Project Report: Game Theory <> IIIT-A - Booking Management System

College Id: IEC2021005

Introduction

This project is focused on developing a **sports booking management system** for a company that operates multiple sports centers. The objective is to build a web application that allows customers to book courts for various sports and provides the operations team with tools to manage bookings efficiently. Each center offers multiple sports, and each sport can have multiple courts or resources. The operations team can manage bookings, and customers can view available slots and make reservations.

Design Decisions

Our design was driven by the following key requirements:

1. **Role-Based Views:**
    - Users have different roles (customers and operations team). We used conditional rendering to provide different interfaces for these roles.
2. **Persistent Layout for Operations Dashboard:**
    - A sidebar navigation layout was used with React Router’s nested routes to provide a seamless experience across the application.
3. **State Management with Context API:**
    - We chose to use the Context API to manage global states like the `centreId` and `userRole` efficiently across components. This decision was crucial for avoiding prop-drilling and maintaining a consistent state throughout the application.
4. **Authentication and JWT Management:**
    - The system checks for a JWT token on every page load to ensure only authenticated users can access the app.

Implementation Details

1. **Technologies Used:**
    - **Frontend:** React, React Router, React Context API, Tailwind CSS for styling
    - **Backend:** Node.js and Express for API development
    - **Database:** MongoDB for storing users, bookings, courts, and centers
    - **Date Management:** `date-fns` for formatting and managing dates
    - **Authentication:** JWT for user authentication
2. **Rationale for Technologies:**

- **React Router:** Facilitated dynamic routing with role-based conditional rendering.
- **Context API:** Allowed efficient state management across components without prop-drilling.
- **MongoDB:** Chosen for flexibility in storing and querying complex relationships between courts, sports, and centers.
- **Tailwind CSS:** Provided quick and responsive styling for the UI.

Challenges and Solutions

1. **Challenge: State Management Across Components**
   - **Problem:** Managing global states like `centreId` and `userRole` across multiple nested components without prop-drilling became cumbersome.
   - **Solution:** We adopted the **React Context API** to store and access these global states. This approach ensured that all components could share the required state seamlessly. Additionally, React's `useEffect` hooks were used to sync state changes with local storage, keeping the user session persistent.
2. **Challenge: Role-Based Routing and UI Control**
   - **Problem:** Displaying different UIs based on user roles (customer vs. operations team) while ensuring smooth navigation was challenging.
   - **Solution:** We used React Router's `useLocation` hook to track the current path and conditionally render components based on the user's role. This kept the interface intuitive and responsive.

