COMPASS

Comprehensive Master's Program Assistant for Student Success

Project Overview

Students rely on scattered resources like university websites, forums, and general-purpose chatbots. These methods lack personalization, integration, and real-time data insights. The goal of the COMPASS project is to support international students who are looking to pursue their master's degree by providing personalized guidance throughout their university application journey. The system considers key factors such as the student's field of study, preferred location, budget, and cost of living to recommend the best-fit universities. It also considers important application requirements, such as ACT scores, and helps students understand the criteria for admission to various institutions. Furthermore, COMPASS provides insights into job opportunities, ensuring students are aware of potential career prospects in different regions. Beyond academic and career-related advice, COMPASS offers moral support that not only answers questions but also encourages and motivates students during the stressful application process. By combining practical guidance with emotional support, COMPASS aims to make the application experience smoother and more manageable for students, empowering them to make informed decisions about their future education.

App Flow (How does the system work?)

1. Login Page:

a. When users first visit COMPASS, they are prompted to log in with their username. If it's their first time, a new user profile is created, and they are guided to set their preferences.

b. After logging in, users are authenticated, and their data is loaded from the system, including their preferences and chat history.

2. User Preferences:

a. Upon login, the system displays the user's saved preferences (field of study, budget, preferred locations, and weather preferences) on the sidebar.

b. If preferences haven't been set, the user is prompted to fill out a form with their desired field of study, budget, location preferences, and weather conditions.

c. The user can edit their preferences at any time by clicking the "Edit Preferences" button.

3. Sidebar Information:

a. The sidebar displays helpful information and tools, including:

i. About COMPASS: A brief introduction to the system.

ii. Tips for using COMPASS: Instructions to help users get the most out of the platform.

iii. Today's Date & Username

iv. User's Preferences

v. Button to Log Out

4. Top Recommendations:

a. Users can click on the "Top 3 Recommendations" button, which will generate personalized university recommendations based on their entered preferences (field of study, budget, location, weather, etc.).

b. Users can follow up by asking more specific questions about any of the recommended universities.

5. Explore University Details:

a. Users can ask questions such as:

i. "Tell me more about Georgia Tech's program?"

ii. "What are the living costs in Atlanta?"

iii. "How's the job market for data science in the Northeast?"

iv. "Compare the weather between Boston and Miami?"

b. The system provides detailed information about the universities, cost breakdowns, campus life, and weather conditions.

6. Application Tracker & Checklist:

a. The user can download an application tracker template in CSV format to edit & keep track of their university applications, deadlines, and required documents.

b. They can also download a checklist document to ensure they have all necessary materials for each application.

7. Chat History:

a. The system keeps a chat history where both user and assistant messages are stored. This allows users to reference previous interactions.

b. Users can clear the chat history at any time by pressing the "Clear Chat" button.

8. Logout Option:

a. Once the user is finished, they can log out by pressing the Logout button on the sidebar. This will end their session and ensure their data is saved for future use.

Features (Strengths)

1. Personalized Recommendations:

a. The system offers tailored university suggestions based on user preferences such as the field of study, budget, location, and weather, ensuring recommendations align with individual needs. The bot's clear communication and actionable recommendations prioritize transparency, enabling students to make informed decisions confidently.

2. User-Friendly Interface:

a. The platform provides an easy-to-use, interactive chat interface for users to ask questions and get immediate responses about universities, application requirements, living costs, and job prospects.

b. Users can edit their preferences anytime, making it convenient to refine their recommendations as their needs evolve.

c. By ensuring an user-friendly interface and lightweight infrastructure, we reduce barriers for students from diverse backgrounds, promoting equal access to valuable resources

3. Comprehensive Information:

a. The system offers detailed insights into various aspects of university selection, including:

i. University programs

ii. Living costs in different locations

iii. Job prospects in various fields

iv. Weather conditions in preferred regions

4. Moral Support:

a. The chatbot not only provides valuable information but also offers emotional support, responding empathetically to user queries, which is particularly helpful for students facing the pressures of applying abroad.

5. Chat History and Continuity:

a. The system maintains a history of interactions, allowing users to review past conversations and continue where they left off without losing context.

