Question Answering System using RAG

Introduction:

The Question Answering System (QAS) using Retrieval-Augmented Generation (RAG) is a natural language processing (NLP) system designed to provide accurate and relevant answers to user queries/questions. It features a user-friendly interface where users can upload their documents and ask questions related to the content. Leveraging the power of RAG, the system retrieves relevant information from the uploaded document and generates concise and informative responses.

A Question Answering System (QAS) is a type of computer program or artificial intelligence (AI) system designed to understand and respond to questions posed in natural language. The primary goal of a QAS is to provide accurate and relevant answers to user queries, similar to how a human would respond to questions.

Project Outline:

The Objective of the project is to create a robust and user-friendly QA system based on RAG allowing users to ask questions and receive accurate answers based on PDF documents. Retrieval-Augmented Generation (RAG), or RAG, is an architectural approach that can improve the efficacy of large language model (LLM) applications by leveraging custom data.

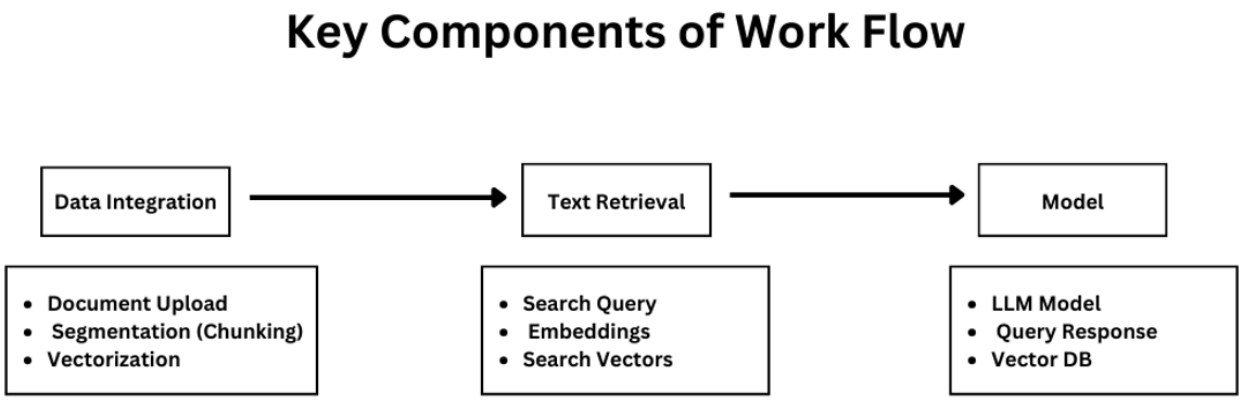
The project's scope involves creating a user-friendly interface enabling document upload and query submission, alongside implementing document processing to handle various formats and extract text content. Integration of the Retrieval-Augmented Generation (RAG) model facilitates retrieving and generating answers based on user queries. Additionally, incorporating user feedback mechanisms aims to enhance system performance, while optimization efforts ensure efficiency with large documents and complex queries. Testing and evaluation procedures assess system accuracy, performance, and usability, followed by comprehensive documentation creation and system deployment for user access.

System Architecture:

The architecture of the Question Answering System using RAG comprises several interconnected components, each playing a vital role in the system's functionality. At the core lies the user interface, providing a seamless interaction platform for users to upload documents and submit queries. Upon document submission, the system initiates document processing mechanisms to extract text content from various formats.

