**UniBuddy – Gen AI Powered Personal Assistant**

Group – 2

Deepak Kumar Ayyasamy

Tarun Siga

Sai Karthik Naladala

University of Missouri – Kansas City

COMP\_SCI5588: Data Science Capstone

Yugyung Lee

**UniBuddy – Gen AI Powered Personal Assistant**

## Abstract:

In today's fast-paced world, students are often inundated with information about universities, making it challenging to navigate through the myriad of options and decisions they face. Navigating the complexities of university life can be a daunting task for students. UniBuddy, a virtual assistant application, aims to empower students by providing a centralized platform for accessing all relevant university information. UniBuddy acts as a centralized platform, offering students seamless access to a wealth of university information. This report details the implemented systems that contribute to UniBuddy's functionalities, including user interaction, data management, and natural language processing. This report provides a comprehensive overview of the project's objectives, methodology, and successful implementation, highlighting the innovative strategies employed to create a seamless and user-centric platform for accessing essential information about universities.

## Introduction:

As universities navigate an ever-increasing complexity in resources and support systems, students can often find themselves overwhelmed and disoriented. Information on academics, extracurricular activities, financial aid, and campus resources can be scattered across various platforms, hindering efficient information access. This report introduces UniBuddy, a virtual assistant application designed to bridge this gap and provide students with a comprehensive and seamless support system throughout their university journey.

UniBuddy goes beyond simply providing information. By leveraging natural language processing and advanced data retrieval techniques, UniBuddy aims to deliver accurate and up-to-date information in real-time. This empowers students to make informed decisions about their educational journey. Furthermore, UniBuddy personalizes the interaction by offering tailored recommendations based on individual preferences and requirements. The following sections of this report will delve into the methodologies employed to create this innovative student support system. The project's origin lies in the recognition of the many challenges students face when seeking information about academic programs, extracurricular activities, financial aid, and various other aspects of university life. Traditional methods of information retrieval often prove cumbersome and inefficient, prompting the need for a more streamlined and intuitive solution. UniBuddy emerges as the answer to this demand, offering a dynamic and responsive virtual assistant that harnesses the power of natural language processing and real-time data management to deliver accurate and personalized information to students.

## Core Functionalities:

* Secure user registration and authentication.
* Intuitive chatbot interface for student inquiries.
* Real-time data storage and communication facilitated by Firebase integration.
* Natural Language Processing for accurate information retrieval and response generation.
* Personalized recommendations based on user preferences.

## Methodology:

The development of UniBuddy is underpinned by a multi-faceted methodology that seamlessly integrates technological innovation with user-centric design principles. The methodology encompasses several key components, each contributing to the overall functionality and user experience of the virtual assistant:

**User Authentication and Registration System:**

* The foundation of UniBuddy's security and user management lies in the implementation of a robust authentication and registration system.
* Utilizing Flask, a lightweight web framework, in conjunction with Firebase Authentication services, ensures secure user authentication through various methods such as email/password, phone number, or social media logins.
* This system guarantees data privacy and integrity while providing users with a seamless and hassle-free registration process.

A screenshot of a computer

Description automatically generated

Figure 1. Conceptual Diagram

**Integration of Firebase with Streamlit for Chat Application and Data Export:**

* Streamlit, a popular open-source framework for building data-driven web applications, serves as the interface for UniBuddy's chat application.
* By integrating Firebase, a real-time NoSQL database, UniBuddy enables seamless communication between users and the virtual assistant, ensuring instant access to relevant information.
* Additionally, the Firebase Admin SDK facilitates the export of chat data to CSV format, allowing for comprehensive data analysis and insights into user interactions.

**User Authentication and Chat Application Development:**

* The development of UniBuddy's chat application is a collaborative effort involving Flask, Firebase Authentication, Firestore, and Streamlit.
* Flask provides the backend infrastructure for user authentication and session management, ensuring secure communication between users and the virtual assistant. Firebase Authentication services authenticate users' identities, while Firestore, a flexible and scalable database, stores chat data in real-time. Streamlit's intuitive interface enhances user engagement, offering a seamless chat experience within the UniBuddy platform.

**Embeddings:**

* Text embeddings play a crucial role in UniBuddy's natural language processing capabilities, enabling efficient text representation and similarity search.
* By creating embeddings from text chunks using **instructor-large** LLM embedding model, UniBuddy enhances the accuracy and relevance of its responses to user queries.
* These embeddings are stored in a vector database, facilitating fast and efficient retrieval of information based on user preferences and requirements.
* Chrome DB is used as a vector database to store all the embedding vectors.

