MSPM's

Deogiri Institute of Engineering and Management Studies, Aurangabad

Department of Computer Science and Engineering (Artificial Intelligence and Machine Learning)

Continuous Assessment – II

B. Tech.- CSE (AI/ML)

(BTAIC701) Natural Language Processing (Theory)

2023-24 (Semester – I)

LLM PDF Chat Web App Architecture

Team Members :

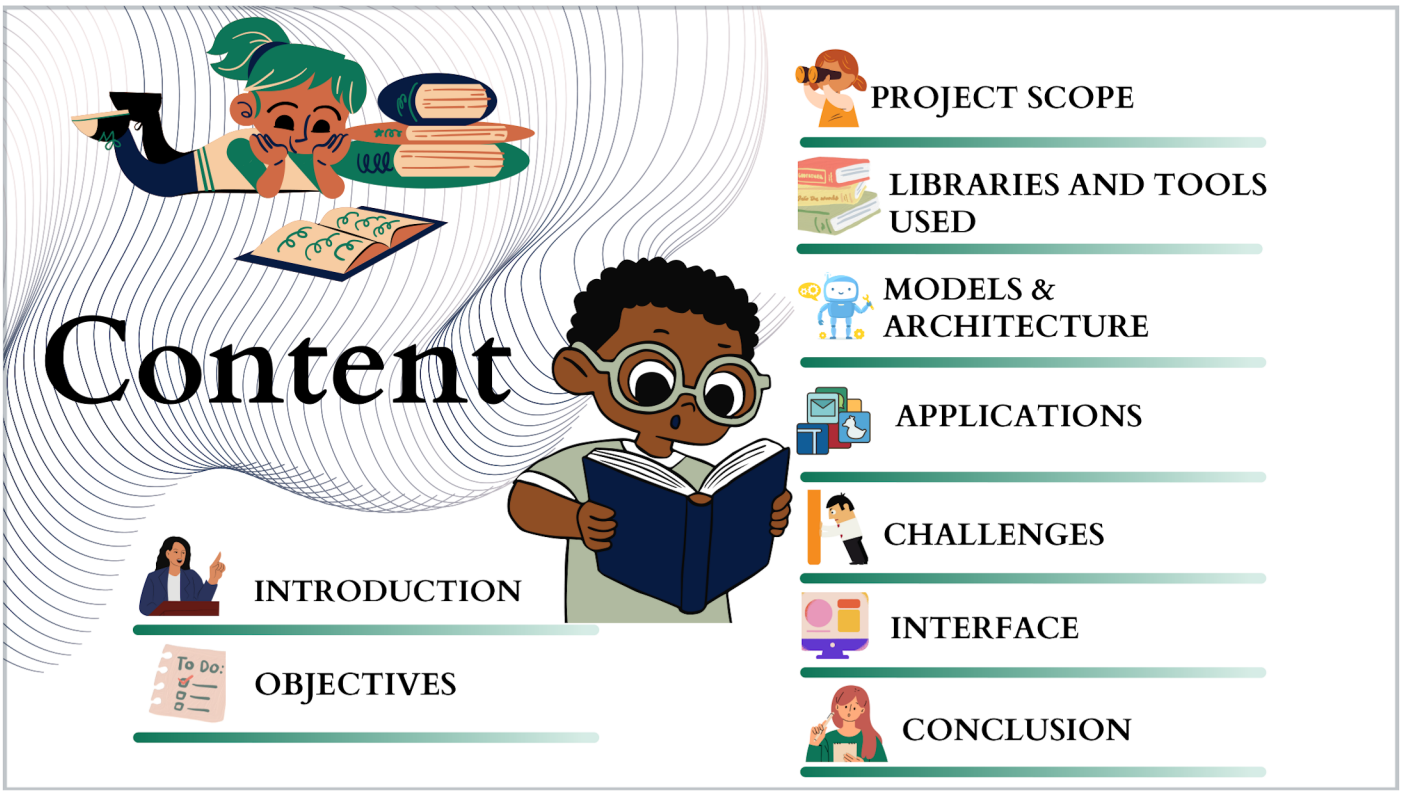
Mihir Pande (AI4131)

Om Patange (AI4137)

Preshit Desai (AI4145)

