Event Ticket Booking Application - Backend Overview for Frontend Developer

This document provides a simple overview of the Event Ticket Booking Application backend to help a React frontend developer integrate with it. The backend manages user authentication, event creation, event browsing, and ticket purchasing, and is ready for you to connect with a React frontend.

The goal is to make it easy for you to understand what the backend does, how to call its APIs, and how to build a user-friendly interface for Normal Users (who buy tickets) and Organizers (who create events).

Table of Contents

What the Backend Does

The backend powers an Event Ticket Booking Application where:

- Normal Users can:
    - Sign up, log in, and browse events.
    - Filter events by type, date, location, or price.
    - Buy tickets for events (specifying how many tickets they want).
- Organizers can:
    - Sign up, log in, and create events (e.g., concerts, conferences).
    - Update or delete their events.

    - View all their events.

The backend uses MySQL to store data and JWT tokens for secure user authentication. All APIs are served at http://localhost:3000/api.

Key Concepts

- Users and Roles:
    - Normal Users: Can browse and buy tickets.
    - Organizers: Can create and manage events.
    - Users are identified by a JWT token after login, which must be sent in API requests as Authorization: Bearer <token>.
- Events:
    - Events have details like title, type (e.g., Concert), date, location, price, and availability (number of tickets left).
    - Stored in a MySQL database and managed via APIs.
- Tickets:
    - When a Normal User buys tickets, the backend records each ticket and reduces the event's availability.
    - Uses transactions to ensure no overselling (e.g., if 10 tickets are left, you can't buy 11).
- Database:
    - Tables:
        - users: Stores user info (email, password, role).
        - events: Stores event details.
        - tickets: Stores purchased tickets.
    - You don't need to interact with the database directly; just use the APIs.

APIs You'll Use

Here’s a simple list of the backend APIs. Some need a JWT token in the header.

1. Sign Up & Log In

- Sign Up: POST /api/auth/signup
    - Send: { "email": "user@example.com", "password": "pass123", "role": "normal" } (or "organizer")
    - Get: { "token": "<jwt_token>", "message": "User created" }
    - Use: Create a new user.
- Log In: POST /api/auth/login
    - Send: { "email": "user@example.com", "password": "pass123" }
    - Get: { "token": "<jwt_token>", "message": "Login successful" }
    - Use: Log in to get a token.

2. Events (For Organizers)

- Create Event: POST /api/events/create
    - Needs Token: Yes (Organizer)
    - Send: { "title": "Rock Concert", "type": "Concert", "event_date": "2025-05-01 18:00:00", "location": "City Arena", "price": 50, "availability": 100 }
    - Get: { "message": "Event created", "eventId": 1 }
- Update Event: PUT /api/events/:eventId (e.g., /api/events/1)
    - Needs Token: Yes (Organizer)
    - Send: Same as create.
    - Get: { "message": "Event updated" }
- Delete Event: DELETE /api/events/:eventId
    - Needs Token: Yes (Organizer)
    - Get: { "message": "Event deleted" }
- Get My Events: GET /api/events/my-events

- Needs Token: Yes (Organizer)
- Get: List of events like [{ "id": 1, "title": "Rock Concert", ... }]

3. Browse Events (For Everyone)

- Get Events: GET /api/events
  - Needs Token: No
  - Optional Filters: ?type=Concert, ?date=2025-05-01, ?location=Arena, ?minPrice=20, ?maxPrice=100
  - Get: List of events like [{ "id": 1, "title": "Rock Concert", "type": "Concert", "event_date": "2025-05-01T18:00:00.000Z", "location": "City Arena", "price": 50, "availability": 100, ... }, ...]
  - Use: Show events with filter options.

4. Buy Tickets (For Normal Users)

- Purchase Tickets: POST /api/events/purchase/:eventId (e.g., /api/events/purchase/1)
  - Needs Token: Yes (Normal User)
  - Send: { "quantity": 2 }
  - Get: { "message": "Tickets purchased successfully", "ticketCount": 2, "totalCost": 100, "eventId": 1 }

Errors:

- 400: Bad input (e.g., invalid quantity).
- 401: No token or invalid token.
- 403: Wrong user role.
- 404: Event not found.
- 500: Server issue.

Integrating with FrontFuckingEnd

Hera sathi yesari hanxau integrate

- **Set Up Your React App:**
    - **Create a new React app: npx create-react-app ticket-booking-frontend.**
    - **Install axios for API calls and react-router-dom for navigation: npm install axios react-router-dom.**
    - **Add the backend URL in a .env file:**

env

REACT_APP_API_URL=http://localhost:3000/api

- **Handle Login/Signup:**
    - **Build login and signup forms to call /api/auth/login and /api/auth/signup.**
    - **Save the JWT token in localStorage after login/signup.**
    - **Send the token in the Authorization header for protected APIs (use axios.defaults.headers.common['Authorization'] = Bearer ${token}).**
    - **Redirect users to a dashboard after login (different dashboards for Normal Users and Organizers).**
- **Show Events:**
    - **Call GET /api/events to fetch events and display them in a list or grid.**
    - **Add filter inputs (e.g., type, date, location, price range) and update the API call with query params (e.g., /api/events?type=Concert&minPrice=20).**
    - **Show event details like title, date, price, and available tickets.**
    - **Add a "Buy Tickets" button linking to a purchase page (e.g., /events/purchase/1).**
- **Let Organizers Manage Events:**

    - **Build a form for Organizers to create/edit events, calling POST /api/events/create or PUT /api/events/:eventId.**
    - **Show a list of their events (GET /api/events/my-events) with "Edit" and "Delete" buttons.**
    - **Call DELETE /api/events/:eventId to remove an event.**
- **Handle Ticket Purchases:**
    - **Create a purchase page where Normal Users pick how many tickets they want (e.g., a number input).**
    - **Call POST /api/events/purchase/:eventId with { quantity: 2 }.**
    - **Show a success message with the number of tickets and total cost (e.g., "Bought 2 tickets for $100!").**
    - **Redirect to the dashboard or event list.**
- **Navigation:**
    - **Use react-router-dom to set up routes:**
        - **/: Show event list.**
        - **/login: Login page.**
        - **/signup: Signup page.**
        - **/dashboard: User or Organizer dashboard.**
        - **/events/purchase/:eventId: Ticket purchase page.**
        - **/events/create: Event creation page (Organizers only).**

Tips to Make It Awesome

- **Look Good: Use a UI library like Material-UI or Bootstrap to style your app (e.g., nice buttons, cards for events).**
- **Show Feedback: Add loading spinners while fetching data and show error messages (e.g., "Not enough tickets!") if APIs fail.**
- **Protect Pages: Redirect to /login if a user tries to access a protected page (e.g., ticket purchase) without a token.**

- **Test First: Use Postman to test APIs before coding:**
    - **Sign up/log in to get a token.**
    - **Try creating events (Organizer token) and buying tickets (Normal User token).**
- **Keep It Simple: Start with login/signup, then event listing, then ticket buying, and finally Organizer features.**