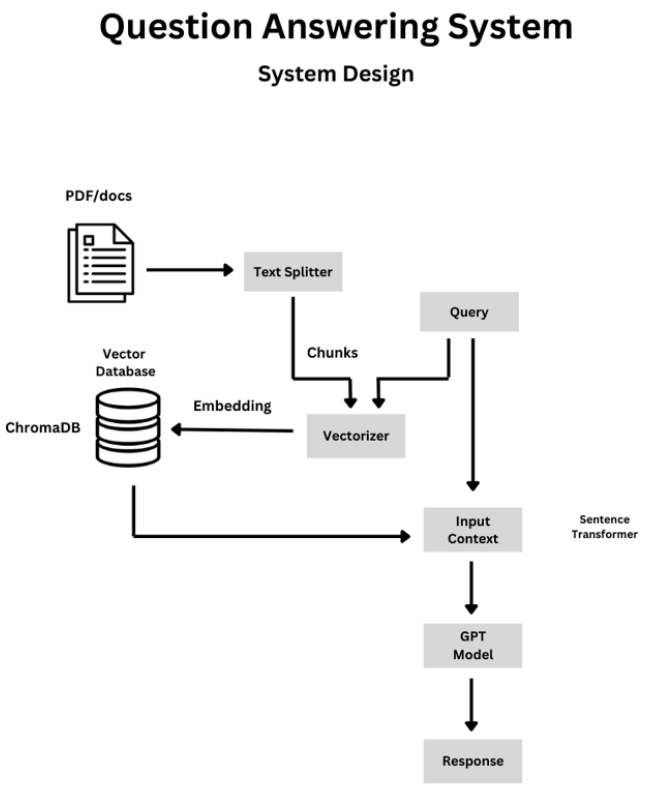
The workflow of the system is divided into three key components or steps, that is Data Integration, Text Retrieval, and Model Synthesis. Initially, in the Data Integration phase, documents are processed to extract text content, which is then indexed for efficient retrieval. Subsequently, in Text Retrieval, user queries are analyzed to identify relevant information, and the system leverages the Retrieval-Augmented Generation (RAG) model to retrieve pertinent passages from the integrated data. Finally, in Model Synthesis, the retrieved passages serve as

input for the RAG model, which synthesizes coherent and contextually relevant answers to the user's queries. This systematic approach ensures the accurate and efficient generation of answers, enhancing the overall user experience.



The Question Answering System (QAS) utilizes a combination of tools to facilitate its functionality. Gradio is employed for developing the user interface, offering a user-friendly platform for document upload and query submission. OpenAI serves as the embedding model, enabling the system to process and understand text inputs effectively. The GPT-3.5 language model, specifically the Langchain variant, is utilized for language modeling tasks, enhancing the system's natural language processing capabilities. Additionally, ChromaDB is employed as the vector database, providing efficient storage and retrieval of vectors for various components of the system. Together, these tools synergize to create a comprehensive and efficient QAS solution

The overall System Design of the Question Answering System,



Model Workflow:

The workflow of the model is divided into various segments or parts, That are

User Interaction: The user interacts with the system through the user interface, submitting questions or queries on the input field.

Document Processing: Upon receiving the user's question, the system accesses the document that was provided by user containing the information needed to answer the query.

Text Chunking: After the document is processed that is segmented into smaller text chunks or passages to facilitate efficient processing.

Embedding Model: Each text chunk is passed through the embedding model, which converts the textual data into numerical representations (vectors) in a high-dimensional space. These embeddings capture the semantic meaning and context of the text chunks.