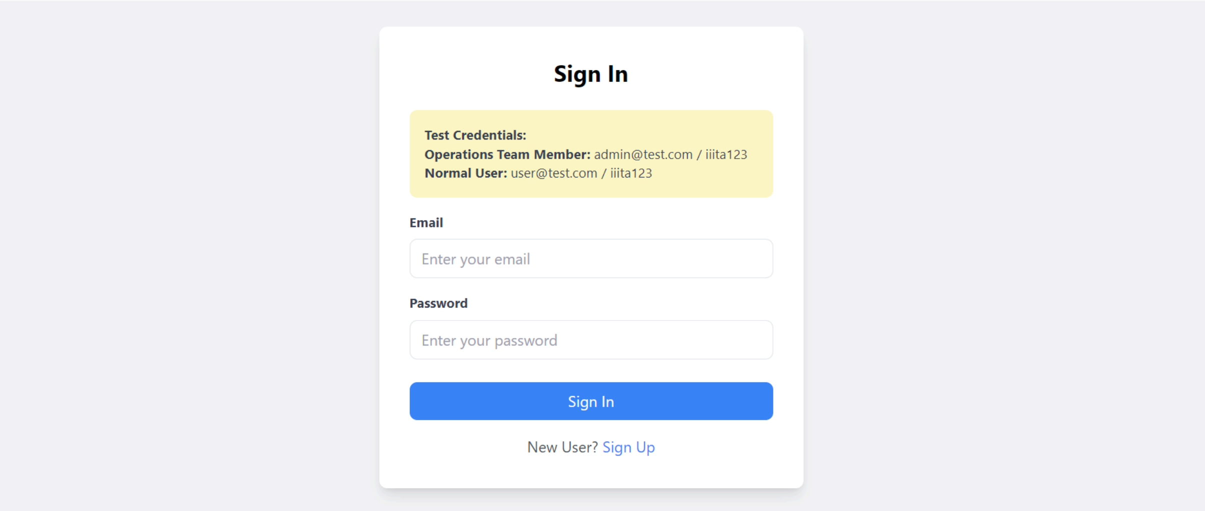
Future Improvements

1. **Improved Error Handling:**
   - Currently, some errors are logged to the console. We plan to implement a global error management system that displays user-friendly messages.
2. **Search and Filtering Options for Bookings:**
   - Adding advanced search options to filter bookings based on courts, sports, and dates would enhance usability.
3. **Real-Time Updates with WebSockets:**
   - Implementing real-time updates for bookings and availability using WebSockets would reduce the need for manual page refreshes.
4. **Mobile-Friendly Interface:**
   - While we used Tailwind CSS for responsiveness, a dedicated mobile UI would further improve the user experience on smaller screens.
5. **Notifications and Reminders:**
   - Adding notification features for booking confirmations and reminders would be helpful for users and operations staff.

Conclusion

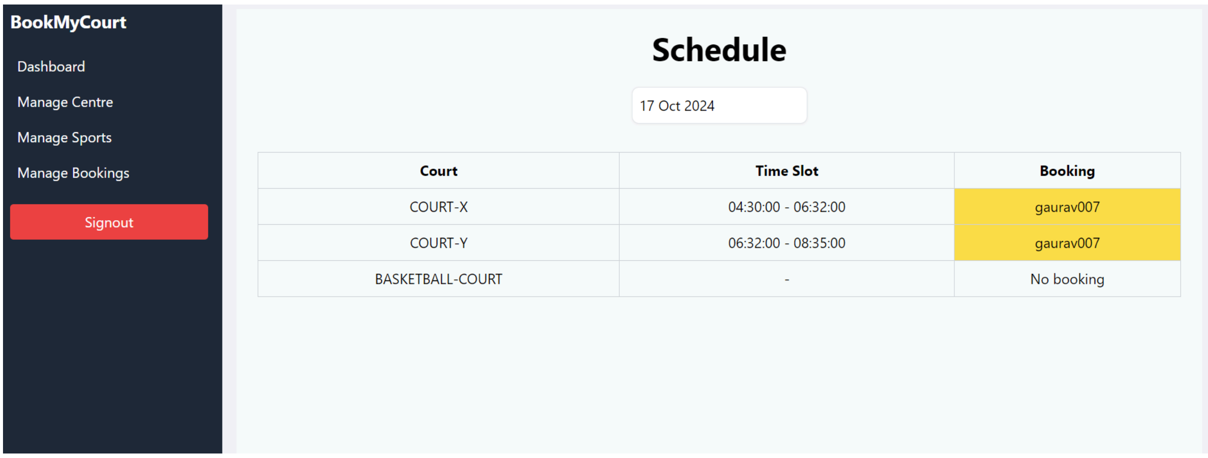
This project was a valuable learning experience, with the core challenge being **state management** across the application. Using the **Context API** allowed us to handle global states efficiently and ensured a smooth user experience. The modular design, role-based rendering, and focus on operational efficiency met the project's primary objectives. With additional time, we plan to add features like real-time updates, advanced search options, and better error handling to further enhance the system.

Application In Action:



Test Credentials:
Operations Team Member: admin@test.com / iiita123
Normal User: user@test.com / iiita123

