Overall Approach

The goal of this project was to develop a chatbot for Jessup Cellars that can interact with users, provide information from a given corpus, and maintain conversation context. The chatbot leverages Google Generative AI for generating responses and uses a combination of a PDF corpus and a JSON file for reference. The main steps included:

1. Corpus Preparation: Extracted text from a PDF document and a JSON file containing sample questions and answers.

2. Preprocessing: Cleaned and tokenized the corpus text.

3. Similarity Check: Implemented a TF-IDF vectorizer and cosine similarity to ensure the chatbot responses are within the scope of the corpus.

4. Conversation Management: Maintained conversation history to handle contextual queries. Choosing a LLM for QA text generation.

5. Frontend Integration: Developed a user interface with instant greeting and message

handling capabilities. Improved the chatbot UI for it to be easy on the eyes and look a bit modern.

Frameworks/Libraries/Tools Used

1. Flask: Used for building the web application and creating RESTful endpoints.

   - Where used: In the entire backend structure to handle HTTP requests and render HTML templates.

2. Google Generative AI (google.generativeai): Used for generating chatbot responses just because they provide free tokens in their API.

   - Where used: For creating dynamic responses to user queries.

3. PyPDF2: Used for reading and extracting text from PDF documents.

   - Where used: To load and extract text from the 'Corpus.pdf' file.

4. nltk (Natural Language Toolkit): Used for text preprocessing, tokenization, and stopwords removal.

- Where used: In preprocessing corpus text to clean and tokenize for checking the relevance of questions.

5. scikit-learn: Used for TF-IDF vectorization and cosine similarity calculation.

- Where used: To create vector representations of the corpus and user queries and calculate similarity scores.

6. HTML/CSS/JavaScript: Used for developing the frontend interface.

- Where used: In `index.html` for the chatbot UI and interactivity.

Problems Faced and Solutions

1. Ensuring Relevant Responses:

- Problem: The chatbot occasionally generated responses to the questions which were out of corpus instead of displaying the said message.

- Solution: Implemented a TF-IDF vectorizer and cosine similarity check to filter out-of-corpus responses.

2. Maintaining Conversation Context:

- Problem: Handling context in a long conversation where user queries could refer to previous messages.

- Solution: Maintained a conversation history and included it in the prompt for generating responses based on previous prompts if required.

3. Web Development:

- Problem: In General issues regarding Web Development. I had very little experience in it before this assignment now I can say I know something in Web Development.

- Solution: Used ChatGPT in solving my problems and learning Basics of CSS . HTML and Java are something I am familiar with.

Future Scope

1. Enhanced Natural Language Understanding:

   - Integrate more advanced NLP models to better understand and process user queries. For example: using the tranformers library or using a open source free LLM for QA. It would help in cost reduction.

2. Multi-turn Conversation Management:

   - Implement more sophisticated context tracking to handle longer and more complex conversations.

3. Knowledge Base Integration:

   - Integrate with a dynamic knowledge base to provide up-to-date information and handle a broader range of queries.

4. Voice Interaction:

   - Add support for voice input and responses to make the chatbot more accessible and user-friendly like gpt-4o.

5. Personalization:

   - Implement user profiling to personalize interactions based on previous conversations and user preferences.

6. Analytics Dashboard:

   - Develop an admin dashboard to monitor chatbot interactions, gather insights, and improve response quality.

7. Multilingual Support:

   - Extend the chatbot's capabilities to handle multiple languages to cater to a diverse user base.