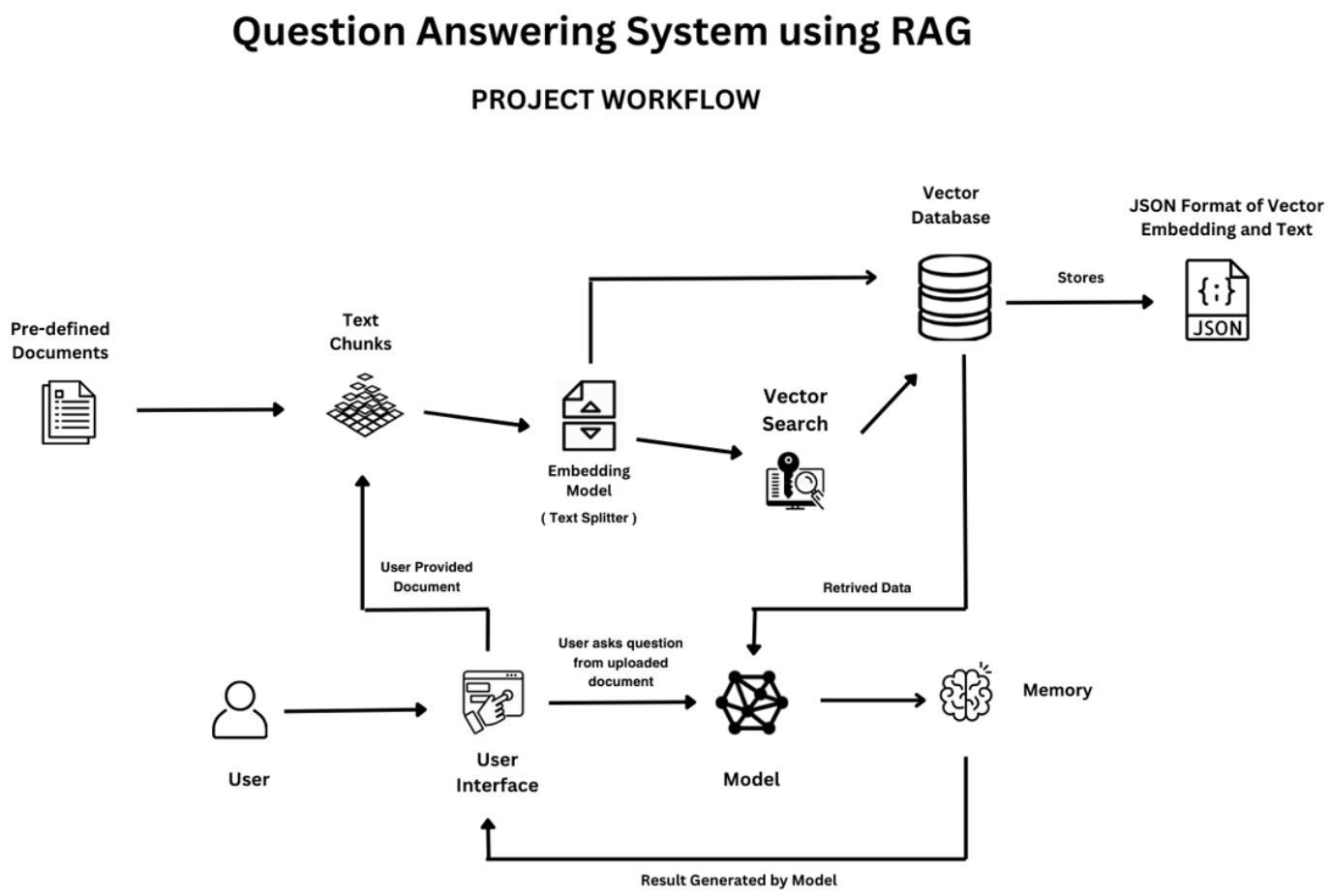
Vector Search: The vector representations of the text chunks are used to perform a vector search, where similar passages are retrieved based on their proximity/score in the embedding space. This step is very helps to identify required answer related to the user's question.

Vector Database: The system queries a vector database, such as ChromaDB, to efficiently retrieve the vector representations of the text chunks. This database stores pre-computed vectors for text passages, allowing for quick retrieval during the search process the data are stored in the JSON format.

Model Processing: The retrieved text chunks, along with their corresponding vectors, are passed to a large language model. The model utilizes the text chunks and their embeddings to generate a response to the user's question. This process may involve advanced natural language processing (NLP) techniques and models like GPT-3.5 language model.

Memory: The system may incorporate a memory component to store relevant information or context from previous interactions. This memory helps maintain continuity and coherence in the conversation, enabling the system to provide more accurate and personalized responses over time. The memory unit will pass the user's question response to the User Interface (UI)

The Schematic representation of the model workflow is ,



This workflow illustrates how the QAS system processes user queries, retrieves relevant information from documents, and generates responses using advanced language models and vector-based search techniques, ultimately enhancing the user experience and accuracy of the system.

Dependencies and Configuration:

