Final Report

Problem Statement:

Retrieval Augmented Generation (RAG) with LangChain

The goal is to create a system that extracts information from a PDF file and provides concise, accurate answers to user queries based on the content. The challenge involves handling large documents, ensuring efficient retrieval, and generating relevant responses.

Objective:

1. **PDF Handling**: Accept a PDF file as input and extract its textual content.
2. **Text Chunking**: Divide the extracted text into manageable chunks to facilitate efficient search and retrieval.
3. **Embedding**: Convert text chunks into embeddings for easy comparison and retrieval.
4. **Question Answering**: Use a language model to 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 up the Google API key for accessing Google services, if applicable.
    - Import necessary libraries, including PyMuPDF for PDF handling, langchain modules for text processing, and Chroma for vector storage.
2. **PDF Loading and Text Extraction**:
    - Open the PDF using fitz from the PyMuPDF library.
    - Iterate over each page to extract the text and concatenate it into a single string.
3. **Text Chunking**:
    - Utilize RecursiveCharacterTextSplitter to divide the extracted text into chunks of 1000 characters, with an overlap of 200 characters to maintain context between chunks.
4. **Embedding and ChromaDB**:
    - Create embeddings for each text chunk using OpenAIEmbeddings from the langchain_openai package.
    - Store the embeddings in Chroma, a vector database, to facilitate efficient similarity search.
5. **Retriever Setup**:
    - Configure a retriever using Chroma, which fetches relevant chunks based on user queries.
6. **Language Model and Prompt Template**:
    - Initialize the language model (e.g., ChatVertexAI with the "gemini-1.5-flash" model).
    - Define a custom prompt template to guide the model in generating answers based on context and chat history.

7. **RetrievalQA Pipeline**:
    - Combine the retriever and language model into a RetrievalQA chain.
    - The RetrievalQA chain uses the retriever to fetch relevant chunks and the language model to generate answers based on the retrieved context.
8. **Implementation**:
    - Define and implement the system's functionality, including PDF loading, text chunking, embedding creation, retriever setup, and answer generation.
    - Develop a user-friendly interface for uploading PDF files and posing 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 provides concise and accurate answers to user queries. This approach ensures scalability and robustness, enabling effective processing of large documents and complex queries. The methodology outlined in this report offers a comprehensive framework for building a robust information retrieval system.