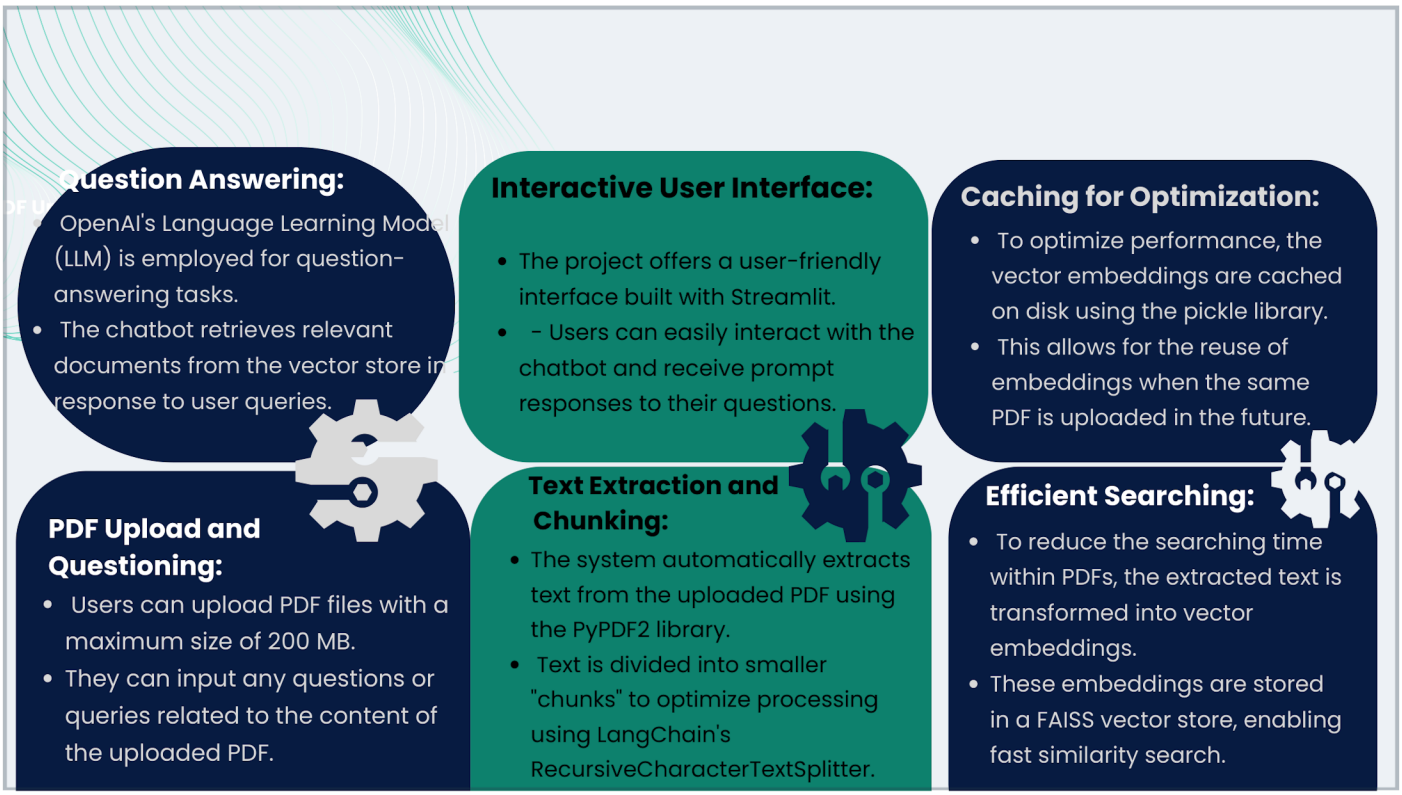
Under the Guidance of

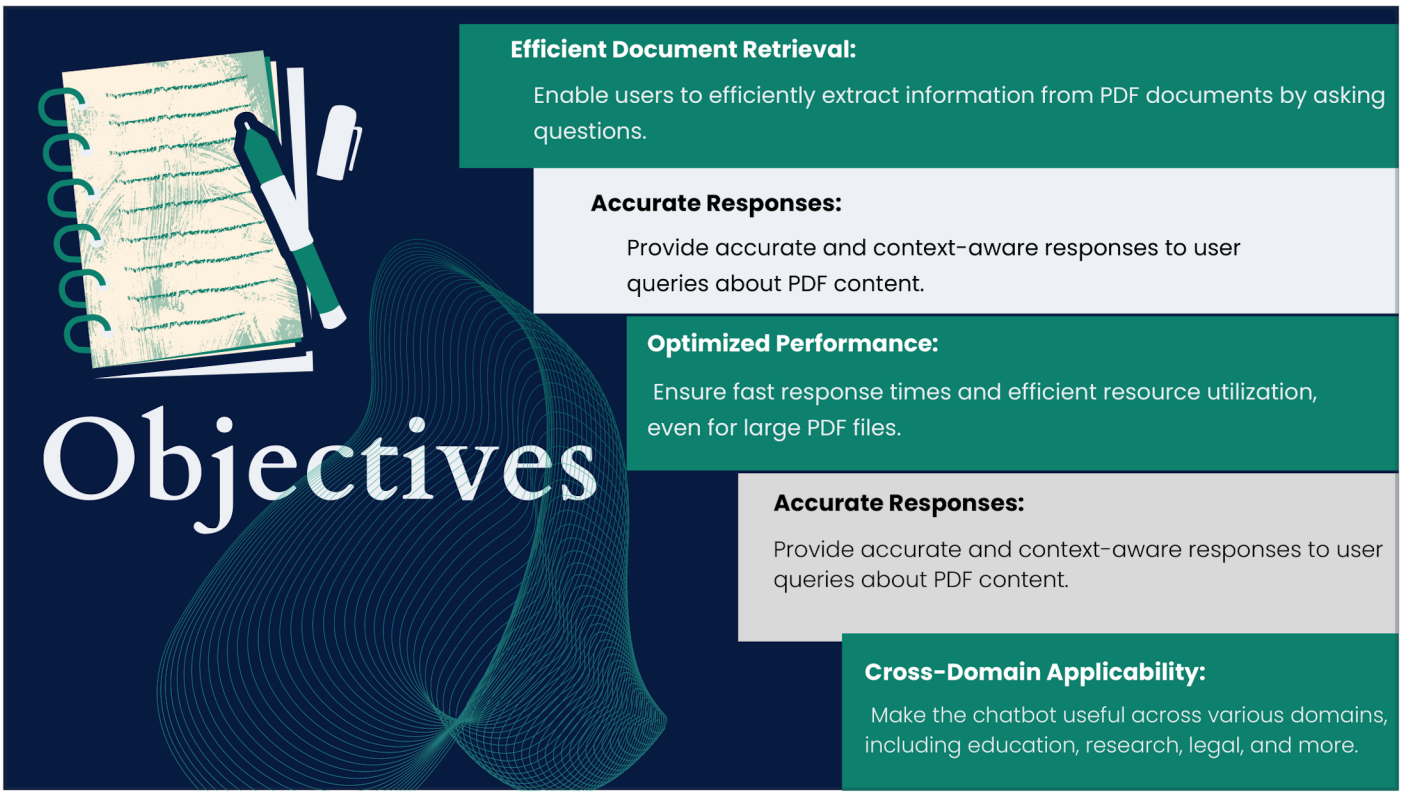
Ms. Sugandha Nandedkar


Content
INTRODUCTION
OBJECTIVES
PROJECT SCOPE
LIBRARIES AND TOOLS USED
MODELS & ARCHITECTURE
APPLICATIONS
CHALLENGES
INTERFACE
CONCLUSION

Introduction

This project is designed to facilitate efficient and accurate question-answering from PDF documents. It provides a user-friendly web application for users to upload PDF files and ask questions related to the content within those PDFs. The chatbot, powered by a Language Learning Model (LLM), then responds with relevant answers.


Question Answering:
OpenAI's Language Learning Model (LLM) is employed for question-answering tasks.
The chatbot retrieves relevant documents from the vector store in response to user queries.
Interactive User Interface:
The project offers a user-friendly interface built with Streamlit.
- Users can easily interact with the chatbot and receive prompt responses to their questions.
Caching for Optimization:
To optimize performance, the vector embeddings are cached on disk using the pickle library.
This allows for the reuse of embeddings when the same PDF is uploaded in the future.
PDF Upload and Questioning:
Users can upload PDF files with a maximum size of 200 MB.
They can input any questions or queries related to the content of the uploaded PDF.
Text Extraction and Chunking:
The system automatically extracts text from the uploaded PDF using the PyPDF2 library.
Text is divided into smaller "chunks" to optimize processing using LangChain's RecursiveCharacterTextSplitter.
Efficient Searching:
To reduce the searching time within PDFs, the extracted text is transformed into vector embeddings.
These embeddings are stored in a FAISS vector store, enabling fast similarity search.


Objectives
Efficient Document Retrieval:
Enable users to efficiently extract information from PDF documents by asking questions.
Accurate Responses:
Provide accurate and context-aware responses to user queries about PDF content.
Optimized Performance:
Ensure fast response times and efficient resource utilization, even for large PDF files.
Accurate Responses:
Provide accurate and context-aware responses to user queries about PDF content.
Cross-Domain Applicability:
Make the chatbot useful across various domains, including education, research, legal, and more.

