PROBLEM STATEMENT:

Retrieval-Augmented Generation (RAG) Problem: Develop a lightweight, real-time Retrieval-Augmented Generation (RAG) system that allows users to upload multiple unrelated multi-page PDF documents and extract relevant highly information based on a query.

APPROACH

1. Introduction

- **Project Overview**: This application is a PDF-based chatbot powered by the Mistral-7B-Instruct model, designed to allow users to ask questions about the contents of uploaded PDF files.
- **Technologies Used**: Streamlit, LangChain, HuggingFace, FAISS, LlamaCpp, PyPDF2, Tesseract OCR, pdf2image.

---

2. WORKING:

- **PDF Upload**: The user can upload multiple PDF files using the Streamlit file uploader in the sidebar.
- **PDF Text Extraction**:
    - If the uploaded file is a PDF, the text is extracted using OCR (pytesseract) and PDF-to-image conversion (pdf2image).
    - The extracted text is processed and cleaned up (e.g., removing line breaks).
- **Text Chunking**: The extracted text is split into smaller chunks using `RecursiveCharacterTextSplitter` to optimize it for embedding and querying.
- **Embedding**: The chunks of text are then converted into vector embeddings using the HuggingFace embeddings model (`sentence-transformers/all-MiniLM-L6-v2`).
- **Vector Store**: The embeddings are stored in a FAISS index, enabling efficient similarity searches.
- **Conversational Chain**: The `ConversationalRetrievalChain` from LangChain is created using the FAISS vector store and the LlamaCpp-based language model (`mistral-7b-instruct-v0.1.Q4_K_M.gguf`).
- **Chat Interface**: The user can ask questions through the chatbot interface, and responses are generated by querying the vector store.

---

4. Key Components

- **Session State**:
  - `history`: Keeps track of the conversation history.
  - `generated`: Stores the chatbot's responses.
  - `past`: Stores user inputs.
- **Text Extraction (`extract_text_from_pdf`)**:
  - Converts PDF pages into images and extracts text from those images using OCR.
- **Chat Functionality**:
  - `conversation_chat`: Handles user input and generates responses using the LangChain ConversationalRetrievalChain.
- **Display Chat History**: Displays past user inputs and generated responses in a chat-like interface.

---

5. Flow of Execution

- **Uploading Files**: The user uploads one or more PDF files.
- **Text Extraction**: Text is extracted from the PDFs using OCR.
- **Text Processing**: The extracted text is split into chunks for efficient processing and vectorization.
- **Embedding**: The text chunks are converted into vector embeddings using HuggingFace's pre-trained model (sentence-transformers/all-MiniLM-L6-v2)
- **Creating Conversational Chain**: The embeddings are stored in a FAISS vector store, and a ConversationalRetrievalChain is set up using the LlamaCpp language model.
- **User Interaction**: The user interacts with the chatbot by asking questions, and the chatbot retrieves relevant information from the vector store to provide responses.

6.INPUT:

STRUCTURED:-

- **TEXTUAL DATA:** Data consists of story books,text books,normal pdf which contains text.

UNSTRUCTURED DATA:-

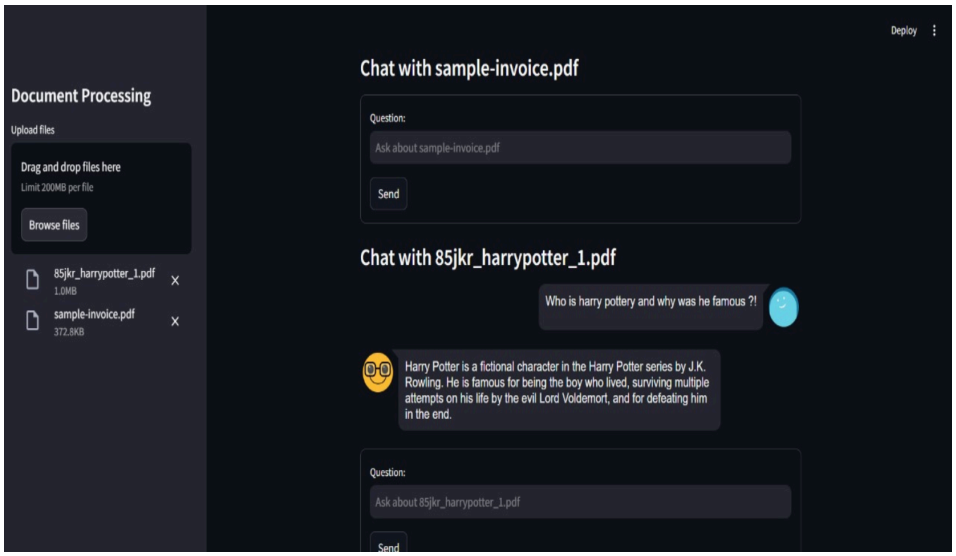
- INVOICES DATA: Data consists of textual data + tabular data.

