

Step-by-Step Implementation

The project starts by installing some important tools like PyMuPDF (to read PDFs), FAISS (to search similar text), LangChain (to help build the QA system), and Gradio (to make a simple web interface).

Next, we bring in useful components:

- **PyMuPDF** helps us pull out text from PDF files.
- **FAISS** helps find pieces of text that are similar to a user's question.
- **SentenceTransformers** turns text into numbers (embeddings) that the computer can understand.
- **Transformers** gives us a language model (FLAN-T5) that can answer questions.
- **LangChain** helps us connect all the parts together.
- **Gradio** is used to build a user-friendly webpage for uploading PDFs and asking questions.

We use two main models:

- **all-MiniLM-L6-v2** to convert text into embeddings (vectors).
- **google/flan-t5-small** to generate answers based on the user's question and related text.

When a user uploads a PDF, the program reads all the text from the file. Then, it breaks that text into smaller overlapping parts, so it can better understand what each part means.

Each of those parts is turned into vectors using the embedding model and saved in a special searchable format called a FAISS index.

When the user asks a question, the system turns the question into a vector too, and then searches for the most similar pieces of text from the PDF.

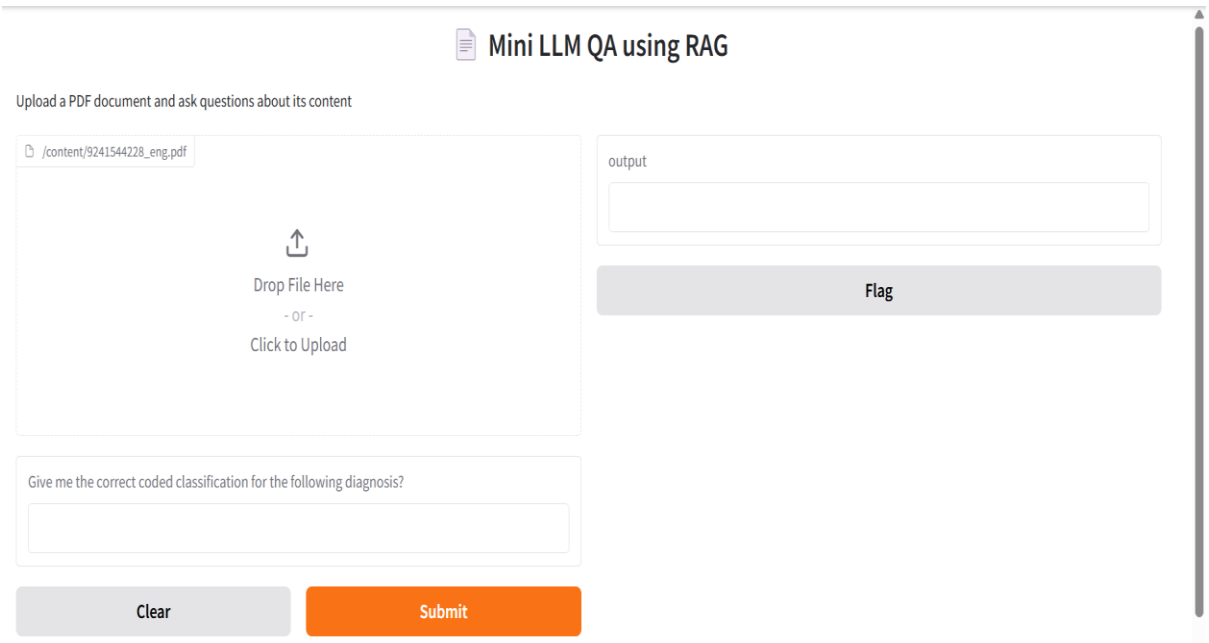
Finally, it sends those pieces, along with the question, to the FLAN-T5 model to generate a clear answer using a method called **Retrieval-Augmented Generation (RAG)**.