Project Scope

The project scope encompasses the development of a PDF question-answering chatbot that allows users to upload PDFs, ask questions, and receive accurate responses. It aims to efficiently process large PDF files, ensure data privacy, and cater to diverse industries and applicatio

Libraries And Tools

Streamlit:
Streamlit is a Python library for effortless web app development, enabling the creation of interactive, data-driven applications using simple Python scripts, ideal for data visualization and analysis.

PyPDF2:
PyPDF2 is a Python library for PDF file operations, including text extraction, merging, splitting, and other manipulations.

dotenv:
dotenv is a Python library for loading environment variables from a .env file into your application. It's often used to securely manage sensitive data, like API keys, in a separate file for enhanced security and flexibility.

pickle:
Pickle is a Python library for serializing and deserializing Python objects, enabling the storage and retrieval of objects, including machine learning models.

Libraries

langchain.llms and langchain.chains.question_answering:

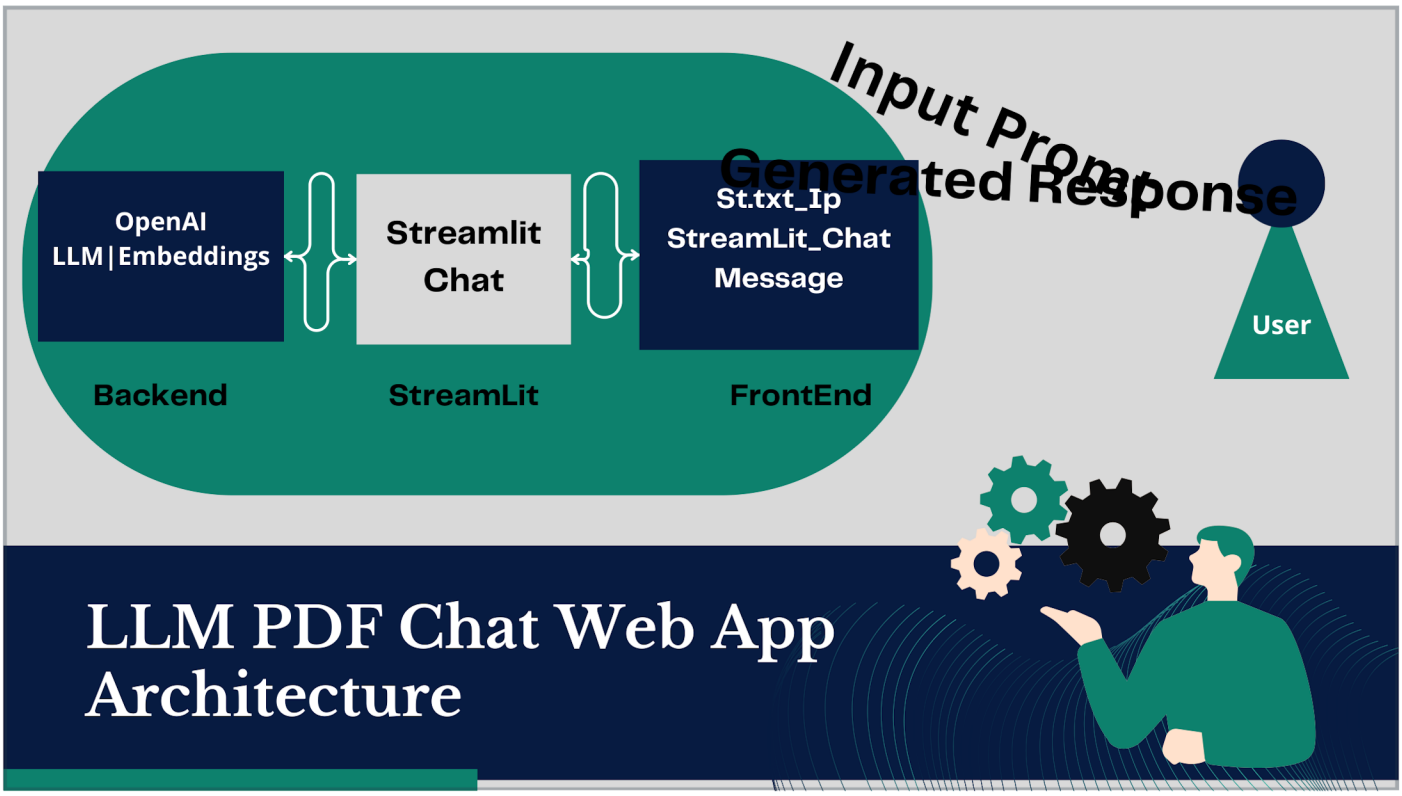
- Components of a larger NLP project, possibly for language model and question-answering tasks.

