**Sai Karthik Naladala Contribution:**

**Evaluation of Question-Answer Pairs using NLP:**

Based on the evaluation of the question-answer pairs, the average similarity score between the system's answers generated by the Gemini bot and the ground truth answers provided by the Unibuddy bot is approximately 0.703. Additionally, the average similarity score between the system's answers generated by the GPT-3.5 bot and the ground truth answers provided by the Unibuddy bot is approximately 0.60.

* Developed a Python script to evaluate the similarity between question-answer pairs.
* Implemented text preprocessing techniques including tokenization, stopword removal, and lowercase conversion.
* Utilized the cosine similarity metric to calculate the similarity score between system-generated answers and ground truth answers.
* Evaluated each question-answer pair, printing the question, system's answer, ground truth answer, and similarity score.
* Calculated the average similarity score across all question-answer pairs.
* This evaluation provides insight into the performance of the Gemini bot's answers compared to the answers provided by the Unibuddy bot.

Link: <https://github.com/TarunSiga/DSCapstoneProject/blob/saikarthiknaladala/Assignment%205.1/Evaluation%20qa%20pairs.ipynb>

Contribution Report:

Two models were assessed: UniBuddy and Perplexity. The evaluation criteria encompassed relevance to the question, coverage of resources and support, clarity and coherence of the answers, and the provision of additional information.

UniBuddy:

Relevance to Question: 5

Coverage of Resources and Support: 5

Clarity and Coherence: 5

Additional Information: 4

Total Score for UniBuddy: 20/20

UniBuddy demonstrated exemplary performance across all criteria, achieving a perfect score of 20 out of 20. Its responses were highly relevant, comprehensive, clear, and coherent, with an added value of additional information.

Perplexity:

Relevance to Question: 4

Coverage of Resources and Support: 3

Clarity and Coherence: 2

Additional Information: 3

Total Score for Perplexity: 12/20

Perplexity, while providing reasonable responses, showed some areas for improvement. It scored slightly lower than UniBuddy, with particular weaknesses identified in clarity, coherence, and coverage of resources and support. Overall, UniBuddy outperformed Perplexity, showcasing superior capabilities in generating relevant, comprehensive, and coherent responses to user queries. Further refinement may be necessary for Perplexity to enhance its performance and align more closely with the standards set by UniBuddy.

**Back End Development :**

♣ Created a visually appealing login/register form using HTML and CSS.  
♣ Integrates Firebase authentication for user registration and login.  
♣ Provides functionality to switch between the login and registration forms.

♣ Includes a section to display a Streamlit app using an iframe.

♣ Initializes Firebase using the provided configuration.  
♣ Implements user registration and login functions using Firebase's authentication methods.

♣ Validates user input for email, password, full name, university, and phone number.  
♣ Redirects the user to the Streamlit application after successful registration or login.  
♣ Defines styles for various elements of the form, including fonts, colors, and dimensions.

♣ Creates a visually appealing layout with background gradients and rounded corners.  
♣ Ensures consistent styling across different input fields and buttons.  
♣ Sets up a Streamlit app with a chat interface.  
♣ Integrates Firebase Firestore for storing chat history.  
♣ Implements speech recognition using the SpeechRecognition library for voice input.  
♣ Displays chat history and handles user input with appropriate responses.

**Firebase Integration**: Integrated Firebase Admin SDK to interact with Firestore, enabling seamless access to Firestore collections and documents within the Streamlit app.

**Dynamic Collection Retrieval:** Developed a function to dynamically retrieve all collection names from Firestore, ensuring flexibility in selecting data for export.

**Efficient Data Extraction:** Implemented a method to fetch all documents from selected collections, extracting relevant data fields such as prompt and response for export.

**User-Friendly Interface:** Designed an intuitive Streamlit interface allowing users to easily select collections and initiate data export with options for single or multiple selections.

**Automated CSV Generation:** Utilized the CSV module to automatically generate a structured CSV file containing extracted Firestore data, simplifying data management and analysis.

**Challenges faced:**

**Firebase Integration:** Setting up Firebase Admin SDK and Firestore presented challenges due to authentication and connection configurations.

**Streamlit UI:** Designing a user-friendly interface with Streamlit required careful consideration of layout and functionality to ensure smooth user experience.

**CSV Export**: Implementing CSV export functionality involved handling file paths, permissions, and ensuring proper data formatting.

**Data Retrieval:** Retrieving data from Firestore collections involved dealing with asynchronous requests and ensuring correct data transformation for further processing.

**Prompt Integration:** Integrating prompt delivery to a fine-tuned model and retrieving responses required understanding and integrating with external APIs or services, potentially involving authentication and data serialization challenges.

**Error Handling:** Implementing robust error handling mechanisms to handle potential failures in data retrieval, CSV export, or integration with external services.