Document QA System with Django Backend

Introduction

This report details the implementation of a document-based question-answering system with a robust backend using Django. The system processes uploaded PDF documents, extracts relevant information, and provides answers to user queries. It leverages a vector database for efficient retrieval and response generation.

System Overview

The system consists of the following key components:

- A Django-based backend for handling file uploads, query processing, and response generation.
- A vector database to store processed PDF content for fast retrieval.
- An AI-powered model to generate responses based on retrieved context.

PDF Processing and Storage

Upon receiving a PDF file, the system:

1. Clears any previously uploaded PDFs to ensure fresh processing.
2. Extracts text and relevant content from the document.
3. Splits the text into manageable chunks for efficient retrieval.
4. Stores these processed chunks in a vector database using Sentence Transformers for embedding generation.

Query Handling and Answer Retrieval

When a user submits a query:

1. The system retrieves relevant content from the vector database.
2. It passes the retrieved information to an AI-powered model.
3. The model generates a structured response based on the retrieved context.

Technical Implementation

- The backend is implemented using Django, ensuring a robust and scalable architecture.
- FAISS is used as the vector database to enable fast similarity-based retrieval.
- Sentence Transformers are utilized to generate embeddings for document chunks.
- The Mixtral-8x7B-32768 model is used for generating accurate responses.
- GROP is integrated for faster model inference.

Key Considerations

- The system is designed to efficiently process and retrieve answers from uploaded documents.

- The vector database enables quick lookups without requiring a full document scan.
- The AI model is optimized for structured and concise answers.

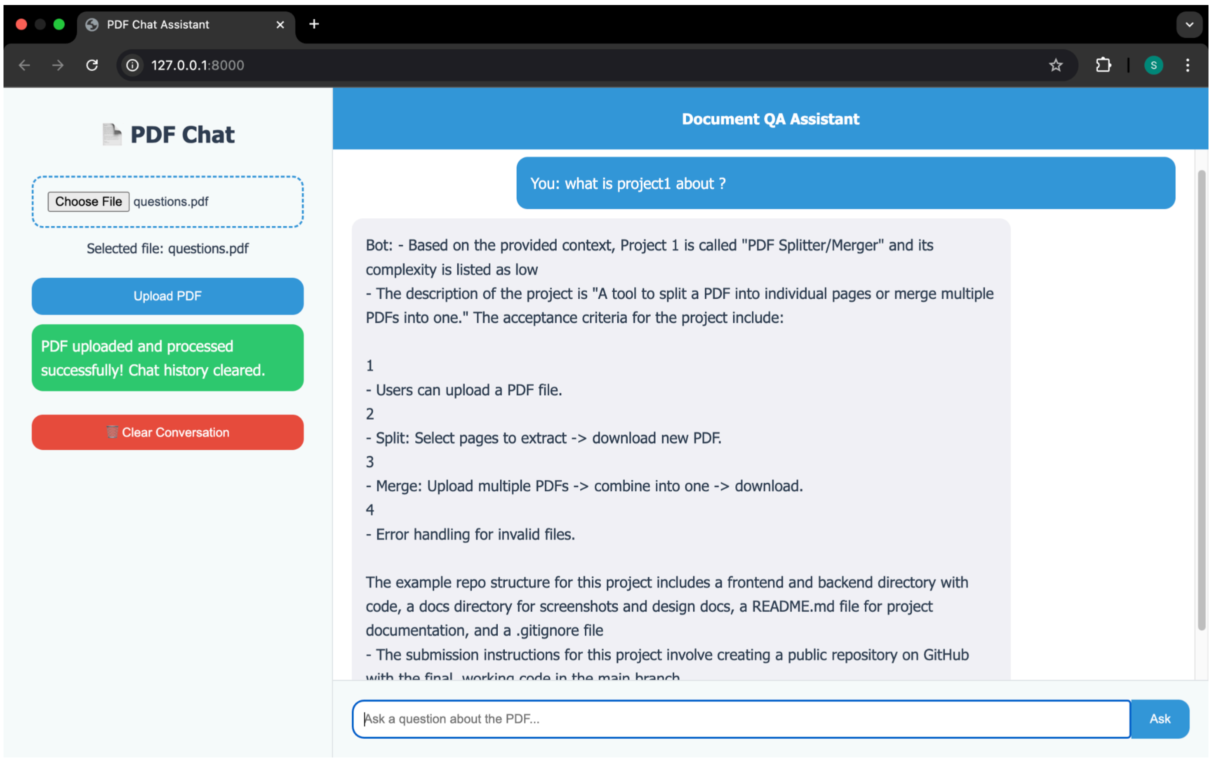
Personal Note

I am into AI solutions and providing a backend with Django. However, I do not have experience with frontend development.

Conclusion

This document QA system provides an efficient way to extract and answer queries from PDFs. The integration of a robust Django backend with a vector-based retrieval mechanism ensures accurate and fast responses to user questions.

Interface :


PDF Chat Assistant
127.0.0.1:8000
PDF Chat
Choose File
questions.pdf
Selected file: questions.pdf
Upload PDF
PDF uploaded and processed successfully! Chat history cleared.
Clear Conversation
Document QA Assistant
Ask a question about the PDF...
Ask

You: what is project1 about ?

Bot: - Based on the provided context, Project 1 is called "PDF Splitter/Merger" and its complexity is listed as low
- The description of the project is "A tool to split a PDF into individual pages or merge multiple PDFs into one." The acceptance criteria for the project include:

1
- Users can upload a PDF file.
2
- Split: Select pages to extract -> download new PDF.
3
- Merge: Upload multiple PDFs -> combine into one -> download.
4
- Error handling for invalid files.

The example repo structure for this project includes a frontend and backend directory with code, a docs directory for screenshots and design docs, a README.md file for project documentation, and a .gitignore file
- The submission instructions for this project involve creating a public repository on GitHub