Skedulargt Frontend Project Report

1. Introduction

The Skedulargt frontend is a React-based web application designed for managing schedules and bookings for sports facilities. It provides a user-friendly interface for viewing available time slots, making bookings, and managing customer information.

2. Design Decisions

Architecture

The application follows a component-based architecture, typical of React applications. It consists of several reusable components that handle specific functionalities:

- ScheduleTable: The main component that displays the booking schedule and manages the overall state of the application.
- BookingCard: A presentational component for displaying individual bookings.
- BookingModal: A form component for creating new bookings.
- SignupModal: A form component for registering new customers.
- Sidebar: A navigation component for the application.

State Management

The application primarily uses React's built-in state management (useState and useEffect hooks) for handling component-level and application-wide state. This approach is suitable for the current scale

of the application, but as the application grows, consider implementing a more robust state management solution like Redux or Context API.

Styling

The project utilizes Tailwind CSS for styling, which allows for rapid UI development and consistent design across the application.

3. Implementation Details

Technologies Used

- **React 18.3.1:** For building the user interface
- **Vite**: As the build tool and development server
- **Tailwind CSS**: For styling
- **React DatePicke**r: For date selection functionality
- **React Toastify**: For displaying notifications

Key Features

1. Dynamic Schedule Display: The ScheduleTable component fetches and displays bookings based on selected location, sport, and date.
2. Booking Creation: Users can create new bookings through the BookingModal.
3. Customer Registration: New customers can be registered using the SignupModal.
4. Responsive Design: The application is designed to be responsive, with a sidebar for navigation on larger screens.

Code Structure

The application is structured into modular components, each responsible for a specific part of the UI or functionality. This promotes code reusability and maintainability.

4. Challenges and Solutions

Challenge 1: Complex State Management in ScheduleTable

The ScheduleTable component manages multiple states (selected location, sport, date, etc.) and fetches data from the API. This could potentially lead to performance issues or complicated code.

Solution: Consider breaking down the ScheduleTable component into smaller, more focused components. Implement custom hooks for data fetching and state management to improve code organization and reusability.

Challenge 2: Form Handling and Validation

Both BookingModal and SignupModal handle form submissions, but there's limited form validation implemented.

Solution: Implement more robust form validation. Consider using a form library like Formik or react-hook-form for more complex forms to handle validation and submission more efficiently.

Challenge 3: API Error Handling

While the application uses try-catch blocks for API calls, the error handling could be more comprehensive and user-friendly.

Solution: Implement a global error handling mechanism. Use more specific error messages and provide user-friendly feedback for different types of errors (network issues, validation errors, server errors, etc.).

5. Future Improvements

1. State Management: As the application grows, consider implementing a more robust state management solution like Redux or Context API.

2. Authentication and Authorization: Implement a proper authentication system to secure the application and provide personalized experiences for different user roles.

3. Optimistic UI Updates: Implement optimistic updates for actions like creating bookings to improve the perceived performance of the application.

4. Accessibility: Enhance the application's accessibility by adding proper ARIA attributes and ensuring keyboard navigation works correctly.

5. Testing: Implement unit and integration tests to ensure the reliability of the components and the overall application.

6. Performance Optimization: Implement code splitting and lazy loading for larger components to improve initial load times.

7. Internationalization: Add support for multiple languages to make the application accessible to a wider audience.

6. Conclusion

The Skedulargt frontend provides a solid foundation for a sports facility booking system. It demonstrates good use of React components and

hooks, along with modern styling solutions like Tailwind CSS. With the suggested improvements, particularly in state management and error handling, the application can be scaled to handle more complex scenarios and provide an even better user experience.