langchain.vectorstores:

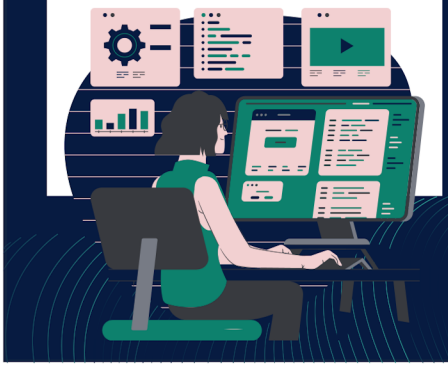
- Manages vector representations of text data, potentially using FAISS for efficient similarity search.

langchain.text_splitter and langchain.embeddings.openai:

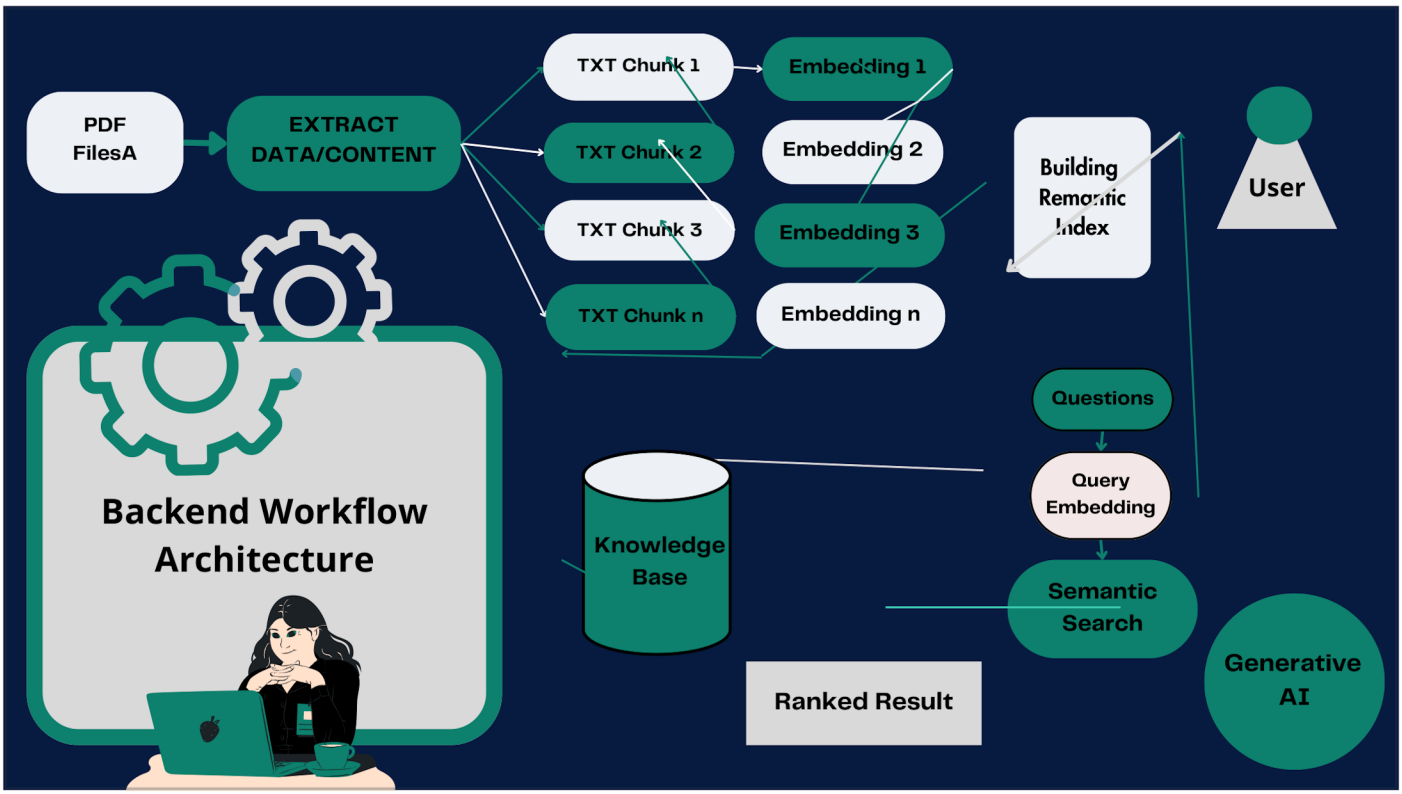
- Specialized libraries for text processing and embedding tasks using OpenAI models.