6. Customizable Preferences:

a. Users can input and update their preferences for fields of study, budget, locations, and weather. The system remembers these settings, improving the experience with each login.

7. Multi-Functional Features:

a. The platform includes a checklist to ensure all necessary application materials are in place. It offers various functions, such as providing recommendations, exploring details, and sharing useful insights about different locations.

Limitations (Weaknesses)

1. Limited Data Sources:

a. The system relies on predefined responses for university information, which may not include all universities or the most current data, limiting its ability to address niche or recent queries.

2. Dependence on User Input:

a. The quality and accuracy of recommendations depend on the specificity of the user preferences. Vague or incomplete input can result in inaccurate or less relevant suggestions.

3. No Real-Time Updates:

a. The system does not incorporate real-time data changes, such as shifts in living costs, tuition fees, or job market conditions, which could lead to outdated advice.

Technical Details

API Usage

The COMPASS project integrates multiple APIs to deliver a dynamic and personalized user experience. OpenWeather API provides real-time weather updates for user-selected cities, enabling students to compare climates and make informed decisions based on their preferences. This feature enhances usability by embedding weather insights directly into the chatbot. The OpenAI API powers the conversational capabilities of COMPASS, using GPT-4o Mini for generating detailed, context-aware responses and *text-embedding-ada-002* for semantic search. This allows the system to offer tailored university and job recommendations based on user preferences like field of study, budget, and location.

Data Used and Data Handling Processes:

University Data: The system sources university data from the niche website specializing in providing detailed information about universities. This websites gives insights into university programs, application requirements, and general university-related data, though the data may not always be the most up-to-date or comprehensive.

Job Market Data: For job market insights, the project pulls data from an external job market website that tracks industry trends, demand for specific job roles, and regional job prospects. This helps guide students on job opportunities related to their field of study.

Living Expenses: The living expenses data comes from the website that offer cost-of-living comparisons across various locations, helping students understand the financial implications of studying in different regions.

RAG (Retrieval-Augmented Generation) Usage

The system uses a Retrieval-Augmented Generation (RAG) process by retrieving specific data points (like living expenses or job market information) from the vector database and then generating contextual responses for the user. This allows the system to combine the latest available data with its predefined responses for more accurate answers.

Vector Database Usage

ChromaDB serves as the backbone for vector-based data storage and retrieval. This vector database stores embeddings (living expenses, university, job trends) that enable the system to generate precise and contextually relevant recommendations. Additionally, ChromaDB facilitates session memory persistence, allowing the platform to recall user preferences and previous interactions. This ensures continuity and personalization, creating a cohesive experience where users can pick up right where they left off.

Functions/Tools Used

COMPASS application uses several user-defined functions to handle different features. Below are key functions and their purposes:

User Management

authenticate_user(username: str) -> bool

- Authenticates users and initializes session data, including preferences, chat history, and application tracking.
- Ensures personalized user sessions with persistent data.

load_user_data() -> dict / save_user_data(data: dict)

- Handles loading and saving user data to a JSON file for maintaining long-term memory across sessions.

Chatbot Interaction

get_recommendations(query: str) -> str

- Processes user queries, retrieves relevant context (e.g., university info, living expenses), and generates responses using the OpenAI API.
- Combines user preferences and data from ChromaDB collections for tailored advice.

is_relevant_query(query: str) -> bool

- Determines if a query aligns with supported topics like universities, costs, job markets, and weather.

Data Retrieval and Management

get_university_info(query: str) -> str

- Fetches university-related information stored in ChromaDB based on semantic search.

get_living_expenses(state: str) -> str

- Retrieves living expense data from ChromaDB for a specified state, providing insights on housing, utilities, and other costs.

get_job_market_trends(field: str) -> str

- Queries job market projections, such as growth rates and median salaries, to guide career planning.

get_weather_info(location: str) -> str

- Uses the OpenWeather API to fetch real-time weather details like temperature, humidity, and conditions for a given city.

Application Tracking

generate_application_tracker_template() -> bytes

- Creates a downloadable CSV template for tracking university applications, including example fields like deadlines and costs.