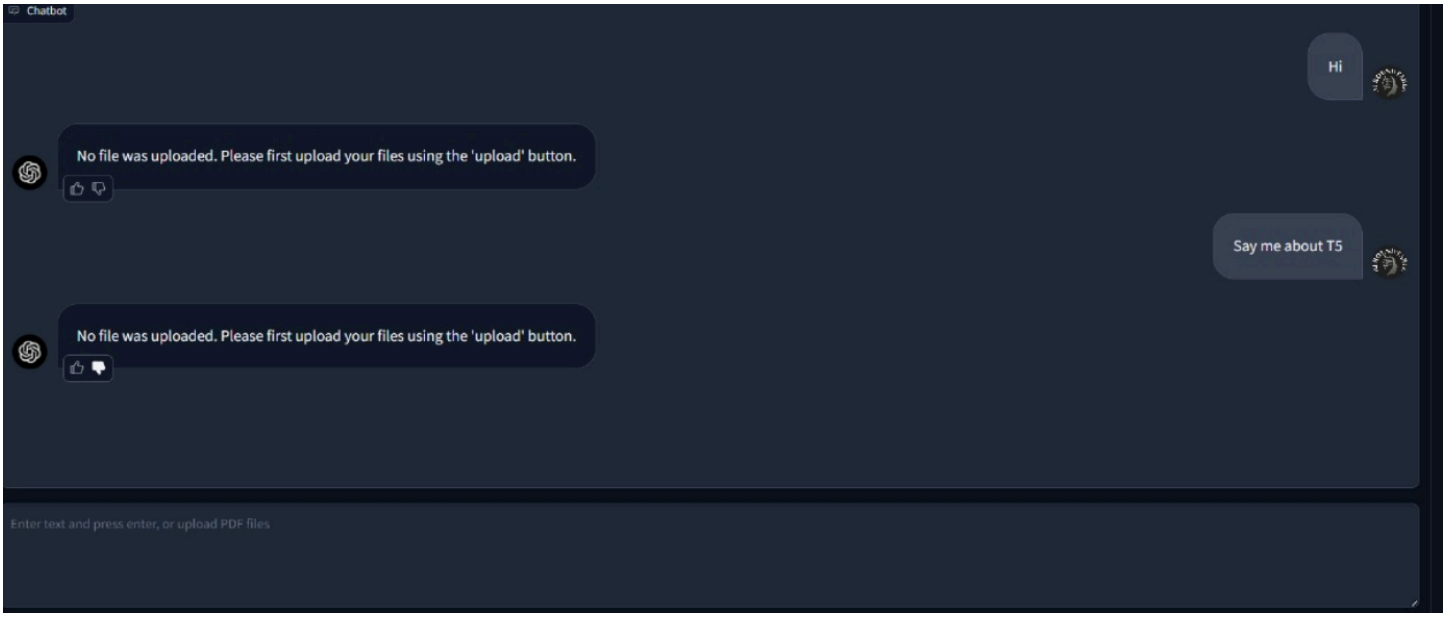
System requirement: Operating System: Windows, macOS, Linux. RAM: Above 13GB (to accommodate the memory-intensive operations of the language models). Disk Space: Sufficient

space for storing documents and model files. Processor: Multi-core processor recommended for faster processing.

