Assignment: Realtime AI Backend (WebSockets + Supabase)

Goal

Build a high-performance, asynchronous Python backend that simulates a real-time conversational session. The focus is on implementing core backend patterns: **real-time bi-directional communication (WebSockets)**, **LLM interaction/streaming**, **asynchronous data persistence (Supabase)**, and **post-session automation**.

Core Requirements

1. Realtime Session & Streaming

- **Framework:** Use an asynchronous Python framework (FastAPI or Quart:preferred).
- **Endpoint:** Implement a WebSocket endpoint, e.g., /ws/session/{session_id}.
- **Prompt Handling:** Implement logic to receive user input and stream the LLM's response tokens back to the client immediately, simulating a low-latency conversation.

2. Complex LLM Interaction

Applicants must demonstrate an ability to handle an interaction beyond simple Q&A. This could be achieved through any complex interaction pattern, such as:

- **Function/Tool Calling (Recommended):** Use the LLM's function-calling capability to execute a simulated internal tool (e.g., fetching data) and feed the result back into the LLM conversation flow.
- **Multi-Step Routing:** Implement conditional logic that changes the LLM's system prompt or workflow based on the user's initial query or context.
- **State Management:** Demonstrate managing conversation state across multiple user turns.

3. Persistence (Supabase)

All application data must be saved to the Supabase Postgres database).

Data Requirement: Design and implement a minimal schema that captures two types of information:

- **Session Metadata:** A primary table to store high-level session details (session_id, user_id, start_time, end_time).
- **Detailed Event Log:** A secondary table to store a granular, chronological log of events during the session (e.g., user messages, AI responses, interaction triggers).

4. Post-Session Processing

Upon the client disconnecting (WebSocket close):

- **Automation:** Trigger an asynchronous task to run a processing job.
- **Outcome:** The job must use the LLM to analyze the stored conversation history (from the event log) and generate a concise **Session Summary**.
- **Finalization:** Persist the final summary, end time, and duration back to the main session record.

Deliverables

1. **GitHub Repository:** A single repository containing all project files.
2. **README.md:** A comprehensive documentation file including:
    - Detailed setup steps and required dependencies.
    - The complete Supabase database schema (SQL commands).
    - Instructions on how to run and test the WebSocket server.
    - A short explanation of key design choices.
3. **Source Code:** All necessary Python files to run the application

    If possible, Simple Frontend preferred as this is not an assessment focused on UI/UX [i.e click a button and get started/ connected to the websocket]