Utility Functions

generate_checklist_docx() -> BytesIO

- Generates a downloadable Word document containing a detailed university application checklist.

initialize_chromadb() / load_initial_data()

- Initializes ChromaDB collections and loads essential data (e.g., university details, living expenses) for efficient querying.

save_chat_history(username: str, chat_history: list)

- Stores the user's chat history, ensuring continuity in interactions.

Third-Party Libraries

The platform relies on several libraries and tools to enhance functionality:

1. **OpenWeather API**: This API is used to fetch weather data based on user preferences. It helps the system generate accurate weather-related recommendations for students, ensuring they choose universities in climates they are comfortable with.
2. **Streamlit**: The system is built using Streamlit, which is a third-party library for creating interactive web applications. It is used to build the user interface and manage the display of chat messages, preferences, and data.
3. **ChromaDB**: This vector database tool helps store user data, preferences, and past interactions, ensuring a seamless user experience.
4. **PyPDF2** (for some PDF-related tasks): This tool is used for extracting text from PDFs if required for certain functionality.
5. **Docx** and **Pandas**: Handle document generation and data manipulation tasks, supporting customized and practical user outputs.

Ethical Implications

The COMPASS project was designed with ethical considerations in mind, recognizing its potential impact on users' decision-making and the sensitivity of the information it provides. Below are the key ethical implications and the measures taken to safeguard against potential risks:

Misinformation:

Providing inaccurate or misleading information about universities, living costs, or job prospects could adversely affect users' decisions and financial planning.

Bias in Recommendations:

The chatbot may inadvertently favor certain universities or regions, leading to unequal representation.

Harmful or Inappropriate Discussions:

Users could prompt discussions on harmful or irrelevant topics, leading to ethical and reputational risks.

Privacy Concerns:

Storing and processing user preferences and chat histories may raise concerns about data security and privacy.

Safeguards Implemented

Prompt Engineering for Content Moderation:

The chatbot's responses are guided by prompt engineering to limit discussions to relevant, ethical, and beneficial topics.

Example:

Queries outside the scope of education, weather, job prospects, or university-related topics are identified as irrelevant. If a query is flagged as harmful or inappropriate, the chatbot responds with a polite refusal and redirects users to permissible topics.

Controlled Knowledge Scope:

The chatbot's database is curated to include only verified and reliable sources, such as university websites, government labor statistics, and weather APIs. This ensures the information provided is accurate and relevant, reducing the risk of misinformation.

Bias Mitigation:

Recommendations are generated based on user preferences, such as budget, location, and field of study, rather than promoting specific universities or regions. Feedback loops ensure that user satisfaction drives improvements in the system's impartiality.

Privacy Protection:

User data (e.g., preferences and chat history) is stored in JSON files without external sharing.

Transparency in Responses:

The chatbot explicitly mentions its limitations when queried about topics it cannot handle or data it does not possess. This builds trust and ensures users are aware of the scope of its expertise. To ensure that the answers provided by the bot truly help students, our team conducted testing with a variety of real-world queries. These queries were designed to simulate the diverse needs of international students, including university selection, cost analysis, job market trends, and weather comparisons. We evaluated the bot's responses based on accuracy, relevance to user preferences, and clarity of communication. Each team member validated responses against reliable sources, such as university websites and labor market data, to ensure factual correctness. Based on feedback from these tests, we iteratively refined the prompts and database queries to improve precision and eliminate ambiguity. This process has allowed us to create a bot that delivers reliable, actionable, and personalized insights, effectively addressing the needs of students.

Ethical Interaction Guidelines:

The chatbot is programmed to:

1. Avoid sensitive or harmful discussions.
2. Provide neutral, fact-based advice.
3. Direct users to appropriate resources for non-supported queries.

How Prompt Engineering Helps

1. The chatbot only engages with supported topics, like university selection, living costs, job prospects, and weather.
2. Responses are structured to align with ethical guidelines, avoiding biases or misinformation.
3. Harmful or sensitive topics (e.g., political, medical, or legal advice) are flagged, and users are politely redirected to appropriate resources.
4. By focusing on usability, inclusivity, and ethical responsibility, the bot upholds its mission to support students in their academic and career journeys while fostering trust and reliability.