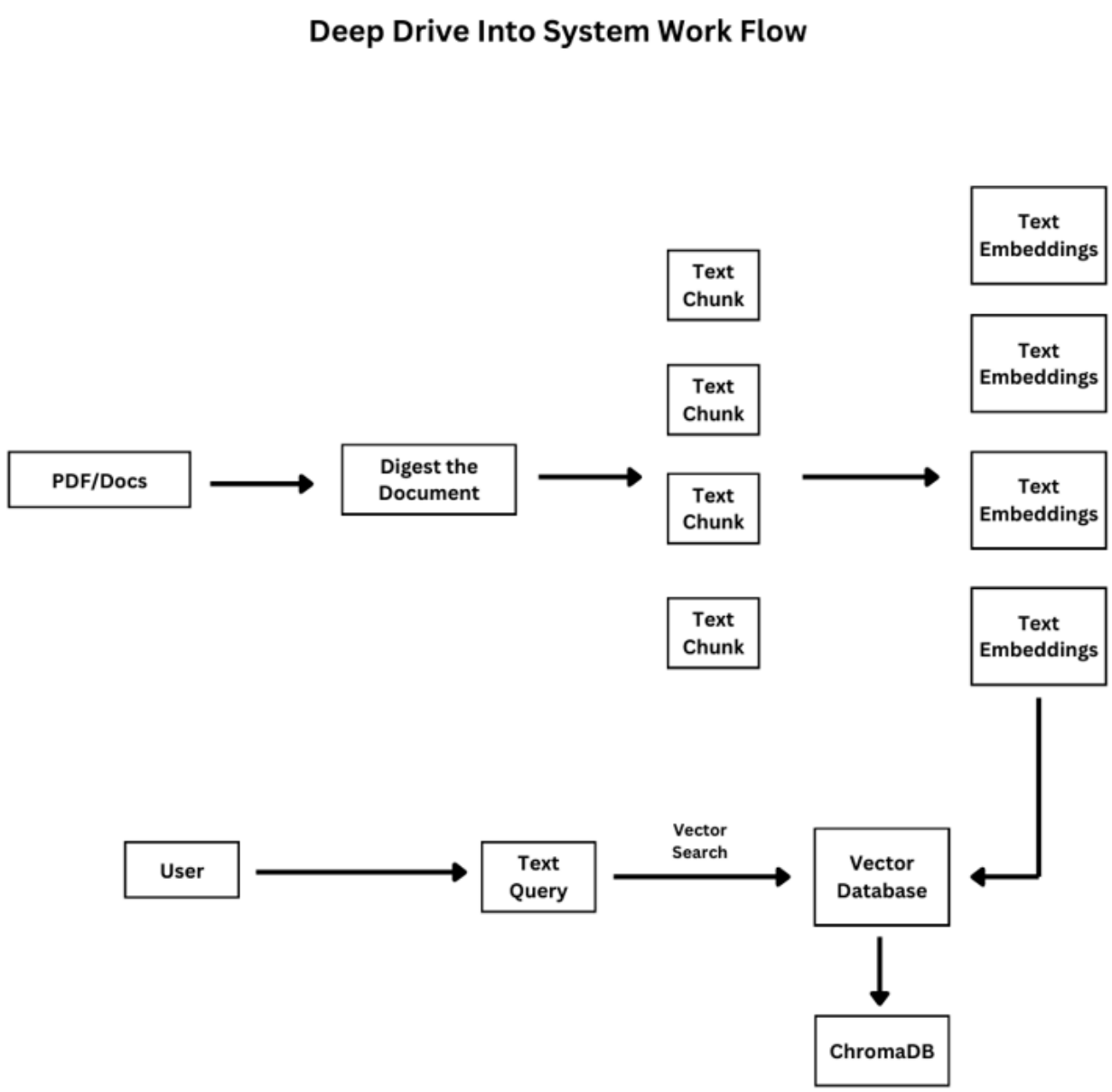
Dependencies: Pandas, Open AI, PyPDF, ChromaDB, Langchain. Ensure all dependencies are installed which are mentioned.

Utilization

Inputting document and questions: In User Interface there is two input fields one is for document upload for querying and the other input field is to provided the questions that needed to be asked to the document. Ensure that your question is clear and specific to obtain accurate answers.



Retrieving Information: After the documented and user query asked to the model it works in retrieval process involves searching for embedded text or segments of text that contain potential answers to user question.



This is how the response retrieved from the model.

Generating Answers: Using the retrieved information, the QAS generates coherent and contextually relevant answers to user's query. The answers are presented to user via the interface in a readable format, providing accurate and informative responses to your questions.

Challenges encountered:

Challenges encountered in developing a Question Answering System (QAS) include:

API Integration: Integrating external APIs, such as Open AI language models or document processing tools, can be complex and may require understanding of API documentation, authentication methods, and data formats and Open AI API is paid service and can't able to get the Open AI API Key to work with it.

PDF Conversion and Text Chunking: Converting PDF documents into text and segmenting them into smaller chunks can pose challenges, especially with complex formatting or non-standard layouts. Handling large PDF files efficiently and accurately extracting relevant text content without loss of information is essential for effective document processing. This model can only able to work with PDF with less than 50 pages.

Text Embedding: Embedding textual data into numerical representations (vectors) using embedding models requires careful preprocessing and understanding of language semantics. Ensuring that embeddings capture the semantic meaning and context of the text accurately is crucial for accurate retrieval and generation of answers.

Accuracy of Answers: Achieving high accuracy in generating answers to user queries is a significant challenge, especially in complex or ambiguous scenarios. Ensuring that the system retrieves and processes relevant information accurately from documents and generates coherent and contextually relevant answers is essential for user satisfaction and trust.

Future Enhancements:

Multi-Document Reading Capability: Enhance the model capability to read and process multiple documents simultaneously. This would allow users to input queries across a broader range of sources and enable the system to provide more comprehensive and nuanced answers. It would be the complex one but can be achievable.

Customizable Fine-Tuning: Provide users with the ability to fine-tune the system's performance and behavior according to their specific needs and preferences. This could include customizable parameters for retrieval, generation, and answer ranking and score.