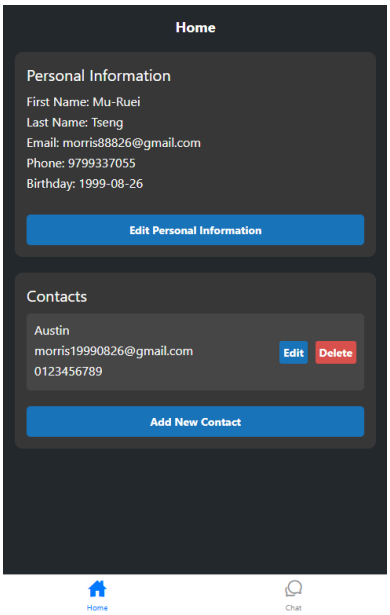
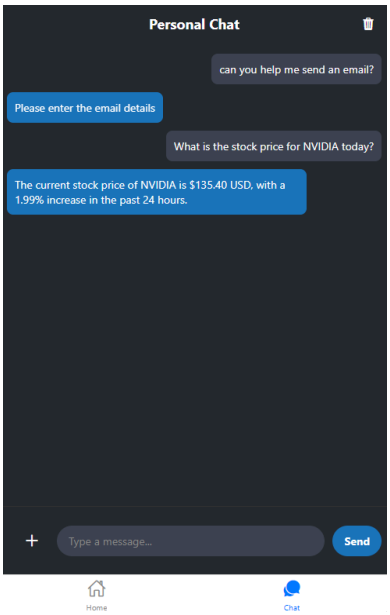
CSCE 689: Special Topics in Programming Large Language Models (LLMs)
Homework 3
Student: Mu-Ruei Tseng
UIN: 133007868

In this assignment, I built a personal AI that can support several functions including:

- Write and send emails on my behalf with Gmail
- Read multiple PDF files and answer questions
- Schedule meetings with Google Calendar
- Search the Internet
- Ask you questions (involving private data)

I developed a React Native frontend to enhance the user experience, focusing on an intuitive and user-friendly interface. The design of the interface is structured as follows:





The home page serves as a secure space for storing private data, such as personal and contact information. Users can easily add or edit their information from this section. The second page features a personal AI chat interface that assists users with various tasks, from handling personal queries to executing specific actions. For setting up the project, please visit here.

To handle both private and public information, I used two LLMs: a public LLM (OpenAI API - gpt-4o-mini) for processing non-confidential data, and a local LLM (Ollama - llama3.2) for searches involving private data (such as personal information, contact lists, and uploaded documents).

One challenging task when designing the Personal AI is to distinguish which tasks it should perform based on user input. For instance, the message "Hello" should not trigger any task (i.e. belongs to NORMAL_CHAT), whereas messages like "How many people are in the contact list?" or "What is the person's GPA?" should trigger the SEARCH_PRIVATE_DATA and READ_PDF tasks, respectively. To identify the potential task requested by the user, I used the global LLM to analyze the user prompt and

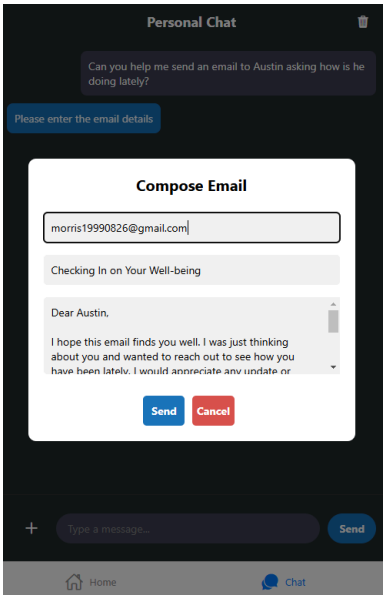
determine the most likely task. However, one downside of this approach is that the LLM can easily become confused when distinguishing between the READ_PDF, SEARCH_INTERNET, and SEARCH_PRIVATE_DATA tasks, or simply engaging in NORMAL_CHAT, compared to the more straightforward SEND_EMAIL and SCHEDULE_MEETING prompts.

