Problem Statement and Purpose

Problem Statement

You have a corpus about the information about a business that sells wines. They have their own website which customers often visit. But in this era of GenAI, people are looking for quick answers. They do not want to read anything; everybody wants quick answers.

Purpose

Taking this into concern, the business owners want to deploy a chatbot on their website. They want the chatbot to be able to answer questions from this corpus and not use other information. If the user asks anything that is not there in the corpus, it should just tell them to contact the business directly.

Sample Question

A sample of question-answers is given in the sample questions PDF.

Project Report

A. Overall Approach

Objective:

The main objective of the project is to develop a chatbot using Streamlit that can answer questions

based on the content of provided documents. The chatbot utilizes various natural language

processing (NLP) tools to process, embed, and retrieve document data to generate accurate

responses.

Steps Taken:

1. **Environment Setup:** Load necessary environment variables for API keys.

2. **Library Imports:** Import required libraries for NLP, document processing, and embeddings.

3. **Streamlit Application Setup**: Set up the title and user interface elements.

4. **LLM Initialization:** Initialize the language model with the required API key and model name.

5. **Prompt Template Creation:** Create a template for the prompt that will be used to generate

responses.

6. **Embedding Function:** Define a function to process documents, split them into chunks, and create

embeddings.

7. User Interaction: Set up user inputs to receive questions and trigger document embedding

creation.

8. Document Retrieval and Response Generation: Implement retrieval and response generation

using the language model and vector embeddings.

B. Frameworks/Libraries/Tools Used

1. Streamlit

Usage: To create the web interface for user interaction.

Location: Throughout the script for setting up the title, text input, buttons, and displaying results.

2. LangChain

Usage: To manage text splitting, combining documents, and creating chains for processing.

Location: Used in text splitting (RecursiveCharacterTextSplitter), creating document chains

(create_stuff_documents_chain), and retrieval chains (create_retrieval_chain).

3. FAISS (Facebook AI Similarity Search)

Usage: For creating vector embeddings and enabling fast similarity search.

Location: Used to store and retrieve document embeddings.

4. PyPDFDirectoryLoader

Usage: To load and ingest documents from a specified directory.

Location: Used in the vector_embedding function for loading PDF documents.

5. Google Generative AI Embeddings

Usage: To generate embeddings for the text chunks.

Location: Used in the vector_embedding function to embed documents.

6. dotenv

Usage: To load environment variables from a .env file.

Location: Used at the beginning of the script to load API keys.

C. Future Scope

1. Enhanced Document Formats

- **Description**: Extend support to additional document formats like DOCX, TXT, and HTML.
- **Benefit**: Broaden the range of documents the chatbot can process.

2. Advanced Retrieval Techniques

- **Description**: Implement more advanced retrieval techniques like context-aware retrieval or semantic search.
- **Benefit**: Improve the accuracy and relevance of the responses.

3. Real-time Updates and Online Document Loading

- **Description**: Integrate real-time document updates and online document loading features.
- **Benefit**: Ensure the chatbot always has the most up-to-date information, and can load documents from both online sources and local devices.

- **Implementation**:
  - **Online Document Loading**: Integrate APIs to fetch documents from online sources (e.g., Google Drive, Dropbox).
  - **Device PDF Loading**: Allow users to upload PDFs directly from their devices.
  - **Real-time Data Integration**: Use webhooks or scheduled tasks to update embeddings as new documents are added or existing ones are modified.

4. User Authentication and Sessions

- **Description**: Add user authentication and session management.
- **Benefit**: Provide personalized experiences and maintain state across user interactions.

5. Enhanced UI/UX

- **Description**: Improve the user interface and experience with more interactive elements and better design.
- **Benefit**: Make the chatbot more user-friendly and engaging.

6. Integration with External APIs

- **Description**: Integrate with external APIs for additional data sources or functionalities.
- **Benefit**: Enhance the capabilities of the chatbot by incorporating more diverse information.