FITQUEST Coding Conceptualization

By: Varsan, Nabil, Farhaan, Shaurya, and Zayed

An additional feature that we would create is a chatbot powered by Artificial Intelligence. This data would collect information from GPT-4 to be able to create a conversation with a user. The objective of this feature is to be able to allow users to have an actual discussion with AI on how they could meet their objectives. This could include tips to create a schedule, plan it out, and motivate oneself to achieve their goal. It could also induce other fitness related discussions relevant to the overall theme of the application.

This feature would be a PRO feature, which would motivate people to pay for a professional membership. This would be accessed through the dashboard page, where clicking a small logo would open the chatbot. Initially, a message would be sent at the start of each conversation introducing the model and explaining its potential capabilities. It would also provide a guide on how to use it, such as asking specific questions. The user would type in their request in a textbox, which is constrained to 256 characters. Then, using a send button they can push their request to the AI, and the response will be presented in a text box. This text box would need to change size based on the size of the text. This can be programmed by using the number of characters and the output and converting that into a two dimensional shape.

We know that the objective of the AI model would be to engage in a chat bot. Firstly, we would need to decide which AI model we would use. There are numerous ones available, ranging from GPT 4 to Gemini. Once we find the correct model, we would incorporate this into the back end of our program. This would need to read a user’s input and send it to that AI. Once the appropriate response has been received, it would display the output to the user. This data would directly be collected from the model, and presented within a text box. The code that would be used is the connection to the AI model, and how it is integrated. Using our program, we would need to load this model within our application and continuously run it. In addition, the AI model must specifically work on the application. It should not be designed for any conversation. Such, there should be limits and conditions placed into the conversation that constrain the conversations. The additional programs which are incorporated should be a separate segment of the code which is separate from other app features. All the importing and the uploads needed should be placed together, because this is more logical in terms of code organization.

Further elaborating on the architecture of the application, the conversation is initiated when the chatbot is opened through the dashboard page. From this point on, the primary program running is the chatbot. This dashboard is easily accessible and is more general in terms of what it serves. It links to all of the features, and the chatbot is an example of a feature. Thus, the location of the chatbot would be very easy to find and comprehendible, enabling our application to be understood. To terminate the chatbot, this can occur through two methods. The first option is to close the chatbot itself using a red X button at the top left of the screen. This should be large enough so that it is easy to find. Having a termination option like this would be a very simple killswitch that a user can use. Another method to end the program is directly within the conversation. If the user activates a break by typing in an input like “Thats it” or “goodbye”, the program would send a confirmation whether the user wishes to end a conversation. For this sort of input, there would be a button (Yes or No), and if the user confirms that it will also close the chatbot. This is a logical method to end the program, since some people like to have flow in conversations. This also makes using the program easier and ensures that a close program request is not misunderstood as something else.