Software Engineer Home Assignment

Real-time GPT-4o-mini CLI with Function Calling

Objective

The goal of this assignment is to develop a Command Line Interface (CLI) tool in Go that interacts with OpenAI's GPT-4o-**realtime**-mini, with a websocket, in real time.
The CLI should allow users to send messages and receive responses in a streaming format, simulating the experience of a real-time conversation. Additionally, the CLI should support function calling, implementing a simple function that multiplies two numbers.

Requirements

1. CLI Interaction with GPT-4o-mini

- Develop a Go-based CLI tool that communicates with GPT-4o-realtime-mini in real time.
- Ask Roey/Ofir for OpenAI key
- Users should be able to input messages, and the chatbot should respond in chunks, similar to how responses appear in OpenAI's ChatGPT interface.
- Ensure the CLI remains interactive, allowing users to continue the conversation without restarting the program.

2. Streaming Responses

- Implement streaming response functionality, ensuring the output is displayed incrementally instead of waiting for the entire response to be generated.
- Maintain smooth and natural output flow for an improved user experience.

3. Function Calling Support

- Extend the CLI to support OpenAI function calling.
- Implement a function that multiplies two numbers and returns the result.
- Ensure that when the model detects a request for multiplication, it correctly calls the function and returns the computed result.

Technical Guidelines [Important]

- Use Go as the primary programming language.
- Utilize OpenAI's API to interact with GPT-4o-realtime-mini.
- Implement streaming using Go's concurrency features (e.g., goroutines, channels).
- Structure the code with maintainability and extensibility in mind.
- Provide clear error handling and logging where appropriate.
- Ensure proper documentation in the README file.

Deliverables

- A GitHub repository (or ZIP file) containing the source code.
- A README file with:
- Instructions on how to install and run the CLI.
- API key configuration details.
- Examples of usage.
- A brief write-up (or inline comments) explaining the architecture and design choices.

Evaluation Criteria

- Functionality – The CLI should work as expected, providing real-time responses and correctly handling function calls.
- Code Quality – Clean, modular, and well-documented code following best practices.
- Efficiency – Proper use of Go concurrency for streaming responses.
- Error Handling – Graceful handling of API failures, invalid inputs, and other edge cases.

If you have any questions, feel free to reach out before the submission deadline.