To address this challenge, I experimented with various prompts and refined them to include more detailed instructions on how to differentiate between tasks. This approach was designed to guide the LLM more effectively. The resulting prompt structure is as follows:

```
"You are an intelligent assistant tasked with classifying user input into one of the following categories: {tasks}.\n"
"Given the user input: '{user_input}', classify the task based on the context.\n"
"Use 'SEARCH_INTERNET' only if the input requires up-to-date or detailed information from external sources.\n"
"Use 'SEARCH_PRIVATE_DATA' if the user asks for specific private data.\n"
"Use 'READ_PDF' if the user asks you to read or extract information from a PDF.\n"
"Use 'NORMAL_CHAT' for simple conversational replies or common knowledge.\n"
"Return in the format:\n"
"Task: <task_type>\n"
"Reason: <reason>"
```

One important point to note is that I currently use a global LLM for classifying user input, as this component does not store any chat history that might contain private data. A potential future improvement would be to incorporate chat history to help the LLM better differentiate between tasks. At that stage, transitioning to a local LLM for task classification would be advisable to ensure data privacy and security. I also experimented with several prompts and evaluated the accuracy of the current approach. See Appendix A.

In the following sections, I will explain how I process the tasks.



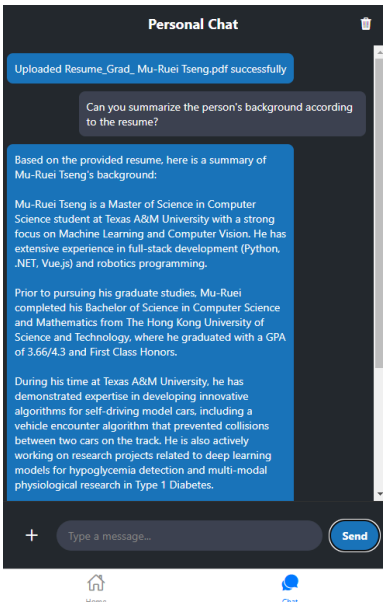
1. Write and send emails on my behalf with Gmail

This task is triggered when a prompt related to sending emails is detected. The local LLM first analyzes whether the contact information or email subject is present in the chat, allowing it to pre-fill the EmailModal, which manages the email-sending function. If the required information is not provided, the corresponding fields remain blank for the user to fill in with additional details. For instance, in the example on the right, I specified only that I wanted to send an email to Austin asking how he has been doing lately. The user input is sent to the local LLM for drafting the email if there is sufficient information; otherwise, the fields remain empty (see Appendix B). Once the user is satisfied with the content, they can simply click "send" to deliver the email.

2. Read multiple PDF files and answer questions

For this function, I designed a plus button located at the bottom left corner, allowing users to upload PDF files. The selected PDF files are transferred to a private data folder on the backend. Once the upload is complete, users can ask questions related to the content of the PDFs. For instance, after uploading a resume, users can type queries such as 'Can you summarize the person's background?' or 'What is the person's GPA?' The Personal AI will respond with the correct information extracted from the uploaded PDF. One issue with the current approach is that it can sometimes be mistaken for a

SEARCH_PRIVATE_DATA or NORMAL_CHAT task without keywords such as 'document,' 'article,' 'resume,' 'file,' etc., which could be an area for future development.



3. Schedule meetings with Google Calendar

Similar to the email-sending functionality, if a user prompt is related to scheduling meetings, a meeting scheduling modal will appear. This modal allows users to input the necessary details to create a meeting event in their Google Calendar. See Appendix B for the example. At this stage, the app currently does not support automatically extracting information from the user prompt or verifying whether the time slot is available. These features will be introduced in a later version.

