Backend Test with Node.js, MongoDB, and PostgreSQL

Description

The task involves creating a backend system that allows users to select cinema seats concurrently while handling conflicts when multiple users attempt to book the same seat. The frontend is not evaluated and can be minimalistic, but the backend solution should be robust.

Tech Stack

- **Node.js**
- **MongoDB**
- **PostgreSQL**

Technical Specifications

1. **RESTful API Endpoints**:
   - **GET /sessions**: Returns a list of cinema sessions, including halls and showtimes.
   - **GET /sessions/:id/seats**: Shows the current status of seats for a specific session.
   - **POST /sessions/:id/seats/reserve**: Allows users to reserve specific seats.

2. **Concurrency and Consistency**:
   - **Optimistic Locking**: Implement an optimistic lock mechanism to allow users to attempt a seat reservation.

3. **Data Models**:
   - **User Model**: Stores user details and reservation history.
   - **Reservation Model**: Tracks seat reservations and associated sessions.
   - **Session Model**: Contains session data, hall ID, seats, and timing.

  - **Hall Model**: Stores cinema hall information, including seat configurations.

  - **Seat Model**: Defines seat layout and availability for each hall.

Key Performance Indicators (KPIs)

Define and export KPIs for your solution. These metrics should help us observe and assess system performance under concurrent use.

Additional Recommendations

1. **Documentation**: Include a README explaining backend setup, concurrency logic, and design choices.
2. **Simple Frontend**: Implement a basic frontend (React or HTML with Fetch API) to simulate seat selection and booking.
3. **Concurrency Libraries**: You may consider using concurrency libraries in Node.js or native transactions in databases for conflict resolution.

Evaluation Criteria

1. **Robustness**: The backend should handle multiple concurrent requests and ensure data consistency.
2. **Efficiency**: Efficient concurrency handling and lock mechanisms in critical endpoints.
3. **API Design**: Clear and well-structured endpoints.
4. **Error Handling**: Informative error handling for booking conflicts.

Please request clarification for any points you need. You have 7 days to complete and submit the test from the date of receipt.