- EMAIL DATA: As we didn’t find any input version for email data, we took pdf version of it.

7.OUTPUT:

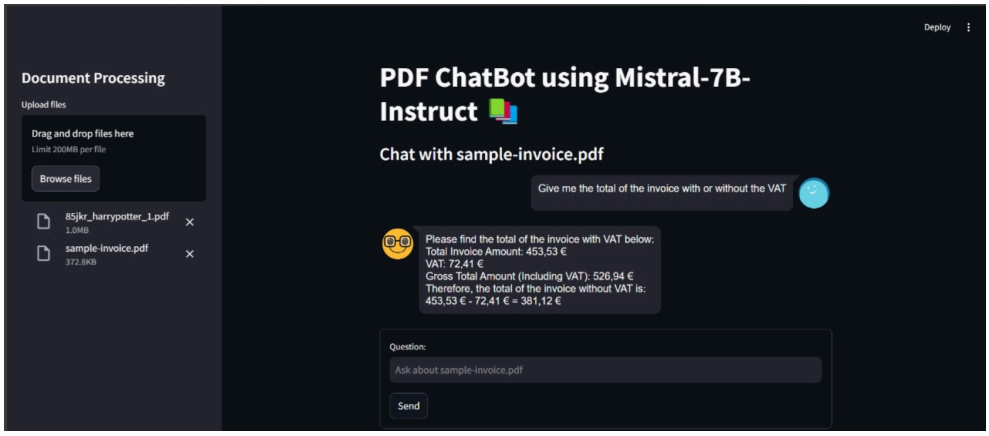
Structured data:

- Textual data: time taking 1 min 20 seconds uploading time

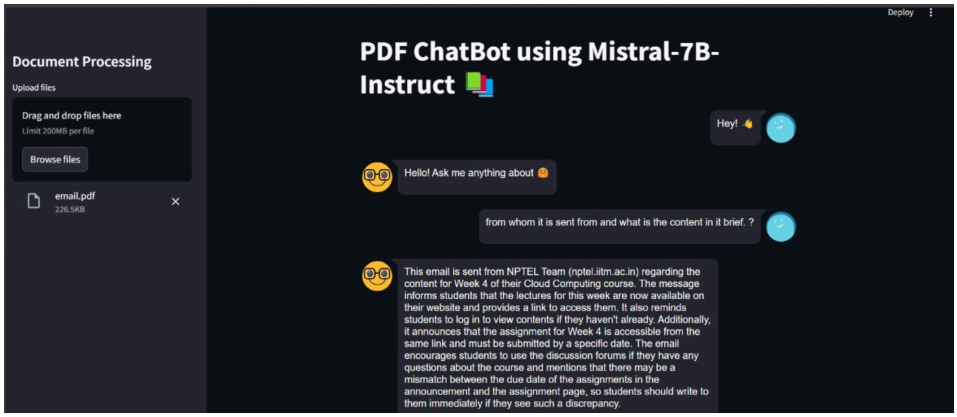


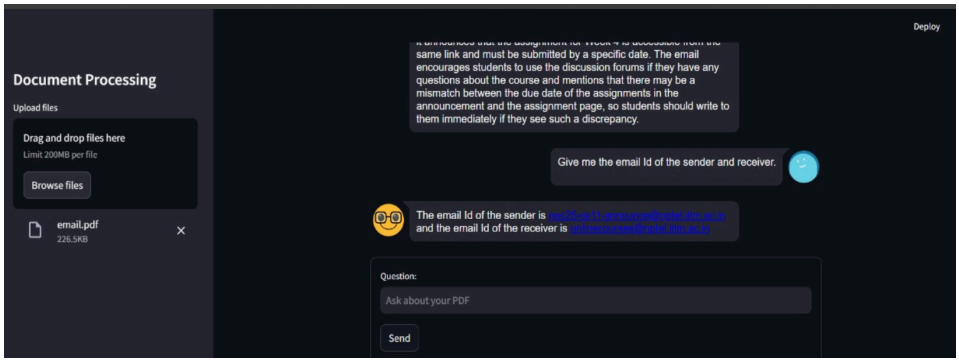
UNSTRUCTRED DATA:

- **INVOICE:**



- EMAIL DATA:





8. Features

- **Multi-file Upload**: Supports uploading multiple PDF files at once.
- **Real-time Chat**: Provides a real-time chat interface where users can ask questions about the uploaded PDFs.
- **Customizable Model**: The language model and embeddings used in the conversational chain can be customized.

---

9. Future Improvements

- **Support for Other File Types**: Extend support for other file types like Word documents, text files, etc.
- **Advanced Querying**: Implement more sophisticated querying options, such as filtering responses based on specific sections of the document.
- **Model Tuning**: Allow for fine-tuning the language model for more domain-specific tasks.
