Scope Of the project - DocQBot

HEXAWARE

Project Objectives:

Automate Document Retrieval:
Develop a system that automates the process of retrieving information from large documents, eliminating the need for manual searching.

Enable Accurate Question Answering:
Use AI (GPT-4) to provide accurate and context-specific answers based on user-submitted questions and relevant document content.

Integrate Advanced Information Retrieval Techniques:
Implement Retrieval-Augmented Generation (RAG) to retrieve the most relevant sections of the document, ensuring that the answer is based on the best available information.

Improve User Experience:
Design an intuitive web interface that allows users to easily upload documents, submit questions, and receive answers quickly.

Scope Of the project – DocQBot - Contd

HEXAWARE

Deliverables:

Functional Web Application:

A user-friendly web interface where users can:
Upload documents (e.g., PDFs).
Submit questions related to the content of the uploaded documents.
View generated answers in real-time.

Backend API:

Integration with the Azure OpenAI API for GPT-4-based answer generation.

Embedding Generation and Storage:

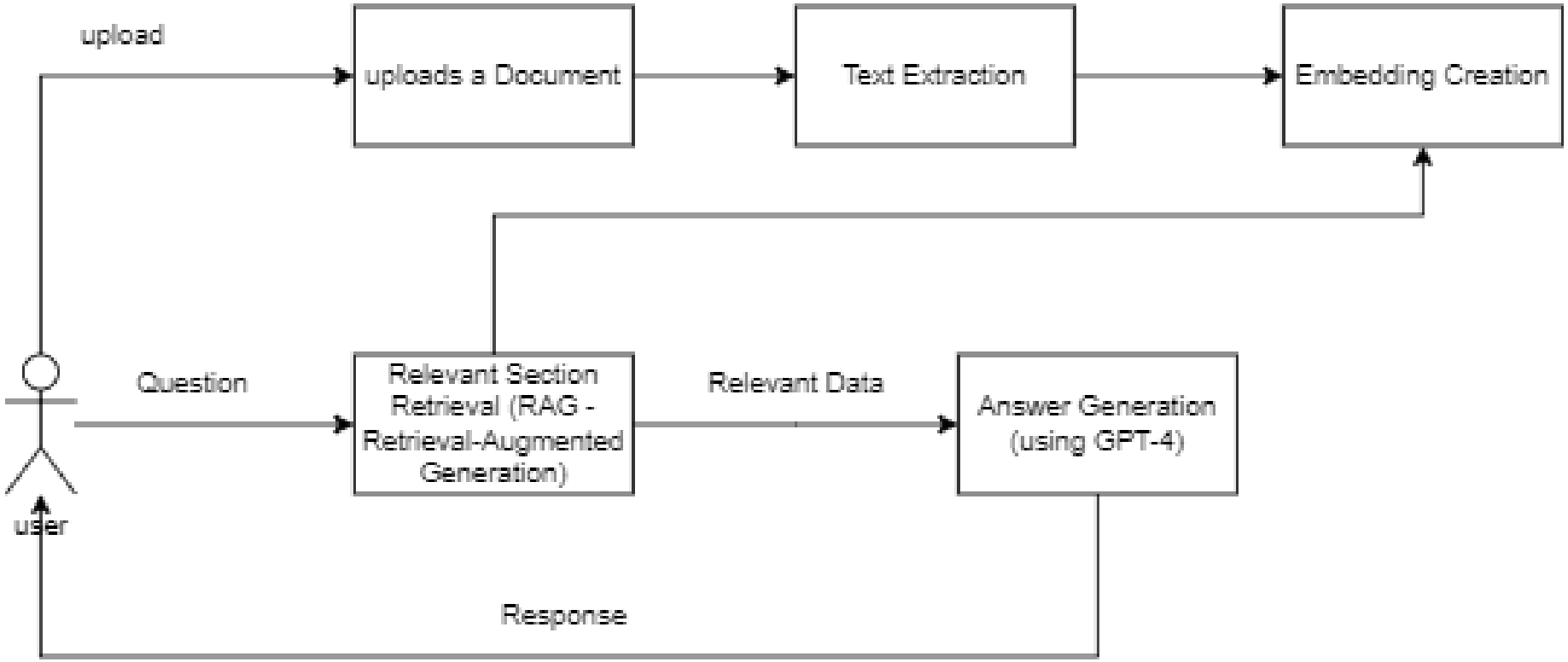
Code modules for converting uploaded document content into embeddings.
A storage mechanism (e.g., ChromaDB) to manage the generated embeddings for efficient retrieval.

