Final Report

Problem Statement:

Retrieval Augmented Generation (RAG) with LangChain

The aim is to develop a system capable of extracting information from a PDF file and providing concise, accurate answers to user queries based on the content. The challenge lies in handling large documents, ensuring efficient retrieval, and generating relevant responses.

Objective:

1. **PDF Handling**: The system should accept a PDF file as input and extract its textual content.
2. **Text Chunking**: The extracted text must be divided into manageable chunks to facilitate efficient search and retrieval.
3. **Embedding**: The text chunks need to be converted into embeddings for easy comparison and retrieval.
4. **Question Answering**: Using a language model, the system should answer questions based on the retrieved chunks.
5. **User Interaction**: Provide an interface for users to upload PDFs and ask questions.

Methodology:

1. **Environment Setup**:
    - Set the OpenAI API key for accessing the language model.
    - Import necessary libraries, including PyPDF2 for PDF handling, langchain modules for processing, and Chroma for vector store.
2. **PDF Loading and Text Extraction**:
    - Load the PDF using PyPDF2.PdfReader and iterate over each page to extract the text.
3. **Text Chunking**:
    - Use RecursiveCharacterTextSplitter to divide the extracted text into chunks of 1000 characters, with an overlap of 100 characters to maintain context.
4. **Embedding and ChromaDB**:
    - Create embeddings for each chunk using OpenAIEmbeddings.
    - Store the embeddings in Chroma, a vector database, to enable efficient similarity search.
5. **Retriever Setup**:
    - Use the vector database to set up a retriever, which will fetch relevant chunks based on user queries.
6. **Language Model and Prompt Template**:
    - Initialize the language model (ChatOpenAI) and define a custom prompt template to guide the model in generating answers.
7. **RetrievalQA Pipeline**:
    - Combine the retriever and language model into a RetrievalQA chain, which uses the retriever to fetch relevant chunks and the language model to generate answers.
    -

8. **Implementation**:
    - Define the system's functionality, including loading PDFs, creating embeddings, setting up the retriever, and generating answers.
    - Implement a user-friendly interface for uploading files and asking questions.

Conclusion:

The developed RAG system efficiently handles the extraction, chunking, embedding, and retrieval of information from PDF documents. By leveraging the capabilities of LangChain and Chroma, the system can provide concise and accurate answers to user queries, making it a valuable tool for information retrieval in various applications. The methodology outlined ensures scalability and robustness, enabling the system to process large documents and complex queries effectively.