Chat with PDF Project Documentation

Overall Approach

The project involves creating a conversational interface using Streamlit that allows users to interact with PDF documents. The main steps include:

1. Uploading a PDF document.
2. Extracting text from the uploaded PDF.
3. Embedding the text using Google Generative AI embeddings.
4. Storing the embeddings in a FAISS vector store.
5. Utilizing LangChain to manage conversation history and context.
6. Generating responses based on user queries and the content of the PDF using Google Generative AI.

Frameworks/Libraries/Tools Used

1. **Streamlit**: Used for creating the web interface where users can upload PDFs and interact with the chatbot.
2. **PyPDF2**: Utilized for extracting text from PDF documents.
3. **dotenv**: Used for managing environment variables securely.
4. **Google Generative AI**: Powers the natural language understanding and response generation using the Gemini 1.5 Flash model.
5. **LangChain**: Manages conversation memory to maintain context during interactions.
6. **FAISS**: Stores and retrieves document embeddings efficiently for quick access to relevant information.

Problems Faced and Solutions

1. **Handling Large PDFs**: Extracting text from large PDF files was initially slow. We optimized the extraction process by processing one page at a time.
2. **Maintaining Context in Conversations**: Keeping track of conversation history and context was challenging. We used LangChain's ConversationSummaryMemory to manage this effectively.
3. **Efficient Embedding and Retrieval**: Storing and retrieving embeddings quickly required an efficient solution. FAISS was implemented to handle this, providing fast and accurate retrieval.

Future Scope

1. **Multi-Document Support**: Extending the application to handle multiple PDFs simultaneously, allowing cross-document queries.
2. **Enhanced UI/UX**: Improving the user interface for a more intuitive experience, including better visualization of document content.
3. **Additional AI Models**: Integrating other AI models to provide more diverse and specialized responses.
4. **Advanced Query Capabilities**: Adding features like keyword highlighting, summarization, and topic detection to provide more comprehensive answers.
5. **User Authentication and Personalization**: Implementing user authentication to save and personalize interactions based on user preferences and history.