SignIn using JWT Page

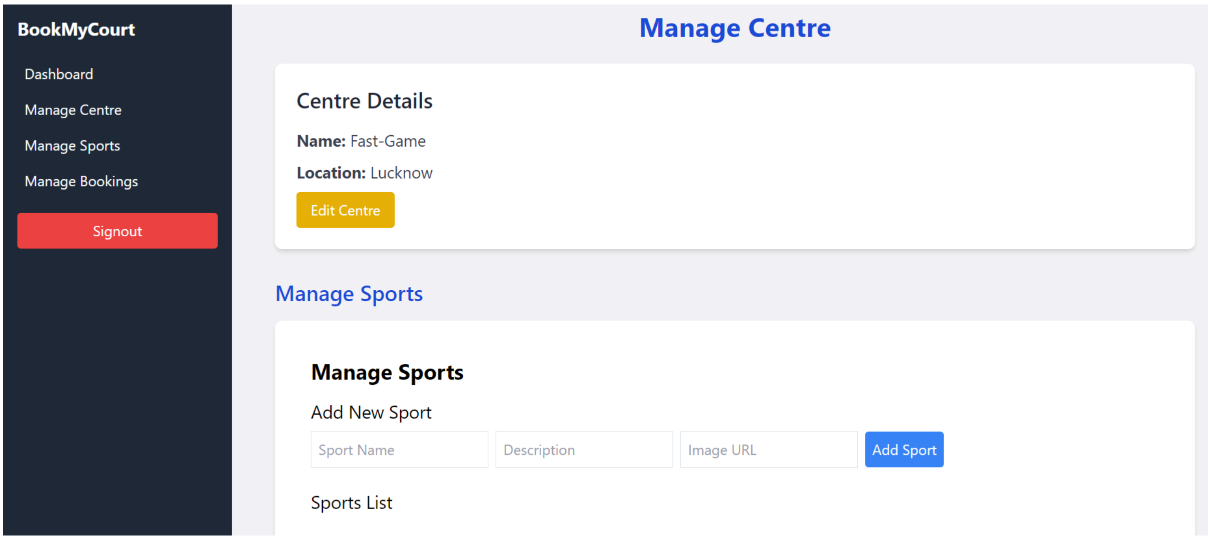


Schedule

17 Oct 2024

Court	Time Slot	Booking
COURT-X	04:30:00 - 06:32:00	gaurav007
COURT-Y	06:32:00 - 08:35:00	gaurav007
BASKETBALL-COURT	-	No booking

