Sports Booking

Mohit Somani
(iit2021105@iiita.ac.in)

Introduction

A backend system called the Sports Booking Application was created to make it easier to reserve sports facilities at various locations. Normal users and centre managers are the two main user roles served by this powerful platform, each of which has unique features catered to their requirements.

Among the application's primary features are:

- Multi-center assistance for a range of courts and sports
- Access control for managers and users based on roles
- Users can create, view, and cancel reservations.
- All-inclusive booking administration for centre managers
- Preventing double booking to guarantee court availability
- This system seeks to improve the booking experience for sports facilities while offering effective
  management resources for centre managers.

Design Decisions

1. Role Based Access

Control The application differentiates between two types of users: customers and managers. Customers can only interact with their own bookings, while managers can view and manage all bookings within their center.

2. Database Design: MongoDB:

Chosen for its flexibility in handling complex relationships between centers, courts, sports, and users. It is highly scalable compared to other SQL counterparts , it is being preferred for horizontal scaling .
.

Normalization:

This database schema follows Third Normal Form (3NF), a solid standard for relational databases. Here's a breakdown:

- **First Normal Form (1NF):**
  The schema meets 1NF since each field contains atomic, indivisible values (e.g., name, role, slot_time). No field holds multiple values.
- **Second Normal Form (2NF):**
  In the Bookings table, all non-primary key fields (like centre, court, sport) rely on the entire primary key (id), with no partial dependencies, ensuring it conforms to 2NF.
- **Third Normal Form (3NF):**
  There are no transitive dependencies. Each non-key attribute is entirely dependent on the primary key and not on other non-key fields. For example, in the User table, email and password depend only on the primary key (id), satisfying 3NF.

Advantages and Key Design Decisions:

1. **Role-Based Design (User Table)**:
   The inclusion of a "role" field in the User table, which differentiates between customers and managers, is an effective choice for managing role-based access control, simplifying user permissions management.
2. **Flexible Booking System (Bookings Table)**:
   The Bookings table is designed to reference entities like centre, sport, and court, creating adaptable relationships. This makes it easier to scale the booking system across multiple centers and sports. Additionally, separating slot_time and slot_date into distinct fields enables easy querying and handling of schedules.
3. **Modular Structure (Court, Centre, Sport Tables)**:
   By organizing courts, centers, and sports into separate tables, the system efficiently manages relationships between different sports facilities. This modularity improves the ease of maintenance and scalability, allowing for the addition of new centers or sports without modifying the existing database.
4. **Referential Integrity**:
   Foreign key relationships between tables (e.g., sport_id, centre_id, user_id) ensure referential integrity, preventing orphaned records and making sure every booking is correctly linked to a valid user, court, and sport.
5. **Enum for User Roles**:
   Implementing an enum for the "role" field in the User table ensures that only specific roles (such as "manager" or "customer") are allowed, reducing the likelihood of input errors and maintaining data consistency.
6. **Scalability**:
   The schema is highly scalable, allowing new features to be incorporated without disrupting core functionality. For example, a review or feedback system could be added by referencing existing tables like Bookings or Users with minimal structural changes.

3. Authentication:

JWT-based authentication is used to protect routes, ensuring that only authorized users can access the booking system. The middleware validates the token before allowing users to access the requested resources, enhancing the security of the application.

4. Implementation Details

Backend:

* The backend is built using Node.js and Express.js, which are commonly used for building fast and scalable web applications.
* Express.js handles routing and HTTP requests, allowing users to interact with different endpoints (such as booking, user management, etc.).
* Mongoose, an ODM (Object Data Modeling) library, is used to interact with the MongoDB database. It provides schema-based validation and simplifies queries, making it easier to work with MongoDB's flexible, non-relational structure.
* Controllers: Each module (user, booking, center, and sport) is managed by its own controller. This modular approach helps keep the code organized and maintainable. Each controller handles the logic for its specific part of the application, like creating, updating, and retrieving data from the database

```
backend_booking_app/
├── controllers/
│   ├── addCourtAndUpdateCentre.js
│   ├── addDummyCentre.js
│   ├── addDummyUsers.js
│   ├── bookingController.js
│   ├── centreController.js
│   ├── sportController.js
│   └── userController.js
├── db/
│   └── connection.js
├── middlewares/
│   └── authMiddleware.js
├── models/
│   ├── Booking.js
│   ├── Centre.js
│   ├── Court.js
│   ├── Sport.js
│   └── User.js
├── routes/
│   ├── bookingRoutes.js
│   ├── centreRoutes.js
│   ├── protectedRoutes.js
│   ├── sportRoutes.js
│   └── userRoutes.js
├── index.js
├── package-lock.json
└── package.json
```

JWT Authentication:

- JWT (JSON Web Token) is used for authenticating users. When a user logs in, they receive a JWT token, which is then used to access protected routes. This ensures that only authenticated users can perform certain actions (e.g., making a booking or canceling one).

- The token contains encoded user information and is passed in the Authorization header of each request.
- In every request to a protected route, the token is verified using the server's secret key. If the token is valid, the user is allowed to proceed; otherwise, access is denied

Challenges and Solutions

- **Double Booking Prevention:**

  Challenge: Preventing multiple users from booking the same court for the same date and time.
  Solution: A database query in the BookingController checks for existing bookings at the court, date, and time. If a conflict is found, the system blocks the new booking request.

- **Role-Based Access Control:**

  Challenge: Implementing different permissions for customers and managers.
  Solution: A database query in the BookingController checks for existing bookings at the court, date, and time. If a conflict is found, the system blocks the new booking request.