To create a booking app like the one in the image using React for the frontend and Express for the backend, here is a high-level plan and a breakdown of what needs to be done:

**1. Backend Setup with Express**

* **Install Express**: Set up a basic Express application.
* **Database**: Choose a database to store booking data (e.g., MongoDB, PostgreSQL, or MySQL).
* **API Endpoints**:
  + GET /slots: Fetch available, booked, or holiday slots.
  + POST /book-slot: Book a specific slot.
  + GET /quota: Fetch user's remaining weekly or monthly quota.
  + POST /automatic-slot-allocation: Handle automatic allocation.
* **Logic**:
  + Implement slot booking logic, ensuring no double bookings.
  + Mark slots as public holidays or unavailable.
* **Authentication**: Implement user authentication if needed (e.g., JWT-based).

**2. Frontend with React**

* **Create Components**:
  + **SlotCalendar**: Display slots in a grid with status (e.g., Booked, Public Holiday).
  + **SlotForm**: Form to manage automatic allocation.
  + **Header**: For navigation and displaying information like quotas.
* **State Management**:
  + Use React Context or Redux for state management.
  + Fetch and manage slots, booking status, and user quotas.
* **Styling**:
  + Use a CSS framework like Bootstrap or TailwindCSS to style the app.
  + Implement responsive design for mobile compatibility.
* **API Integration**:
  + Use axios or fetch to call backend APIs.
  + Handle errors (e.g., when a slot is already booked).

**3. Database Schema**

* **Slots Table/Collection**:
  + id (primary key)
  + date (date of the slot)
  + time (time range)
  + status (e.g., Booked, Available, Public Holiday)
  + userId (optional, to track which user booked the slot)
* **User Table/Collection**:
  + id (primary key)
  + name
  + email
  + quotaRemainingWeek
  + quotaRemainingMonth

**4. Implementation Steps**

**Backend**

1. Set up the database and define models for Slots and Users.
2. Create RESTful APIs for fetching, booking, and quota management.
3. Implement middleware for input validation (e.g., express-validator).
4. Test API endpoints using Postman or similar tools.

**Frontend**

1. Set up React using create-react-app or Vite.
2. Create components for the slot calendar, booking form, and quota display.
3. Fetch slot data and display it dynamically in a table/grid.
4. Add interactivity for selecting a slot and submitting a booking.
5. Implement conditional rendering for slot statuses (e.g., color-coding).

**5. Additional Features**

* **Automatic Slot Allocation**: Use an algorithm to assign slots based on user preferences.
* **Notifications**: Send email or SMS confirmations for bookings.
* **Admin Panel**: Allow admin users to manage slots, mark holidays, or edit bookings.
* **Pagination**: Handle large datasets with paginated or lazy-loaded slot grids.

**6. Technologies Used**

* **Frontend**: React, Axios, Bootstrap/TailwindCSS.
* **Backend**: Express, Node.js, MongoDB/PostgreSQL.
* **Authentication**: JWT for user login and session management.
* **Deployment**: Use platforms like Heroku for backend and Vercel/Netlify for frontend.

Let me know if you'd like a step-by-step implementation for any part of this app, and I can provide detailed code examples!

To implement the functionality and design of the given HTML structure in a React app using React components and hooks, you can break the implementation into manageable steps. Here's how to proceed:

**1. Set Up the React App**

1. Create a new React project using create-react-app or Vite:
2. npx create-react-app booking-app
3. cd booking-app

Or, using Vite:

npm create vite@latest booking-app --template react

cd booking-app

npm install

1. Install required dependencies:
   * **Bootstrap** for styling:
   * npm install bootstrap
   * **React Table** for slot listings (optional):
   * npm install @tanstack/react-table
2. Add Bootstrap to index.js:
3. import 'bootstrap/dist/css/bootstrap.min.css';
4. import './index.css';

**2. Create Components**

**Header Component**

Handles the navigation and user information display.