Advanced Topics Used in COMPASS

1. Long-Term Memory:

The chatbot retains user data (preferences, chat history, application tracking) using persistent storage in JSON files and ChromaDB. ChromaDB stores structured data (universities, living expenses, job trends) and retrieves relevant information via OpenAI embeddings. This enhances user experience by providing personalized, context-aware responses across sessions.

2. LLM Evaluation:

We tested multiple models, including OpenAI's GPT-4, Claude, and Google Vertex. GPT-4o Mini was chosen for its precise, relevant responses, low latency, and cost-efficiency. The evaluation involved qualitative testing with diverse queries, assessing metrics like accuracy, relevance, latency, and cost per API call, ensuring it met COMPASS's goals effectively.

3. Higher-Level APIs – LangChain-Like Principles:

Although LangChain itself was not used, the system adopts similar methodologies with a modular approach to handle sequential operations, such as data retrieval, embeddings, and querying. OpenAI's *text-embedding-ada-002* powers semantic search, improving accuracy and maintainability in complex workflows.

App Link: https://compass-education.streamlit.app/

App Demo Link: https://www.youtube.com/watch?v=rXJV2C41s28

Individual Section

Shashank Guda

What I Did

As the lead contributor to COMPASS, I handled coding, hosting, and ensuring the application's overall functionality. I developed key features such as API integration (OpenAI, OpenWeather), ChromaDB for data retrieval, and user-centric tools like application tracking and checklist downloads. Additionally, I led the team by coordinating efforts, troubleshooting technical issues, and managing project timelines. I also implemented prompt engineering to enhance chatbot responses and ethical safeguards.

Challenges Faced

One major challenge was integrating LangChain and ChromaDB, as compatibility issues caused significant delays. The complexities of using LangChain for building agents and managing workflows made it difficult to align with the project's goals. After spending considerable time troubleshooting, we decided to pivot ourselves to a simpler system. By using modularized functions and prompt templates, we streamlined the architecture while maintaining functionality. This approach reduced complexity, improved efficiency, and ensured compatibility with the existing database and API infrastructure.

Lessons Learned About Working in a Team

This project taught me the value of adaptability and collaboration. When faced with technical challenges, the team collectively evaluated alternatives, leading to better decision-making. Open communication and clear delegation helped us stay on track despite setbacks, ensuring every member contributed effectively.

Applying These Lessons in Professional Settings

In future projects, I will prioritize simplicity in design to avoid overcomplicating systems unnecessarily. I also aim to foster teamwork by encouraging open discussions to tackle challenges collaboratively. This experience reinforced the importance of balancing innovation with practicality, ensuring projects remain both feasible and impactful.

Rajat Rohilla

What I Did

As a key contributor to the team, I implemented data retrieval functions and integrated external APIs. This involved ensuring the seamless flow of data between the chatbot and

various APIs, optimizing the backend processes to enhance performance, and addressing compatibility issues during integration.

Challenges Faced

One major challenge was ensuring consistency in API responses, as some endpoints occasionally returned unexpected data formats, requiring additional error handling. Another hurdle was coordinating with team members on prompt designs and clarifying roles, which sometimes led to delays in task execution due to misaligned expectations.

Lessons Learned About Working in a Team

1. Importance of Communication: Open and regular communication significantly reduced misunderstandings, helping align tasks with project goals.

2. Collaborative Problem-Solving: Brainstorming with teammates helped overcome technical roadblocks more effectively than working in isolation.

3. Adaptability: Understanding and accommodating different working styles within the

Applying These Lessons in Professional Settings

These experiences underscored the value of clear communication and the need for adaptability when working in diverse teams. In professional settings, I will prioritize transparent updates to stakeholders, leverage collaborative tools for better team coordination, and proactively address challenges by seeking collective input. These skills will help me navigate complex projects and contribute meaningfully to organizational goals.

Rithika Gurram

What I Did

I played a key role in creating and implementing a personalized chatbot system designed to support international students in their academic journey. My contributions included:

- Taking charge of university data collection to provide accurate and comprehensive information about programs, living costs, job opportunities, and admission requirements.
- Building the foundational chatbot framework using Streamlit, integrating features like tailored recommendations to enhance user engagement.
- Compiling a thorough report to outline the project's technical details and ethical considerations.
- Refining the user interface, performing usability tests, and ensuring the chatbot was user-friendly and ready for deployment.

