Real-Time Analytics Dashboard with Custom API Integration

Objective

Develop a real-time analytics dashboard for a fictional e-commerce platform. The dashboard should display various metrics such as active users, total sales, and top-selling products, updated in real-time. You will need to create a custom API to fetch this data and integrate it with the frontend.

Requirements

1. Backend (API)
    - Create a RESTful API using any backend framework (e.g., C# .NET).
    - Implement endpoints to fetch:
        - Active users (simulated data with random values)
        - Total sales (simulated data with random values)
        - Top-selling products (list of products with sales figures)
    - Use WebSockets to push updates to the frontend in real-time.
    - Include necessary unit tests for the API endpoints.
2. Frontend
    - Create a single-page application (SPA) using a modern JavaScript framework/library (e.g., React).
    - Display the following metrics on the dashboard:
        - Active users (updated every 10 seconds)
        - Total sales (updated every 10 seconds)
        - Top-selling products (updated every 30 seconds)
    - Implement real-time updates using WebSockets.
    - Use a charting library (e.g., Chart.js, D3.js) to visualize the data.
    - Include responsive design to ensure the dashboard is mobile-friendly.
3. Data Simulation
    - Implement a simple data simulation mechanism in the backend to generate random data for the metrics.
    - Ensure the data changes over time to simulate a real-world scenario.
4. Documentation
    - Provide clear documentation on how to set up and run the project.
    - Include a brief explanation of the architecture and design choices.

Evaluation Criteria

- Code Quality: Clean, readable, and well-documented code.
- Functionality: All required features are implemented and working correctly.
- Design: User-friendly and responsive UI.

- Real-Time Capabilities: Effective use of WebSockets for real-time updates.
- Testing: Comprehensive unit tests for backend endpoints.
- Documentation: Clear and concise setup instructions and project overview.

Submission Guidelines

- Provide all code in a zip file (e.g., code, build scripts, SQL scripts)
- Include a README file with setup instructions and any other relevant information.
- Ensure all dependencies are listed in `package.json` (Node.js) or equivalent for other frameworks.