Tools & Models Used

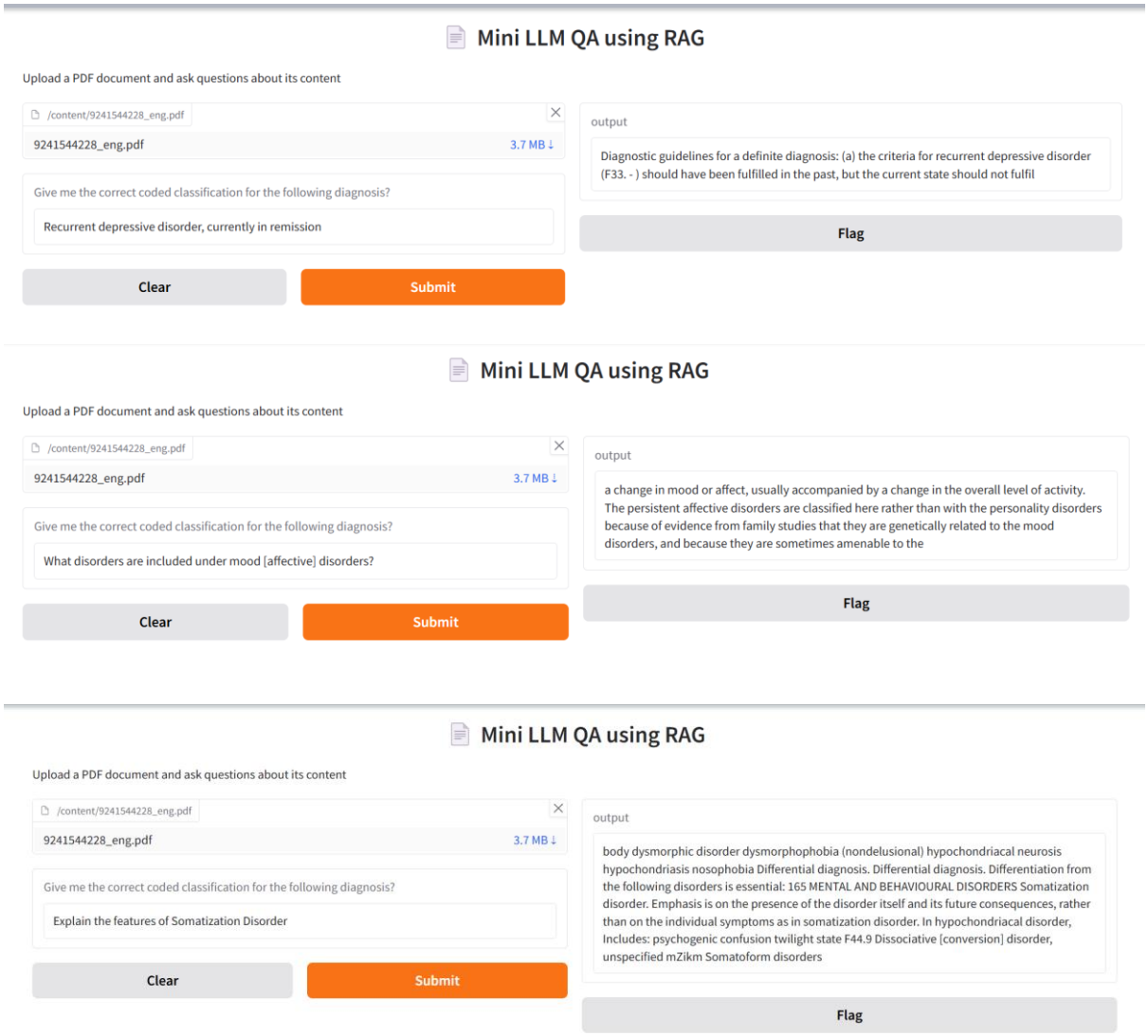
Tool/Library	Purpose
PyMuPDF (fitz)	PDF text extraction
FAISS	Vector similarity search
SentenceTransformers	Embedding generation
Transformers (Hugging Face)	LLM model (FLAN-T5-small)
LangChain	Modular LLM chaining (RAG integration)
Gradio	Web UI deployment

Use of AI Tools

- **ChatGPT**: Generated and debugged initial boilerplate code, FAISS logic
- **Hugging Face**: Provided pretrained models (FLAN-T5) and tokenizer
- **LangChain**: Integrated embedding retrieval and prompt handling
- The Gradio interface shown in your screenshot is a simple **Mini LLM QA system using RAG (Retrieval-Augmented Generation)**



Sample Output



Conclusion

This mini-project showcases an efficient and lightweight RAG-based QA system using Python and open-source models. It supports real-time semantic search and LLM-based question answering for user-uploaded documents. With future enhancements such as larger LLMs and multi-document indexing, it can evolve into a production-ready enterprise-grade search assistant.

Google Cloab Link

https://colab.research.google.com/drive/1YWxaIsW0rHUUlYlmA0Hcxn3SlDderUdS?usp=sharing