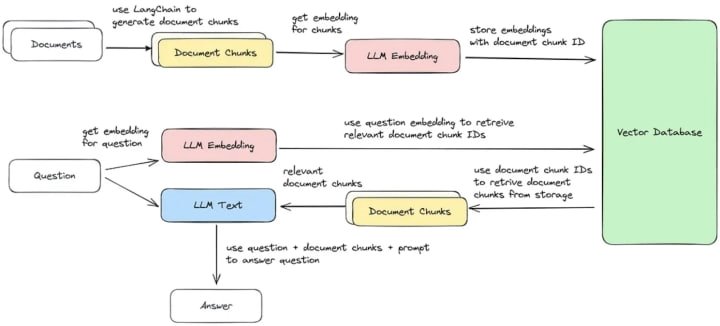


Figure 2. Embedding Workflow

**LLM Fine-tuning:**

* Once the embeddings are created, the base LLM receives the user query and it is supposed to send a response to that query.
* User query text is converted into embeddings using the same embedding model used for converting input file and the question vector is compared with all the generated vectors in the vector database, i.e. Chroma DB.
* Once a similar vector is identified, LLM retrieves the vector, decodes it into text and refines it to make it look like human generated response.
* Thus, the generated response is given as an output to the user.

**Ingestion Module Development:**

* An efficient document loading module is developed to facilitate seamless ingestion and processing of data within the UniBuddy ecosystem.
* Utilizing multithreading and multiprocessing techniques, along with robust error handling mechanisms and logging systems, ensures the smooth operation of the ingestion module. This module enables UniBuddy to efficiently process large volumes of data, providing users with timely and relevant information.

A close-up of several white rectangular boxes

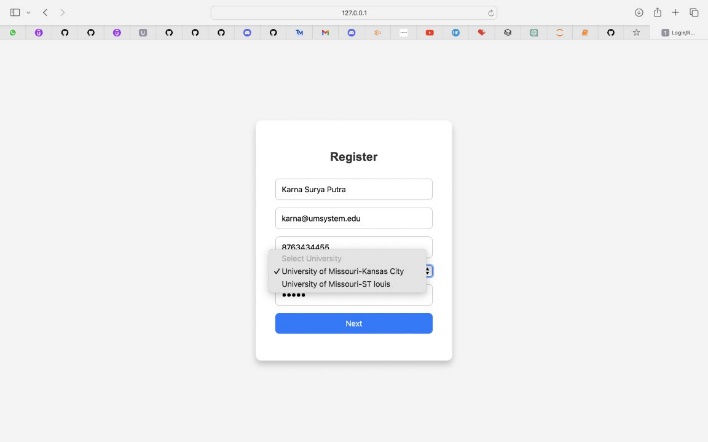
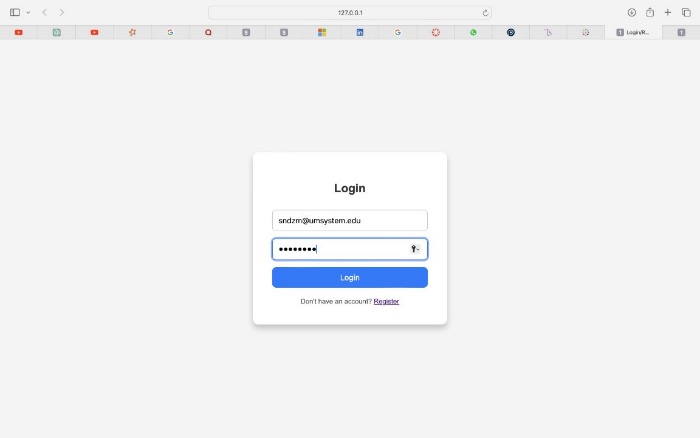
Description automatically generated

Figure 3. Model Diagram

**Local GPT Module Development:**

* UniBuddy's local GPT module is specifically designed for retrieval question-answer tasks, leveraging advanced techniques to optimize performance and resource utilization.
* Integrating embeddings and vector stores enhances the efficiency of text representation and retrieval, enabling UniBuddy to deliver accurate and timely responses to user queries.
* ­­­­By optimizing performance and resource utilization, the local GPT module ensures a seamless and responsive user experience within the UniBuddy platform.

