NAME:- Prachi Varshney

Github project :- https://github.com/Varhneyprachi/ai-pdf-chatbot-1

Deployed link:- https://ai-pdf-chatbot-hzb9.onrender.com/

PDF Question-Answering System Using LangChain

1. Title & Objective

Title: PDF Question-Answering System Using LangChain
Objective:
To build an intelligent system that can answer questions from PDF documents using a combination of **large language models (LLMs)**, **document embeddings**, and **vector-based retrieval**. The system allows users to upload PDFs and query their content in natural language.

2. Tools & Frameworks Used

- **Python 3.x** – Core programming language.
- **Streamlit** – For building an interactive web interface.
- **LangChain** – Framework to integrate LLMs, vector stores, and document handling.
- **Ollama LLM & Embeddings** – Language model for generating answers and converting text into embeddings.
- **Chroma** – Vector database for storing and retrieving document embeddings.
- **PyPDFLoader** – Reads and extracts text from PDF files.
- **RecursiveCharacterTextSplitter** – Splits documents into manageable chunks.

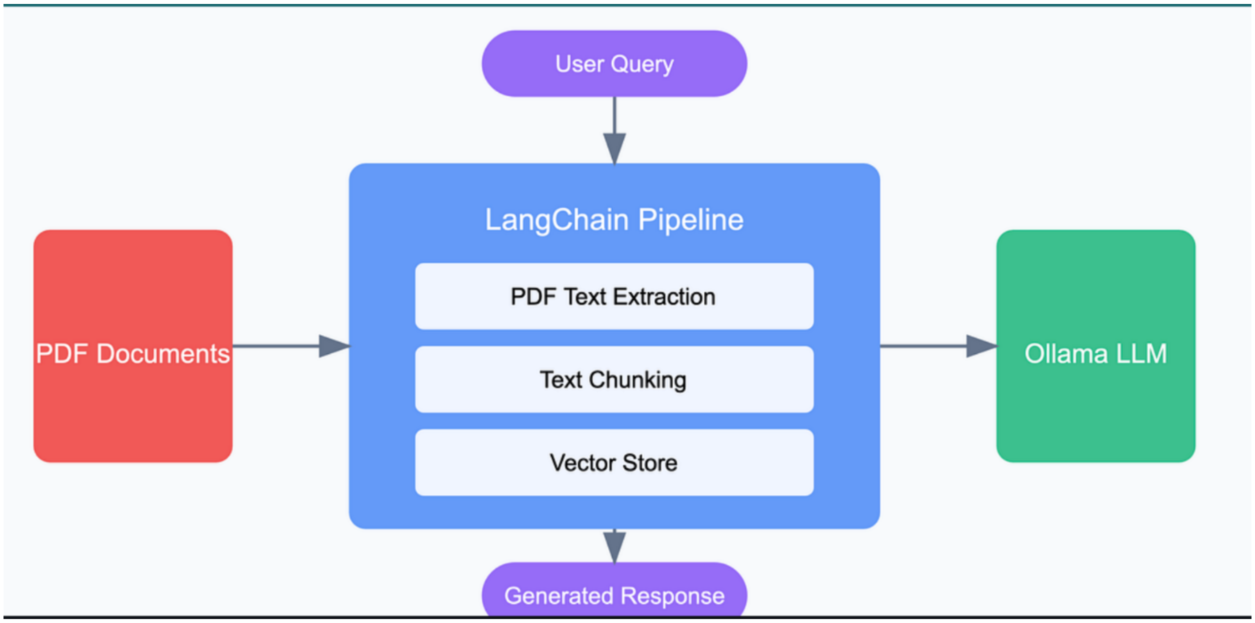
3. Workflow / Approach

Workflow Summary:

1. **PDF Upload** → User uploads a PDF file.
2. **Text Extraction** → PyPDFLoader extracts all text from the PDF.
3. **Text Chunking** → RecursiveCharacterTextSplitter splits text into smaller, overlapping chunks.

4. **Embedding Generation** → OllamaEmbeddings converts chunks into numerical vectors.
5. **Vector Store Creation** → Chroma stores embeddings for semantic search.
6. **Query Handling** → User enters a query; relevant chunks are retrieved using similarity search.
7. **Answer Generation** → LLM (Ollama) uses the retrieved chunks to generate a response.

