Sports Facility Booking System Backend Project Report

Introduction

The Sports Facility Booking System is designed to streamline the process of reserving sports facilities across various centers. This system aims to provide an efficient and user-friendly platform for managing bookings, sports facilities, and user accounts.

Design Decisions

Entities and Relationships

Booking

Attributes: - court (Number) - time (String) - date (Date)

Relationships: - user → User (One-to-One relationship, since a booking is made by one user) - location → Center (Many-to-One relationship, as multiple bookings can be made for one center) - sport → Sport (Many-to-One relationship, as multiple bookings can be made for the same sport)

Center

Attributes: - location (String)

Relationships: - sports → Sport (Many-to-Many relationship, as a center can host multiple sports and a sport can be available at multiple centers)

Sport

Attributes: - name (String) - courts (Number) - timeSlots (Array of Strings)

Relationships: - center → Center (Many-to-One relationship, as a sport belongs to one center)

Court

Attributes: - number (Number) - timeSlots (Array of Strings)

Relationships: - sport → Sport (Many-to-One relationship, as multiple courts are associated with a sport)

ER Diagram Structure

- **User:** Represents the users making the bookings.
    - Booking (Many-to-One) → User
- **Booking:** Holds details of each booking such as court, time, and date.
    - Booking (Many-to-One) → Center
    - Booking (Many-to-One) → Sport
    - Booking (Many-to-One) → User
- **Center:** Represents a sports center (location) that has multiple sports.
    - Center (One-to-Many) → Sport
    - Center (One-to-Many) → Booking
- **Sport:** Represents different sports available at a center, like Badminton or Tennis.
    - Sport (One-to-Many) → Court
    - Sport (Many-to-One) → Center
    - Sport (One-to-Many) → Booking
- **Court:** Represents the courts in a center that are associated with a sport.
    - Court (Many-to-One) → Sport

Database Schema

We chose MongoDB as our database, utilizing Mongoose for object modeling. The schema design includes: 1. User Model: Stores user information including username, PIN (hashed), email, and balance. 2. Center Model: Represents sports centers with location and available sports. 3. Sport Model: Defines sports available at each center, including courts and time slots. 4. Booking Model: Manages booking information, linking users, centers, sports, and specific time slots.

This design allows for flexible querying and efficient data retrieval.

API Structure

We implemented a RESTful API structure using Express.js. Key endpoints include: - User management (registration, authentication) - Booking creation and retrieval - Center and sport information retrieval and creation - Booking deletion (New feature)

Implementation Details

Technologies Used

- **Backend:** Node.js with Express.js
- **Database:** MongoDB with Mongoose ODM
- **Authentication:** PIN-based system using bcrypt for hashing
- **Additional libraries:** cors, body-parser, dotenv

Key Components

1. **Server Setup (server.js):**
    - Configures Express application
    - Sets up middleware (cors, body-parser)
    - Connects to MongoDB
    - Defines API routes
2. **Route Handlers:**
    - centers.js: Manages center-related operations
    - bookings.js: Handles booking creation, retrieval, and deletion
    - sports.js: Manages sports-related operations
    - users.js: Handles user registration and authentication
3. **Models:**
    - User.js: Defines user schema with username, hashed PIN, email, and balance
    - Center.js: Represents sports centers with location and sports
    - Sport.js: Defines sports with name, courts, time slots, and associated center
    - Booking.js: Manages booking information
4. **Authentication:**
    - Implemented a PIN-based authentication system
    - Uses bcrypt for secure PIN hashing and comparison

New Feature: Booking Deletion Endpoint

We added a new route to allow users to delete their bookings: - Endpoint: `/api/booking/:id` - Requires user authentication - Validates user's PIN before deleting the booking - Ensures users can only delete their own bookings

Challenges and Thier Solutions

1. **Booking Conflicts:** Implemented a time-slot checking algorithm in the booking creation process to prevent double bookings.
2. **Data Relationships:** Utilized Mongoose's reference system to create relationships between models, allowing for efficient data querying and population.
3. **PIN-based Authentication:** Chose a simple PIN system for ease of use, but implemented secure hashing with bcrypt to protect user credentials.
4. **Cross-Origin Resource Sharing (CORS):** Implemented CORS middleware to allow frontend applications to communicate with the API securely.
5. **Booking Deletion:** Added secure deletion mechanism with PIN verification to prevent unauthorized booking removals.

Future Improvements

1. **Advanced Authentication:** Implement JWT-based authentication for enhanced security and session management.

2. **Real-time Availability:** Integrate WebSocket technology for live updates on facility availability.
3. **Payment Integration:** Add a payment gateway to handle transactions for bookings.
4. **Admin Dashboard:** Develop an interface for facility managers to manage centers, sports, and view analytics.
5. **Mobile Application:** Create a mobile app version for improved accessibility.
6. **Input Validation:** Implement more robust input validation and error handling throughout the API.
7. **API Documentation:** Create comprehensive API documentation using tools like Swagger.
8. **Enhanced Booking Management:** Develop more sophisticated booking modification and cancellation features.

Conclusion

The Sports Facility Booking System provides a solid foundation for managing sports facility reservations. The current implementation offers core functionality to meet basic booking needs efficiently, with room for future enhancements to improve security, user experience, and administrative capabilities.