Challenges Faced and How They Were Overcome

- **Data Integration**: Handling diverse data sources for university information, job markets, and weather required using niche APIs and databases. Collaborative teamwork and reliable tools like the OpenWeather API ensured the accuracy of our data.
- **Personalization Features**: Implementing advanced features like RAG and database management was complex. We simplified the process by dividing tasks into manageable modules and leveraging tools like Streamlit and ChromaDB.
- **User Interface Design**: Creating an intuitive interface required continuous testing and user feedback, which helped us improve usability with each iteration.
- **Team Coordination**: Communication gaps and task alignment issues were resolved through regular meetings and shared project management tools, fostering smooth collaboration.

Lessons Learned About Working in a Team

Working on this project underscored the importance of effective communication and clear role delegation. By aligning tasks with each team member's strengths, we streamlined workflows and tackled challenges efficiently. The experience taught me to embrace flexibility, as we often had to adjust our strategies based on user feedback or technical hurdles. A collaborative atmosphere built on mutual respect fostered creativity and problem-solving throughout the project.

Applying These Lessons in Professional Settings

- **Stronger Team Dynamics**: I will prioritize open communication and active engagement to build productive and harmonious teams.
- **Flexible Project Management**: My ability to adapt and focus on incremental improvements will help me manage evolving projects effectively.
- **Collaborative Problem-Solving**: I've learned to use brainstorming sessions and shared tools to address challenges systematically and inclusively.
- **User-Centric Solutions**: Balancing user needs with technical feasibility has prepared me to deliver practical, goal-oriented solutions in professional settings.

Roja Bugade

What I Did

I played an integral role in the development and implementation of COMPASS, a chatbot tailored for international students pursuing higher education.

Data Management:

- I oversaw the organization of university data, ensuring comprehensive coverage of essential information such as program details, admission criteria, living costs, and job opportunities.
- Structured and stored this information in a format optimized for retrieval within the chatbot system.

Chatbot Enhancement:

- Refined chatbot prompts to improve the accuracy and relevance of recommendations, utilizing Retrieval-Augmented Generation (RAG) and memory-based functionality.
- Collaborated on building the foundational chatbot framework using Streamlit, incorporating advanced features for personalization.

User Experience Design:

- Designed and optimized the web interface, focusing on a seamless and user-friendly experience through iterative usability testing and enhancements.

Deployment and Troubleshooting:

- Resolved deployment issues related to ChromaDB and external API integrations, ensuring a robust and operational chatbot system.

Challenges Faced and How They Were Overcome

Data Integration:

- Integrating diverse datasets like university data, job market trends, and weather information was challenging. By utilizing tools such as the OpenWeather API and ChromaDB, and breaking tasks into smaller modules, we ensured smooth integration and real-time functionality.

Personalization:

- Implementing features like RAG and session-based memory required a deep understanding of database management and prompt engineering. These were addressed through collaborative brainstorming and leveraging robust frameworks.

Interface Design:

- Designing an intuitive user interface required constant user feedback and iterative testing. This process led to a cleaner, more accessible design that met user needs effectively.

Lessons Learned About Working in a Team

Effective Communication:

- Regular updates and open channels for feedback fostered a transparent and productive work environment.

Leveraging Strengths:

- Assigning tasks based on team members' expertise led to faster problem resolution and higher-quality outputs.

Flexibility and Adaptability:

- Adjusting strategies to incorporate user feedback and resolve technical issues taught me the value of being flexible and solution-oriented.

Mutual Respect:

- Recognizing and valuing each team member's contributions created a supportive and collaborative atmosphere.

Applying These Lessons in Professional Settings

Enhanced Teamwork:

- I aim to foster strong team dynamics by prioritizing open communication, role alignment, and shared accountability.

Dynamic Project Management:

- My ability to adapt to evolving challenges ensures I can manage complex projects with efficiency and focus.

User-First Approach:

- By balancing user needs with technical feasibility, I am prepared to develop impactful, user-centric solutions that achieve organizational goals.