Full Stack Development Report: Sports Center Booking System

1. Introduction:

This report covers the design, implementation, and challenges encountered during the development of a sports center booking system. The system supports two types of users: standard users and managers, each with different levels of access. Users can book courts for various sports based on availability, while managers can book courts on behalf of other users for a predefined center. The frontend is built using React.js, while the backend leverages Node.js, Express, and MongoDB to handle the API, database interactions, and business logic.

Frontend Development Report

2. Design Decisions:

The frontend design focuses on simplicity, role-based access, and responsiveness. Key features include:

- **Login Page**: Users authenticate via a login form, which redirects them to the appropriate dashboard based on their role.
- **Role-Based Dashboard**: Standard users can select sports and centers, while managers are limited to their assigned center and have the additional ability to book for other users.
- **Dynamic Booking Page**: Users can choose a date and view available time slots. Each slot is displayed as either "available" or "booked," and users can confirm bookings through a dialog box.

3. Implementation Details:

Technologies used:

- **React.js** for the component-based UI.
- **React Router** for smooth navigation between login, dashboard, and booking pages.

4. Challenges and Solutions:

- **Role-Based Customization**: Implementing different dashboards for users and managers was handled using conditional rendering based on user roles.
- **Time Slot Conflicts**: Backend validation ensures that no overlapping bookings occur, and the UI reflects real-time booking status by fetching slot availability dynamically.

5. Future Improvements:

- **Push Notifications**: Real-time alerts for users about upcoming bookings.
- **Booking History and Cancellations**: Users could view their booking history and cancel or reschedule.
- **Payment Integration**: Future iterations could include payment handling for booking fees.

Backend Development Report

2. Design Decisions:

The backend architecture was designed to support a scalable, role-based booking system with MongoDB as the database. Key design decisions include:

- **User Authentication**: User login is validated using plain-text comparison for simplicity, but this can be enhanced using bcrypt in the future for password hashing(authController).
- **Role-Based Access**: Different routes and controllers handle role-specific logic. For instance, managers can book for other users, whereas standard users can only book for themselves(authRoute)(bookingroute).
- **Booking System**: The system prevents overlapping bookings by checking time slot availability before creating a new booking(bookingController).
- **MongoDB Relations**: Collections for users, centers, courts, sports, and bookings are interconnected using MongoDB's references. For instance, each booking references a center, court, and user(Booking)(Center)(Court)(Sport).

3. Implementation Details:

The backend utilizes:

- **Node.js** and **Express** for server-side logic and routing.
- **MongoDB** as the database to store users, bookings, sports, and centers.
- **Mongoose** for schema management and data validation(index).

Key controllers include:

- **Authentication**: Validates user login and returns user details like role and center(authController).
- **Booking**: Handles booking creation, preventing overlapping bookings(bookingController).
- **Center and Sports Management**: Fetches centers and sports based on user role(centerController)(sportsController).
- **Booking View**: Generates available time slots for a selected date, marking booked slots as "booked"(viewController).

4. Challenges and Solutions:

- **Booking Overlaps**: To prevent overlapping bookings, a validation check was added to ensure no time conflicts when booking a court(bookingController).
- **Dynamic Time Slot Generation**: Time slots from 6 AM to 10 PM are generated dynamically, and booked slots are marked accordingly(viewController).
- **Handling Role-Specific Logic**: Separate controllers and routes were used to handle the different requirements of users and managers, ensuring clear code separation(centerRoute)(sportsRoute)(viewRoute).

5. Future Improvements:

- **Password Encryption**: Use bcrypt for secure password storage and JWT for token-based authentication.
- **Advanced Slot Booking**: Implement pagination or lazy loading for managing larger time ranges and more courts.
- **Data Analytics**: Add features for managers to view booking statistics for their center, including daily or weekly booking trends.

Conclusion:

This project successfully delivered a full-stack sports booking system with role-based features. Both the frontend and backend have been designed to offer scalability, flexibility, and user-friendly interfaces. While there are areas for future improvements, the core functionality of the system has been implemented effectively, ensuring a smooth booking experience for both users and managers.