Input Prompt
Generated Response
OpenAI
LLM | Embeddings
Streamlit
Chat
St.txt_Ip
StreamLit_Chat
Message
User
Backend
StreamLit
FrontEnd
LLM PDF Chat Web App
Architecture

Conclusion From Architecture Diagram



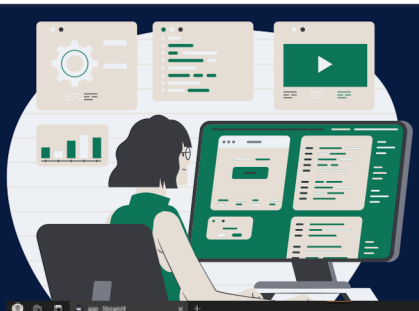
- **User sends a message**: The user interacts with the streamlit chat application's frontend and types a message..
- **Message sent to streamlit chat server**: The message is sent from the frontend to the streamlit chat server.
- **Message processed by backend**: The backend processes the message, which involve NLP tasks like Embedding, topic modeling, or intent recognition.
- **OpenAI LLM embeddings generated:** The backend generates embeddings for the message using OpenAI's large language model (LLM). Embeddings are numerical representations of words or sentences that capture their semantic meaning.
- **Generated response**: Based on the message embeddings and other contextual information, the backend generates a response using the OpenAI LLM.
- **Response sent to streamlit chat server**: The generated response is sent back to the streamlit chat server.
- **Response sent to user**: The streamlit chat server sends the response to the user's frontend.

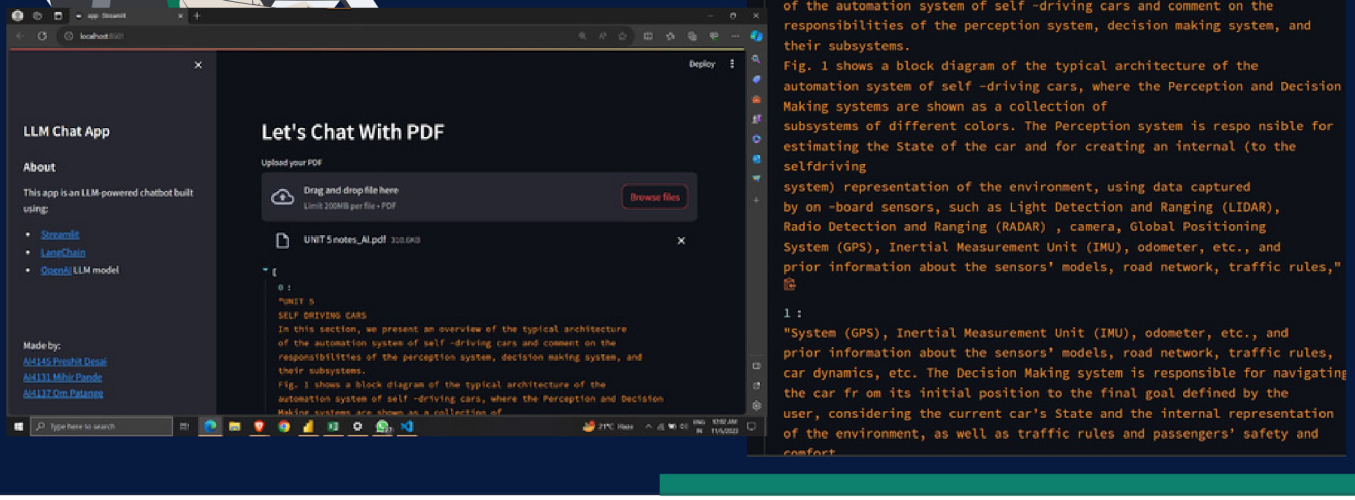

PDF FilesA
EXTRACT DATA/CONTENT
TXT Chunk 1
TXT Chunk 2
TXT Chunk 3
TXT Chunk n
Embedding 1
Embedding 2
Embedding 3
Embedding n
Building Remantic Index
User
Backend Workflow Architecture
Knowledge Base
Questions
Query Embedding
Semantic Search
Ranked Result
Generative AI