RAG-based Information Retrieval:

Implementation of a Retrieval-Augmented Generation (RAG) technique to fetch the most relevant document sections in response to a user's question.

Design Diagram

HEXAWARE



Design Description:

Chatbot Interface:

A user-friendly web-based interface developed using streamlit, enabling users to upload documents and ask questions.

Backend API:

Developed using streamlit , this handles document uploads, text extraction, and communication with the AI model.

Embedding Creation: Converts document content into vector embeddings using ChromaDB, allowing for efficient retrieval of relevant sections.

RAG Technique (Retrieval-Augmented Generation):

Retrieves the most relevant sections of the document based on the user's question, providing the context for GPT-4 to generate accurate answers.

Azure OpenAI Integration:

Uses GPT-4 for generating answers based on the user's question and the retrieved document content.

Storage and Retrieval:

Uses a vector database to store and retrieve document embeddings for fast access during question processing.

Workflow:

Document Upload:

Users upload a PDF document through the web interface.

The document is processed to extract text using PyPDF2.

Embedding Creation:

Extracted text is converted into embeddings using ChromaDB.

These embeddings are stored for efficient retrieval.

Question Submission:

Users submit their question through the chat interface.

The backend API processes the question and uses the RAG technique to find relevant document sections.

RAG-based Retrieval:

The system queries the stored embeddings to find the most relevant sections of the document.

The retrieved sections serve as context for generating an accurate answer.

Answer Generation:

The question and retrieved context are sent to Azure OpenAI's GPT-4.

GPT-4 generates a detailed answer using the provided context.

Response Display:

The generated answer is displayed to the user on the web interface.

Positive Test Cases

HEXAWARE

Document Upload Success
Objective: Verify that the system accepts and processes valid PDF documents.
Expected Result: Successful upload message and generation of embeddings.

Question Submission and Answer Retrieval
Objective: Ensure the system retrieves relevant document sections and generates an accurate answer.
Expected Result: Correct answer displayed based on the document content.

Embedding Creation and Storage
Objective: Verify that document embeddings are created and stored correctly.
Expected Result: Embeddings are generated and saved for future retrieval.

RAG Technique Functionality
Objective: Validate that the RAG process retrieves the correct document sections.
Expected Result: The most relevant sections are retrieved for the given question.

Azure OpenAI Integration
Objective: Test that the GPT-4 model generates answers based on the question and provided context.
Expected Result: GPT-4 returns a coherent and accurate answer.

Negative Test Cases

HEXAWARE

Document Upload Failure
Objective: Ensure the system handles unsupported file formats.
Expected Result: The system rejects the upload and displays an error message.

Question Submission with No Document
Objective: Verify behavior when a user submits a question without any uploaded document.
Expected Result: Error message indicating that a document is required for question answering.

Incomplete Embedding Data
Objective: Test system behavior with incomplete or failed embedding creation.
Expected Result: The system should handle the error and prompt for re-upload.

RAG Technique with Irrelevant Data
Objective: Test the system’s handling of irrelevant data during retrieval.
Expected Result: The system should indicate that no relevant data was found.

Azure OpenAI Downtime
Objective: Ensure the system handles Azure OpenAI service unavailability gracefully.
Expected Result: Display a message indicating that the service is temporarily unavailable.

Tools and Code details

HEXAWARE

Third party tools Details:

Tools name	Open source/Licensed	URL	Purpose
ChromaDB	Open Source	https://pypi.org/project/chromadb/	Stores and manages vector embeddings of document text for efficient retrieval during the RAG process.
PyPDF2	Open Source	https://pypi.org/project/chromadb/	Extracts text from PDF documents for further processing
Azure OpenAI GPT-4	Licensed	https://azure.microsoft.com/en-us/products/ai-services/openai-service/	Generates answers based on user questions and the context retrieved from document embeddings.

Technologies used

HEXAWARE

Technology name	Version
Python	Version 3.8