Advanced Real-Time Sales Analytics System

Objective:

Build a system to manage and analyze sales data in real-time. The task will involve leveraging AI systems like OpenAI's ChatGPT or Gemini for assistance and generating recommendations, while other parts of the system must be written manually. The project must include a real-time reporting feature and integration with external APIs.

Task Details:

Backend Requirements:

1. **APIs to Manage Orders and Analytics:**
    - **POST /orders:** Add a new order to the database with fields: product ID, quantity, price, and date.
    - **GET /analytics:** Return real-time sales insights, including:
        - Total revenue.
        - Top products by sales.
        - Revenue changes in the last 1 minute.
        - Count of orders in the last 1 minute.
2. **Real-Time Reporting:**
    - Implement WebSocket support for the frontend to subscribe to updates for:
        - New orders.
        - Updated analytics.
    - Publish updates in real-time when new orders are added.
3. **AI Integration (via API like ChatGPT/Gemini):**
    - Provide an endpoint:
        - **GET /recommendations:**
            - Sends recent sales data to the AI system.
            - Receives product promotion suggestions or strategic actions.
    - Example AI prompt:
        - "Given this sales data, which products should we promote for higher revenue?"
4. **External API Integration:**
    - Integrate with a weather API (e.g., OpenWeather) to adjust recommendations:
        - Promote cold drinks on hot days or hot drinks on cold days.
    - Add logic to suggest dynamic pricing based on weather or seasonality.
5. **Manual Implementation Requirements:**
    - All database queries, real-time logic, and API integrations must be written manually (no ORM or prebuilt frameworks).
    - Use a lightweight database like SQLite for simplicity.

Evaluation Criteria:

1. Code Quality and Structure (30 points):

- Clean, organized, and readable code.
- Proper separation of concerns (e.g., frontend, backend, data handling, real-time updates).

2. Real-Time Functionality (25 points):

- Successful implementation of real-time reporting using WebSockets.
- Updates are accurate and reflect live data changes.

3. AI Integration (20 points):

- Correct integration with ChatGPT/Gemini to generate actionable recommendations.
- Appropriate and logical prompts for AI.
- Effective use of AI-generated insights in the frontend.

4. External API Integration (15 points):

- Successful integration with a weather API or similar service.
- Accurate and relevant use of external data in recommendations.

5. Documentation and Testing (10 points):

- Clear documentation (README) explaining:
    - Which parts were assisted by AI.
    - Manual implementation details.
    - How to run and test the project.
- At least one test case for APIs or real-time functionality.

AI-Assisted vs. Manual Implementation:

Backend (30% AI, 70% Manual):

- **AI-Assisted:**
    - Generate API prompts for ChatGPT/Gemini to suggest product promotions.
    - Structure API endpoints and generate boilerplate code.
- **Manual:**
    - Write all database logic, WebSocket publishing, and external API integration.
    - Ensure real-time updates and error handling are implemented.

Project Timeline:

- **Duration:** 2 days.

- **Day 1:** Set up the backend structure, create basic APIs, and implement WebSocket functionality.
- **Day 2:** Build the frontend, integrate AI and external APIs, and test the real-time updates.

Key Notes for Candidates:

1. Clearly document which parts of the code were generated or assisted by AI.
2. Avoid using prebuilt libraries for tasks that can be implemented manually (e.g., charting, data fetching).
3. Focus on clean, modular, and scalable code.
4. Prioritize real-time functionality and smooth integration between frontend and backend.