- PDF Files Uploaded: Users start by uploading one or more PDF files containing the data to be processed for responses.
- Extract Data: Uploaded PDFs are processed to extract relevant content using techniques like text extraction and segmentation.
- Split Text: Extracted text is divided into smaller chunks for efficient processing, treating each chunk as an independent unit.
- Apply Embedding: Text chunks are represented as numerical vectors using embedding models, capturing semantic meaning.
- Build Semantic Index: Collected embedding vectors form a semantic index for efficient text retrieval and comparison.
- Knowledge Base: A repository of information is used to answer user queries, with the semantic index facilitating rapid search.
- Ranked Results: Results are ranked based on relevance to the query, with the most relevant ones presented first.

- LLM Generative AI: Ranked results are processed by a Large Language Model (LLM) to generate informative responses.
- Response to User: Generated responses are provided to the user, who can continue the conversation by asking more questions.
- User Asks Question: Users engage with the system by asking various questions, and the system aims to understand their intent.
- Query Embedding: User queries are transformed into embedding vectors, capturing query semantics.
- Semantic Search: Query embeddings are used to search the semantic index for the most relevant knowledge base entries.
- Check Knowledge Base: Retrieved knowledge base entries are validated for accuracy and relevance.
- Generate Answer: Using the knowledge base entries, the system produces tailored and informative responses to user questions.

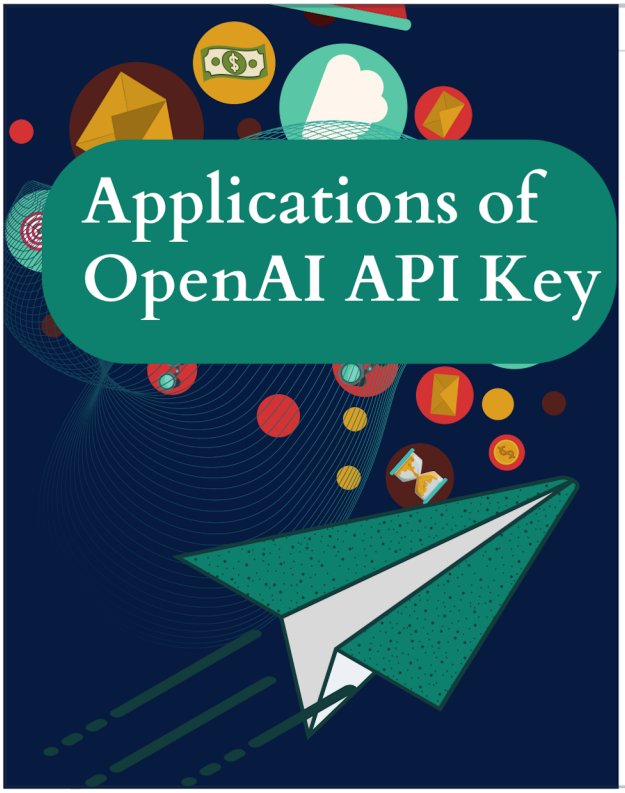
Extracting Text From PDF.





Open AI API Key Applications in Project

- OpenAI API Key: Enables access to OpenAI's Language Learning Models (LLMs) for text processing and question-answering, ensuring accurate responses to user queries.
- Model Training: Allows training and fine-tuning LLMs for specific tasks, enhancing contextual relevance to project requirements.
- LLM Integration: Integrates LLM into the chatbot for understanding and responding to user queries about PDF content.
- NLP Capabilities: Leverages advanced natural language processing capabilities for improved query interpretation.
- Continuous Enhancement: Facilitates staying updated with OpenAI's model advancements in natural language understanding and question-answering.



- Language Translation
- Question Answering
- Text Summarization
- Content Generation
- Sentiment Analysis
- Data Extraction and Analysis
- Speech Recognition
- Accessibility
- Financial Analysis
- Chatbots and Virtual Assistants

Challenges

User Experience: Ensuring a user-friendly interface for users with varying technical expertise.