4. Search the Internet

To conduct internet searches, I utilize a global LLM in conjunction with GoogleSearchAPIWrapper and LangChain Agents to fetch relevant information. To prevent excessive web searches, I set the max_iterations parameter to 10, limiting the number of websites queried.

5. Ask questions regarding the private data

For private data, the system employs a local LLM to search within the private database and extract the required information. It first reads all relevant information from the query in each schema and includes it in the prompt, allowing the local LLM to effectively extract the necessary information. See the sample output in Appendix B.

Regarding security concerns and preventing private data leakage, I use the public LLM only for analyzing the user's prompt to determine the appropriate task to perform. Additionally, the public LLM is used to execute the SEARCH_INTERNET function. However, once the response is obtained, I utilize the local LLM to analyze the content in the cascading search, as the responses from other tasks may contain information related to private data. By maintaining a clear separation of task distribution between the global and local LLMs, private data leakage can be effectively prevented.

Future Work

1. Add Chat History: Integrate chat history to provide better context for interactions, enabling the AI to classify tasks more accurately and respond more effectively.
2. Enhance Task Classification: Improve the task classification system to better interpret user input, ensuring the correct task is identified and executed.
3. Improve Auto-Fill Capabilities: Develop the ability to automatically extract relevant information from user prompts to pre-fill forms for scheduling meetings.
4. Expand App Functionality: Continue adding new features to increase the app's functionality and provide users with a broader range of tools.

Appendix A:

Performance regarding the task classification using a global LLM (gpt-4o-mini).

```
User Input: How many people are in the contact list?
Predicted Task: NORMAL_CHAT
Expected Task: SEARCH_PRIVATE_DATA
==================================
User Input: How many contacts are in my data?
Predicted Task: SEARCH_PRIVATE_DATA
Expected Task: SEARCH_PRIVATE_DATA
==================================
User Input: What is the email address of Austin?
Predicted Task: SEARCH_PRIVATE_DATA
Expected Task: SEARCH_PRIVATE_DATA
==================================
User Input: Can you find information on recent AI trends?
Predicted Task: SEARCH_INTERNET
Expected Task: SEARCH_INTERNET
==================================
User Input: Can you help me send an email to John?
Predicted Task: SEND_EMAIL
Expected Task: SEND_EMAIL
==================================
User Input: Can you schedule a meeting for tomorrow?
Predicted Task: SCHEDULE_MEETING
Expected Task: SCHEDULE_MEETING
==================================
User Input: What is the capital of France?
Predicted Task: SEARCH_INTERNET
Expected Task: NORMAL_CHAT, SEARCH_INTERNET
==================================
User Input: According to the document, what is the deadline for the project?
Predicted Task: READ_PDF
Expected Task: READ_PDF
```

```
==================================
User Input: Can you give me a summary uploaded file?
Predicted Task: READ_PDF
Expected Task: READ_PDF
==================================
User Input: What is the stock price of Tesla?
Predicted Task: SEARCH_INTERNET
Expected Task: SEARCH_INTERNET
==================================
User Input: Hello, how are you?
Predicted Task: NORMAL_CHAT
Expected Task: NORMAL_CHAT
==================================
User Input: What is the weather forecast for tomorrow?
Predicted Task: SEARCH_INTERNET
Expected Task: SEARCH_INTERNET
==================================
User Input: Can you help me find the phone number of John?
Predicted Task: SEARCH_PRIVATE_DATA
Expected Task: SEARCH_PRIVATE_DATA
==================================
User Input: I need to schedule a meeting with the team.
Predicted Task: SCHEDULE_MEETING
Expected Task: SCHEDULE_MEETING
==================================
User Input: According to the manual, what are the safety precautions?
Predicted Task: READ_PDF
Expected Task: READ_PDF
==================================
Accuracy: 14/15
```

Appendix B:

More examples regarding the Personal AI app.