***// Frontend Code Snippets for Flight Booking App (React)***

***// 1. Login/Register Component (Login.js)***

**import React, { useState } from 'react';**

**import axios from 'axios';**

**import { useNavigate } from 'react-router-dom';**

**const Login = () => {**

**const [formData, setFormData] = useState({ email: '', password: '' });**

**const navigate = useNavigate();**

**const handleChange = (e) => {**

**setFormData({ ...formData, [e.target.name]: e.target.value });**

**};**

**const handleSubmit = async (e) => {**

**e.preventDefault();**

**try {**

**const res = await axios.post('/api/auth/login', formData);**

**localStorage.setItem('user', JSON.stringify(res.data));**

**if (res.data.usertype === 'admin') navigate('/admin');**

**else navigate('/dashboard');**

**} catch (err) {**

**console.error(err);**

**alert('Invalid credentials');**

**}**

**};**

**return (**

**<form onSubmit={handleSubmit}>**

**<input name="email" placeholder="Email" onChange={handleChange} required />**

**<input type="password" name="password" placeholder="Password" onChange={handleChange} required />**

**<button type="submit">Login</button>**

**</form>**

**);**

**};**

**export default Login;**

***// 2. Flight Search Component (FlightSearch.js)***

**import React, { useState } from 'react';**

**import axios from 'axios';**

**const FlightSearch = () => {**

**const [searchParams, setSearchParams] = useState({ origin: '', destination: '' });**

**const [flights, setFlights] = useState([]);**

**const handleChange = (e) => {**

**setSearchParams({ ...searchParams, [e.target.name]: e.target.value });**

**};**

**const handleSearch = async () => {**

**const res = await axios.get('/api/flights', { params: searchParams });**

**setFlights(res.data);**

**};**

**return (**

**<div>**

**<input name="origin" placeholder="From" onChange={handleChange} />**

**<input name="destination" placeholder="To" onChange={handleChange} />**

**<button onClick={handleSearch}>Search</button>**

**<ul>**

**{flights.map((flight) => (**

**<li key={flight.\_id}>**

**{flight.flightName} - {flight.departureTime} to {flight.arrivalTime}**

**<button onClick={() => bookFlight(flight)}>Book</button>**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

**};**

**export default FlightSearch;**

***// 3. Add Flight (AdminAddFlight.js)***

**import React, { useState } from 'react';**

**import axios from 'axios';**

**const AdminAddFlight = () => {**

**const [formData, setFormData] = useState({**

**flightName: '', flightId: '', origin: '', destination: '', departureTime: '', arrivalTime: '', basePrice: '', totalSeats: ''**

**});**

**const handleChange = (e) => {**

**setFormData({ ...formData, [e.target.name]: e.target.value });**

**};**

**const handleSubmit = async (e) => {**

**e.preventDefault();**

**await axios.post('/api/admin/flights', formData);**

**alert('Flight Added');**

**};**

**return (**

**<form onSubmit={handleSubmit}>**

**<input name="flightName" placeholder="Flight Name" onChange={handleChange} />**

**<input name="flightId" placeholder="Flight ID" onChange={handleChange} />**

**<input name="origin" placeholder="Origin" onChange={handleChange} />**

**<input name="destination" placeholder="Destination" onChange={handleChange} />**

**<input name="departureTime" placeholder="Departure Time" onChange={handleChange} />**

**<input name="arrivalTime" placeholder="Arrival Time" onChange={handleChange} />**

**<input name="basePrice" placeholder="Base Price" onChange={handleChange} />**

**<input name="totalSeats" placeholder="Total Seats" onChange={handleChange} />**

**<button type="submit">Add Flight</button>**

**</form>**

**);**

**};**

**export default AdminAddFlight;**

**Frontend Development**

**11.1 Login / Register**

* A dedicated **Login/Register Component** is created in React, which consists of a form to collect username, password, and user type.
* Upon submission, the input credentials are verified by sending a request to the backend.
* If the login is successful, users are redirected to their respective dashboards:
  + **User Dashboard**
  + **Admin Dashboard**
  + **Flight Operator Panel (if implemented)**

**11.2 Flight Booking (User)**

All booking functionality is implemented through a **Modal component** to enhance UX without navigating away from the search results.

**Flight Search:**

* Users input:
  + Departure city
  + Destination
  + Journey date
  + Class
* A **flight search API call** is made, and available flights are displayed with details such as:
  + Airline name
  + Departure time
  + Arrival time
  + Price
* Each result includes a **"Book Now"** button, which opens the booking modal.

**Booking:**

* Users select:
  + Number of seats
  + Passenger information
  + Class
* On submission, the frontend sends an HTTP POST request to the booking route.
* A confirmation message is displayed upon successful booking.

**11.3 Fetching User Bookings**

* The **Bookings Page** displays all current and past bookings made by the user.
* Each booking entry contains:
  + Flight name
  + Journey date and time
  + Booking status
  + Cancel button
* Clicking the **Cancel** button sends a DELETE request to the backend to cancel the selected booking.

**11.4 Add New Flight (Admin)**

* In the **Admin Dashboard**, an HTML form is presented for adding new flights.
* Input fields include:
  + Flight name
  + Flight ID
  + Origin and destination
  + Departure and arrival time
  + Base price
  + Total seats
* On submission, an HTTP POST request is sent to the backend to store the new flight data in the database.

**11.5 Update Flight (Admin)**

* Admins can edit flight details through an editable table or a separate update form.
* Editable fields include:
  + Departure/Arrival time
  + Base price
  + Total seats
* An HTTP PUT/PATCH request is made to update flight records.

**11.6 Admin Dashboard Features**

* Admin dashboard also includes views to:
  + List all flights
  + View all user bookings
  + Access all registered users
* Each section provides buttons for editing or deleting entries, ensuring full CRUD functionality from the admin side.