Diagram:

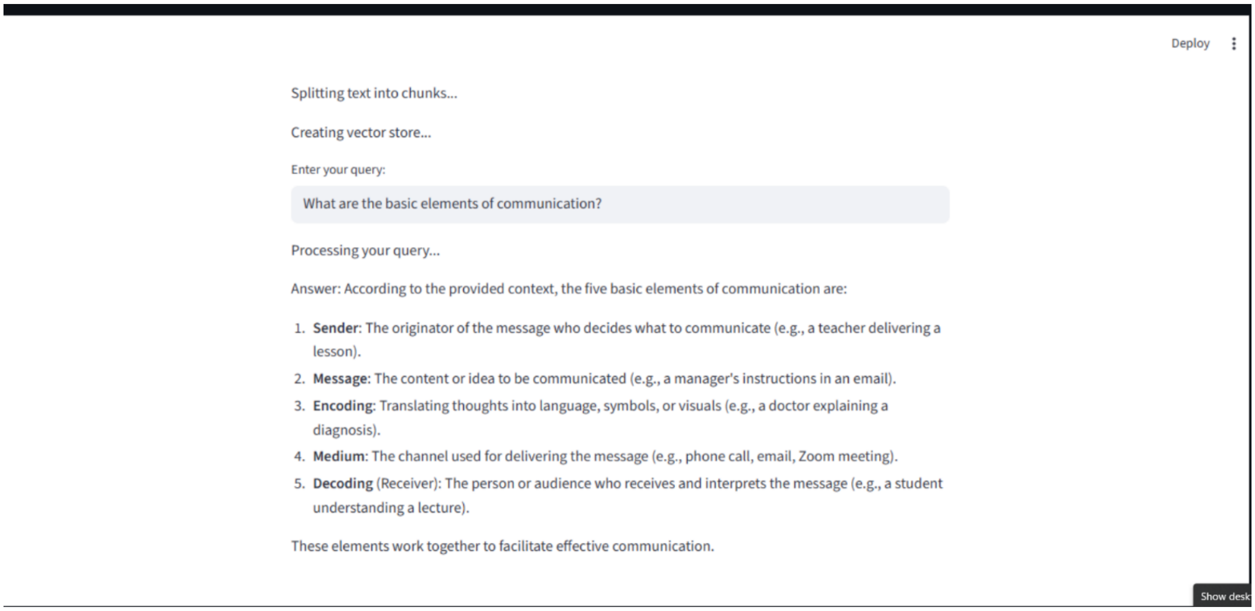


4. Key Implementation Steps

- **PDF Text Extraction:** Use PyPDFLoader to read PDF pages.
- **Text Chunking:** Split documents into overlapping chunks for better context retrieval.
- **Embeddings & Vector Store:** Convert chunks to embeddings with OllamaEmbeddings and store in Chroma for semantic search.
- **LLM Prompting:** Use ChatPromptTemplate to structure prompts combining context + question.
- **Answer Generation:** StuffDocumentsChain integrates retrieved chunks into LLM for final answer.
- **Web Interface:** Streamlit provides a user-friendly interface to upload PDFs and ask questions.

5. Results / Observations

- Successfully answered queries from multi-page PDFs.



Splitting text into chunks...

Creating vector store...

Enter your query:

What are the basic elements of communication?

Processing your query...

Answer: According to the provided context, the five basic elements of communication are:

1. **Sender**: The originator of the message who decides what to communicate (e.g., a teacher delivering a lesson).
2. **Message**: The content or idea to be communicated (e.g., a manager's instructions in an email).
3. **Encoding**: Translating thoughts into language, symbols, or visuals (e.g., a doctor explaining a diagnosis).
4. **Medium**: The channel used for delivering the message (e.g., phone call, email, Zoom meeting).
5. **Decoding** (Receiver): The person or audience who receives and interprets the message (e.g., a student understanding a lecture).

These elements work together to facilitate effective communication.

6. Future Improvements

- Add support for multiple PDF uploads.
- Integrate summarization and highlighting of answers in PDF.
- Use more advanced LLMs for better contextual reasoning.
- Deploy as a cloud web application for scalability.