## Results:

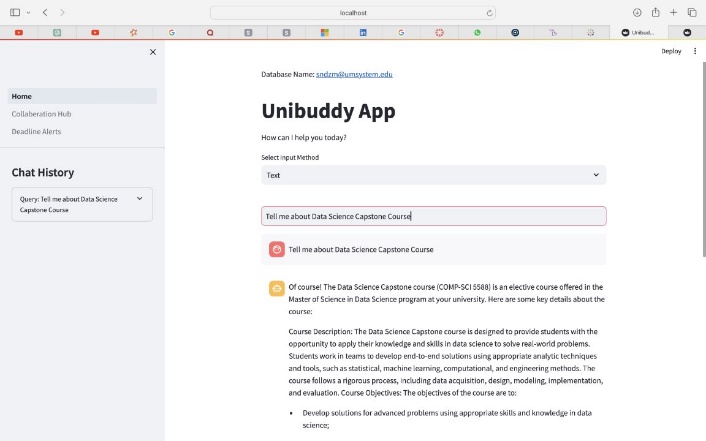


Figure 4. User Interface

It is observed that the UniBuddy application successfully answers the queries asked by user.

## Evaluation:

To evaluate the performance of our fine-tuned LLM module, we compared the results of our LLM model with the top Gen AI application in the market, like, GPT 3.5, Gemini 1.5 and Perplexity AI.​

The evaluation included quantifying the responses of different LLM responses based on four factors - ​

* **Comprehensiveness:**Comprehensiveness refers to the extent to which a response covers all necessary aspects or components of the topic or query at hand.​
* **Relevance:** Relevance pertains to how directly and closely a response addresses the specific topic, question, or query posed.​
* **Clarity:** Clarity refers to how easily understandable and coherent a response is to the intended audience.​
* **Helpfulness:** Helpfulness assesses the practical utility or value of a response in aiding the audience's understanding or addressing their needs.​

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aspect** | **UniBuddy** | **GPT 3.5** | **Gemini** | **Perplexity** |
| Reality | 10 | 10 | 10 | 6 |
| Comprehensiveness | 8 | 8 | 10 | 4 |
| Clarity | 10 | 8 | 10 | 6 |
| Helpfulness | 8 | 8 | 10 | 4 |
| Overall Score | 9 | 8.5 | 10 | 5 |

Table 1. Comparative Analysis

We calculated this for 25 different questions and then arrived at mean values. Results suggested Unibuddy outperformed GPT 3.5 and Perplexity AI and came second to Gemini.

​**Challenges:**

**Data Collection:** Faced difficulties in finding domain specific data that is latest and has enough information.

**Resource intensive functionalities:** File ingestion to create embeddings required high end processor to withstand the resource intensive task of fine-tuning with domain data and to perform in a efficient manner.

**Lack of community:** Since LLMs burst out into the scene, the community is still growing and building anything including a niche idea takes time and effort to tie all loose ends.

## Future work:

**Transition to a Software as a Service (SaaS) Model:**

* Implement subscription-based access to UniBuddy, offering users access to premium features and personalized services.
* Develop a pricing strategy that caters to different user segments, including individual students, educational institutions, and corporate clients.
* Explore partnerships with universities and educational organizations to integrate UniBuddy into existing platforms and services.

**Expansion of Features and Services:**

* Enhance UniBuddy's capabilities beyond university-related information to include career planning, internship opportunities, and personal development resources.
* Integrate additional modules for academic assistance, such as study guides, exam preparation materials, and virtual tutoring services.
* Incorporate features for skill development and certification programs, providing users with opportunities to enhance their professional qualifications.

**Integration with External Platforms and APIs:**

* Integrate UniBuddy with external platforms and APIs to access additional data sources and services, such as academic databases, career portals, and social media networks.
* Collaborate with industry partners to leverage their expertise and resources in specific domains, such as job placement, internship coordination, and professional networking.
* Develop standardized interfaces and protocols for seamless integration with third-party applications, ensuring interoperability and ease of use for end-users.

## Conclusion:

UniBuddy project represents a significant milestone in the realm of virtual assistant technology, offering a comprehensive and user-centric solution for navigating the complexities of university life. Through meticulous implementation of advanced technologies such as natural language processing, real-time data management, and user authentication systems, UniBuddy has successfully revolutionized the way students interact with university-related information.

## References:

https://towardsdatascience.com/how-to-build-an-llm-from-scratch-8c477768f1f9

https://thomascherickal.medium.com/how-to-create-your-own-llm-model-2598615a039a

# 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.

**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.

**Individual Member Contributions in Percentage:**

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

## Github Repositories

Repo Link: <https://github.com/TarunSiga/DSCapstoneProject>

***Individual GitHub links (Branched with project):***

Deepak Ayyasamy:

<https://github.com/TarunSiga/DSCapstoneProject/tree/DeepakAyyasamy>

Tarun Siga:

<https://github.com/TarunSiga/DSCapstoneProject/tree/TarunSiga>

Sai Karthik:

<https://github.com/TarunSiga/DSCapstoneProject/tree/saikarthiknaladala>