# Team Contribution Report

Group – 2

University of Missouri – Kansas City COMP\_SCI5588: Data Science Capstone Yugyung Lee

April 3rd, 2024

# Team Contribution Report

# Team Contribution:

**Deepak Ayyasamy: Machine Learning Development**

**Ingestion Module Development**:

* Developed ingest.py module responsible for loading documents from specified directories and splitting them into appropriate formats.
* Implemented multithreading and multiprocessing techniques for efficient document loading and processing.
* Incorporated error handling mechanisms to manage exceptions during the document loading process.
* Created a logging system (file\_ingest.log) to track document loading progress and any encountered errors.

# Local GPT Module Development:

* Developed run\_localGPT.py module responsible for running retrieval question-answer tasks using Language Chain's RetrievalQA pipeline.
* Implemented model loading functions to load pre-trained models based on specified configurations.
* Integrated embeddings and vector stores for efficient text representation and retrieval.
* Leveraged Mistral 7B quantized version as LLM model and “instructor-large” as embedding model.
* Implemented callbacks for streaming responses and managing output verbosity.
* Provided options for users to specify device type, show/hide source documents, use history, and select model types.

# Code Refinement and Documentation:

* Refactored codebase for improved readability and maintainability.
* Added comprehensive inline documentation to explain module functionalities and code logic.
* Conducted thorough testing and debugging to ensure code correctness and reliability.
* Incorporated user-friendly CLI interfaces using the Click library for easy parameter configuration.

# Challenges Faced:

* **Concurrency Management**: Implementing efficient concurrency management strategies for document loading and processing required careful consideration to avoid race conditions and deadlocks.
* Addressed issues related to thread and process synchronization to ensure smooth execution.
* **Model Loading and Configuration:** Ensuring compatibility and proper configuration of pre-trained models posed challenges, especially in managing different model types and device types.
* Implemented robust error handling mechanisms to handle model loading failures and configuration errors gracefully.
* **Optimization and Performance Tuning:** Optimizing document loading and processing performance while minimizing resource utilization was a significant challenge.
* Explored various optimization techniques and fine-tuned parameters to achieve a balance between performance and resource efficiency.

# Tarun Siga: Front End Development

# Advanced user profile management:

* Expanded upon existing user profile management features by enabling users to further personalize their profiles with additional customization options.
* Introduced themes, color schemes, and layout options to allow users to tailor their profile interface according to their preferences and style.
* Implemented a user-friendly interface for profile customization, ensuring intuitive navigation and ease of use.

# Improved chat experience:

* Implemented typing indicators to show when a user is typing a message.
* Added read receipts to indicate when a message has been read by the recipient.
* Implemented emojis and stickers for richer communication.
* Introduced message threading to organize conversations more effectively.

**Scalability and performance optimization:**

* Conducted rigorous testing and optimization to ensure the platform's scalability and performance under heavy load conditions.
* Implemented caching mechanisms, query optimization techniques, and resource allocation strategies to enhance system responsiveness and minimize latency**.**

**Improved user-friendly interface:**

Focused on designing and implementing a streamlined and intuitive user interface using Streamlit, aimed at enhancing the overall user experience within the UniBuddy platform. The primary objective was to empower users with a user-friendly interface that facilitates effortless navigation, efficient data selection, and seamless data export capabilities.

**Streamlined export options:**

* Integrated seamless export functionalities within the interface, allowing users to initiate data export operations with just a few clicks.
* Provided users with options for single or multiple selections, enabling them to export selected data sets in a structured and convenient manner.
* Implemented clear and concise instructions and visual cues to guide users through the export process, ensuring a hassle-free experience from selection to export.

**Responsive Design:**

* Ensured responsiveness and adaptability across various devices and screen resolutions, optimizing the interface for both desktop and mobile usage.
* Leveraged responsive design principles and best practices to create a fluid and flexible layout that adjusts dynamically to different viewport sizes, ensuring consistent user experience across devices.

**Automated CSV generation:**

* Leveraged the CSV module in Python to dynamically generate structured CSV files containing the extracted Firestore data.
* Implemented robust error handling mechanisms to handle potential exceptions and edge cases during the CSV generation process, ensuring reliability and data integrity.

# Challenges Faced:

# Integration Complexity:

Integrating the Streamlit interface with Firestore and ensuring seamless communication between the front-end and backend components can pose challenges, particularly when handling complex data structures or large volumes of data. Ensuring compatibility and data consistency across different systems and platforms may require careful planning and coordination.

**User Experience Optimization:**

Designing an intuitive and user-friendly interface that caters to diverse user needs and preferences can be challenging. Balancing functionality with simplicity while accommodating various use cases and user personas requires thorough user research, prototyping, and iterative testing to ensure optimal usability and satisfaction.

**Data Parsing and Formatting:**

Extracting and formatting data from Firestore to generate structured CSV files may encounter challenges related to data consistency, schema evolution, and complex nested structures. Handling irregularities or inconsistencies in the data, such as missing values or unexpected data types, requires robust error handling and data validation mechanisms to ensure accurate and reliable results.

# Sai Karthik Naladala: 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.
* Done working on integrating functionality to send prompts to a fine-tuned model and retrieve responses. Additionally, add a logout functionality to the Unibuddy Home page where the user will be redirected to the login page after clicking on it.
* Stored updated user-friendly interface details such as Degree, Major, Current Semester fields, and Interests fields in Firebase for future reference.

**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.

**Model Evaluation Method 1:**

**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>

**Model Evaluation Method 2:**

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 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.

**Integration of Hugging Face Instructor Embedding:**

* Implemented Hugging Face Instructor Embedding to convert words into unique numerical representations.
* Demonstrated how this embedding technique creates distinct numerical vectors for each word in a sentence, akin to special fingerprint-like numbers.

**Implementation of ChromaDB:**

* Developed ChromaDB, a specialized data warehouse, to store numerical representations generated by Hugging Face Instructor Embedding.
* Utilized ChromaDB to retrieve similar sentences based on numerical representations, facilitating efficient information retrieval.

**Performance Optimization with Chunking:**

Introduced a strategy to handle large documents by splitting them into smaller chunks.

Implemented chunking to ensure efficient processing and mitigate memory issues during information retrieval.

Utilized Instructor Embedding to convert each chunk into numerical vectors, which were then

stored in ChromaDB for retrieval.

**Streamlined Information Retrieval Process:**

Established a seamless process for users to input text queries.

Demonstrated how Instructor Embedding converts text inputs into numerical representations.

Showcased ChromaDB's capability to search for similar numerical representations in the database, facilitating the retrieval of relevant answers.

**Individual Member Contributions in Percentage:**

* **Deepak Ayyasamy : 33.3%**
* **Tarun Siga : 33.3%**
* **Sai Karthik Naladala : 33.3%**