const Header = () => {

return (

<nav className="navbar navbar-expand-lg navbar-light bg-light">

<div className="container-fluid">

<a className="navbar-brand" href="#">EMS Booking</a>

<div className="navbar-nav">

<a className="nav-link" href="/dashboard">Dashboard</a>

<a className="nav-link" href="/booking">Book Equipment</a>

<a className="nav-link" href="/wallet">Wallet History</a>

<a className="nav-link" href="/logout">Log Out</a>

</div>

</div>

</nav>

);

};

export default Header;

**SlotCalendar Component**

Displays the booking slots as a table.

const SlotCalendar = ({ slots }) => {

return (

<table className="table table-bordered text-center">

<thead>

<tr>

<th>Date / Time</th>

<th>Sunday</th>

<th>Monday</th>

<th>Tuesday</th>

<th>Wednesday</th>

<th>Thursday</th>

<th>Friday</th>

<th>Saturday</th>

</tr>

</thead>

<tbody>

{slots.map((row, index) => (

<tr key={index}>

<td>{row.time}</td>

{row.days.map((day, idx) => (

<td

key={idx}

style={{

backgroundColor: day.color,

color: '#fff',

}}

>

{day.status}

</td>

))}

</tr>

))}

</tbody>

</table>

);

};

export default SlotCalendar;

**BookingForm Component**

Handles user input for booking slots.

const BookingForm = ({ onSubmit }) => {

const [slots, setSlots] = React.useState(1);

const [goldCoating, setGoldCoating] = React.useState(false);

const handleSubmit = () => {

onSubmit({ slots, goldCoating });

};

return (

<div>

<div className="form-group">

<label>Number of Slots:</label>

<input

type="number"

className="form-control"

value={slots}

onChange={(e) => setSlots(e.target.value)}

/>

</div>

<div className="form-group">

<label>Do you want Gold Coating?</label>

<input

type="radio"

name="goldCoating"

value="yes"

checked={goldCoating}

onChange={() => setGoldCoating(true)}

/>{' '}

Yes

<input

type="radio"

name="goldCoating"

value="no"

checked={!goldCoating}

onChange={() => setGoldCoating(false)}

/>{' '}

No

</div>

<button className="btn btn-primary" onClick={handleSubmit}>

Submit Booking

</button>

</div>

);

};

export default BookingForm;

**3. Create Main Page**

Combine all components and manage state with React Hooks.

import React, { useState } from 'react';

import Header from './Header';

import SlotCalendar from './SlotCalendar';

import BookingForm from './BookingForm';

const App = () => {

const [slots, setSlots] = useState([

{

time: '9:30AM - 11:00AM',

days: [

{ status: 'Public Holiday', color: '#DC143C' },

{ status: 'Not Available', color: '#808080' },

{ status: 'Booked', color: '#0000FF' },

{ status: 'Booked', color: '#0000FF' },

{ status: 'Not Available', color: '#808080' },

{ status: 'Not Available', color: '#808080' },

{ status: 'Public Holiday', color: '#DC143C' },

],

},

// Add more rows as needed...

]);

const handleBookingSubmit = (data) => {

console.log('Booking Data:', data);

};

return (

<div>

<Header />

<div className="container mt-4">

<h1>Slot Booking System</h1>

<SlotCalendar slots={slots} />

<BookingForm onSubmit={handleBookingSubmit} />

</div>

</div>

);

};

export default App;

**4. Add Styling**

Customize the design using Bootstrap or your CSS to match the given HTML structure.

**5. Backend Integration**

* Create an Express API to serve slot data and handle bookings.
* Example API Endpoint:
* app.get('/api/slots', (req, res) => {
* res.json([
* {
* time: '9:30AM - 11:00AM',
* days: [
* { status: 'Public Holiday', color: '#DC143C' },
* { status: 'Not Available', color: '#808080' },
* { status: 'Booked', color: '#0000FF' },
* ],
* },
* ]);
* });

This approach modularizes the application, making it scalable and easy to maintain. Let me know if you'd like help with the backend setup or additional features like API integration!