Scheduled Dashboard



Manage Centre

Centre Details

Name: Fast-Game

Location: Lucknow

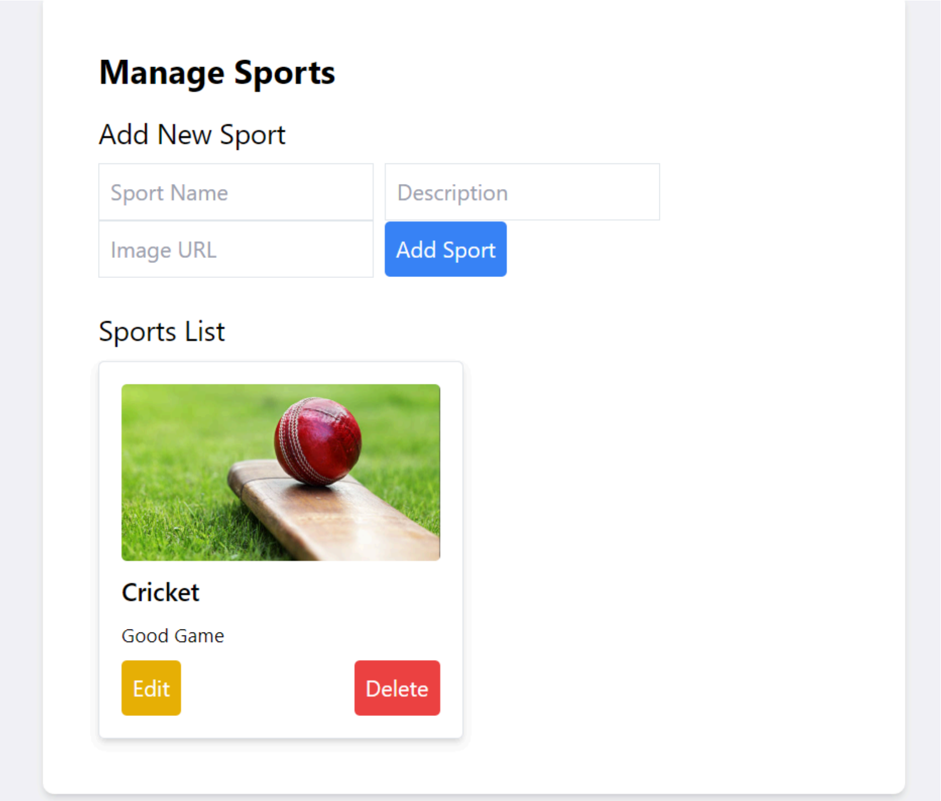
Manage Sports

Manage Sports

Add New Sport

Sports List

Manage Centre Page



Manage Sports

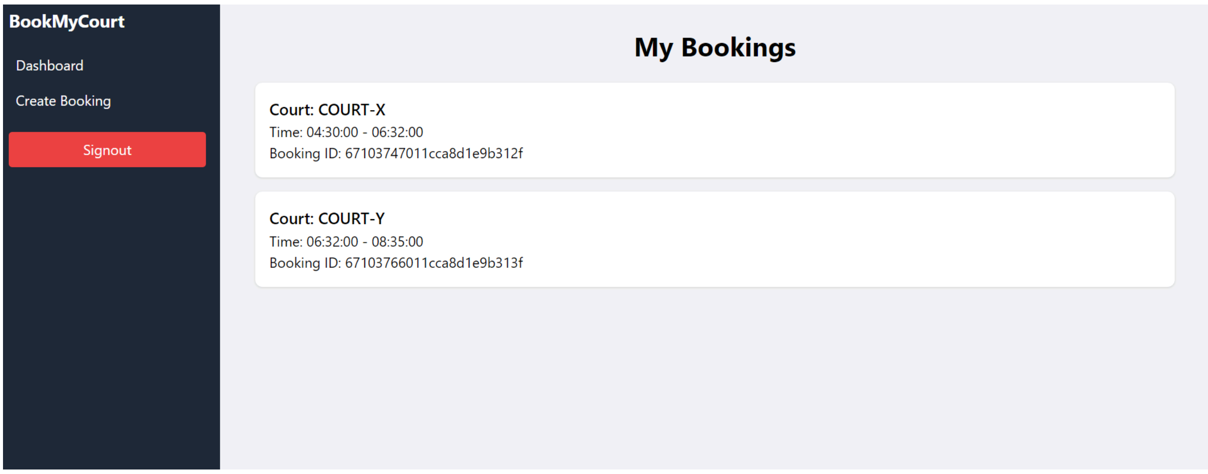
Add New Sport

Sports List

Cricket

Good Game

Adding a new sport to the Centre

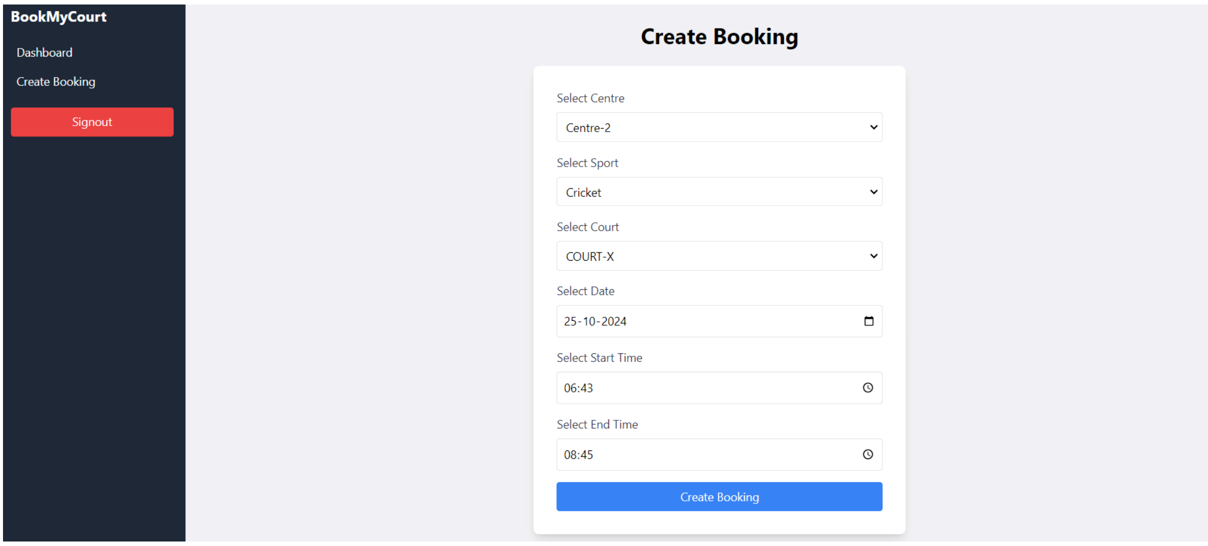


My Bookings

Court: COURT-X
Time: 04:30:00 - 06:32:00
Booking ID: 67103747011cca8d1e9b312f

Court: COURT-Y
Time: 06:32:00 - 08:35:00
Booking ID: 67103766011cca8d1e9b313f

View bookings API



Create Booking

Select Centre: Centre-2

Select Sport: Cricket

Select Court: COURT-X

Select Date: 25-10-2024

Select Start Time: 06:43

Select End Time: 08:45

Create Booking API