Betterzila Assignment solution

AI Engineer Assignment: Development of a Fitness Chatbot

Project Report:

Building a Fitness Chatbot with Flask, OpenAI, HTML, and CSS

Introduction

This project involves creating a chatbot using Flask, a Python web framework, OpenAI's GPT-3.5-turbo model for generating responses, and HTML/CSS for the graphical user interface (GUI). The chatbot allows users to interact by sending messages and receiving responses in a conversational manner.

Technologies Used

Flask: A micro web framework for Python used to create web applications.

OpenAI API: Provides access to natural language processing models, including GPT-3.5-turbo for generating text-based responses.

HTML: Markup language used for creating the structure of the web pages.

CSS: Styling language used for enhancing the visual presentation of the web pages.

Project Setup

Flask Setup: Created a Flask web application with routes for the homepage (`/`) and API endpoint (`/api`).

OpenAI API Setup: Configured the OpenAI API with the required API key.

HTML/CSS Setup: Designed a simple and intuitive chat interface using HTML for structure and CSS for styling.

Approach

Homepage (`/`): The homepage contains a form element for users to input messages and a submit button.

API Endpoint (`/api`):

Receives POST requests containing a JSON payload with a `message` field.

Uses the `openai.ChatCompletion.create` method to send the user message to the OpenAI API and receive a response.

Returns the generated response or a failure message if the API request fails.

HTML/CSS for GUI:

Designed a chat interface using HTML for the structure, including chat bubbles for messages.

Used CSS to style the chat interface, including colors, fonts, and layout.

Challenges Faced

Integration of OpenAI API: Ensuring the API key is properly configured and handling API responses appropriately.

HTML/CSS Styling: Designing an attractive and user-friendly chat interface using HTML and CSS.

User Input Validation: Validating user input to prevent malicious or invalid inputs.

Future Improvements

User Authentication: Implement user authentication to restrict access to the chatbot.

Improved UI/UX: Enhance the user interface to make the chatbot more visually appealing and intuitive.

Error Handling: Implement better error handling to provide more informative messages to users.

Conclusion

This project demonstrates how to build a chatbot with Flask, OpenAI, HTML, and CSS. By combining these technologies, we created a chatbot that can interact with users in a conversational manner, providing a seamless user experience.

How to Run Project :

1. **Install Flask and OpenAI**: If you haven't already, install Flask and the OpenAI Python package. You can do this using pip: `pip install Flask openai`
2. **Set Up Your OpenAI API Key**: Replace **'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'** with your actual OpenAI API key in your Flask application.
3. **Create Your Flask App**: Create a new Python file (e.g., **app.py**) and paste the Flask application code into it.
4. **Create Your HTML/CSS Files**: Create an **index.html** file for the chatbot interface and a **style.css** file for styling. You can use the provided HTML/CSS code or customize it as needed.
5. **Run Your Flask App**: In your terminal, navigate to the directory where your Python file is located and run the Flask application: bashCopy code: `python app.py`
6. **Access Your Chatbot**: Open a web browser and navigate to **http://127.0.0.1:5000/** to access your chatbot. You should see the chat interface where you can input messages and receive responses.
7. **Interact with Your Chatbot**: Start typing messages in the chat interface and press Enter to send them. Your chatbot should generate responses using the OpenAI API and display them in the chat interface.
8. **Stop Your Flask App**: To stop the Flask application, press **Ctrl + C** in your terminal.