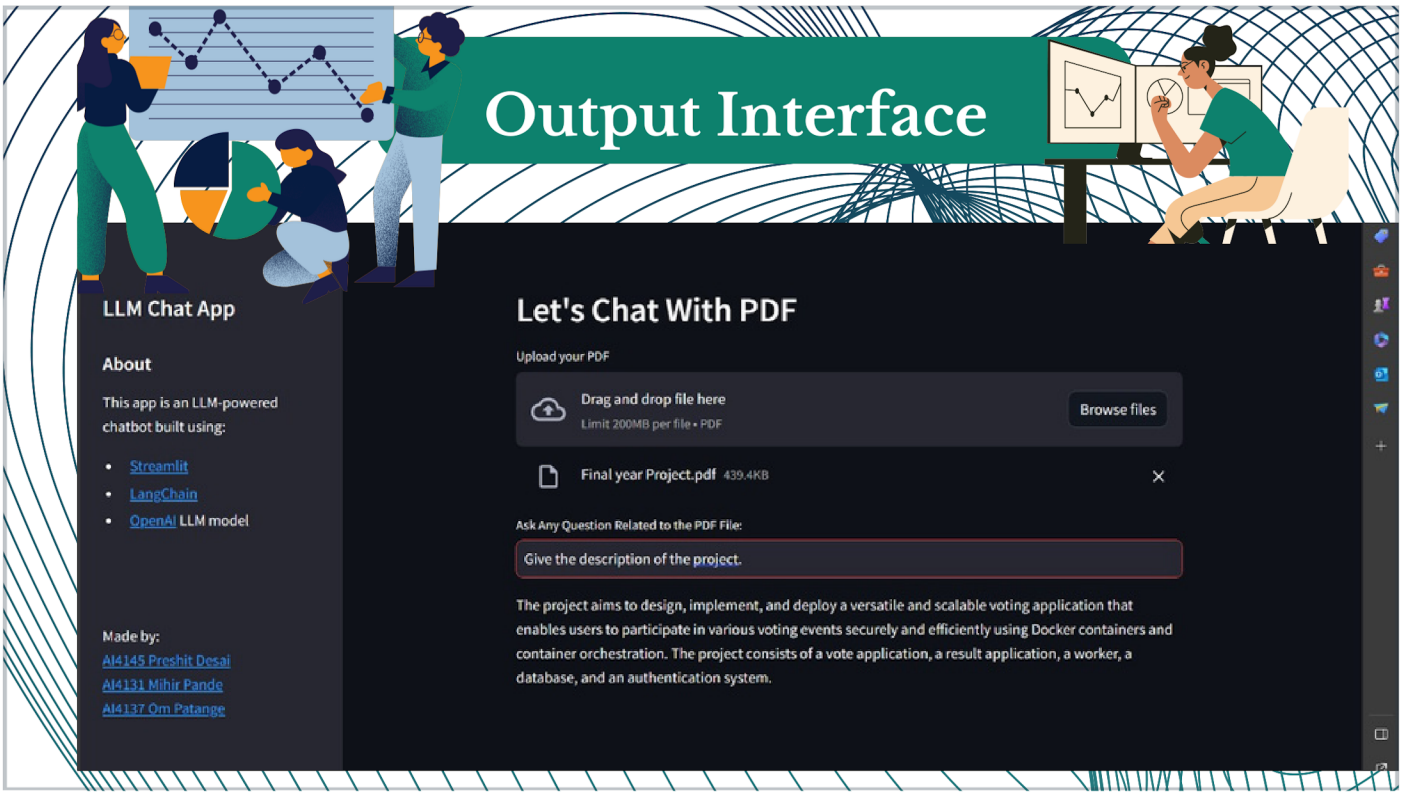
Language and Vocabulary: Handling diverse languages and specialized terminology.

Text Extraction Accuracy: Ensuring accurate text extraction, especially from scanned documents.

Question Ambiguity: Addressing ambiguous and context-dependent user questions.

PDF Variability: Handling various PDF formats, including scanned images and complex layouts.

Large PDF Files: Optimizing processing for large PDF files to maintain performance.


Output Interface
LLM Chat App
About
This app is an LLM-powered chatbot built using:
Streamlit
LangChain
OpenAI LLM model
Made by:
AI4145 Preshit Desai
AI4131 Mihir Pande
AI4137 Om Patange
Let's Chat With PDF
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
Final year Project.pdf 439.4KB
Ask Any Question Related to the PDF File:
Give the description of the project.
The project aims to design, implement, and deploy a versatile and scalable voting application that enables users to participate in various voting events securely and efficiently using Docker containers and container orchestration. The project consists of a vote application, a result application, a worker, a database, and an authentication system.

In conclusion, this project successfully creates a PDF question-answering chatbot, offering users an efficient means to extract information from PDFs. By addressing challenges like PDF variability and data privacy, the project enhances accessibility and security. With potential applications across diverse industries and a user-friendly interface, it streamlines information retrieval while continuously improving performance.

Conclusion


Thank you
so much!