Functional Specification: Airline Reservation System Frontend

1. Overview

This document outlines the functional requirements, data structures, and user interface modules for the Airline Reservation System frontend. It is derived from the finalized database schema and stakeholder requirements.

Target Audience: Frontend Developer
Tech Stack: React, TypeScript, Tailwind CSS (recommended)

2. Data Models (TypeScript Interfaces)

These interfaces represent the data shapes expected from the Backend API. Use these as the "contract" for your components.

2.1. User & Auth

```
interface User {
  userId: number;
  firstName: string;
  lastName: string;
  email: string;
  role: "PASSENGER" | "ADMIN";
}

interface UserProfile extends User {
  phoneNumbers: string[]; // Array of phone numbers from UserPhone table
}
```

2.2. Fleet & Flights

```
interface Flight {
  flightId: number;
  flightNumber: string;      // e.g., "MS777"
  origin: string;            // IATA Code, e.g., "CAI"
  destination: string;       // IATA Code, e.g., "DXB"
  departureTime: string;     // ISO 8601 Date String
  arrivalTime: string;       // ISO 8601 Date String
  status: "Scheduled" | "Delayed" | "Cancelled";
  basePrice: number;         // From FlightRoute.BaseFare
```

```
  aircraftType: string;      // e.g., "Boeing 737"
}

interface Seat {
  seatId: number;
  seatNumber: string;        // e.g., "1A"
  seatClass: "Economy" | "Business" | "First";
  isAvailable: boolean;      // Computed field from backend
}
```

2.3. Booking & Payment

```
interface BookingRequest {
  userId: number;
  flightId: number;
  selectedSeats: {
    seatId: number;
    travelerFirstName: string;
    travelerLastName: string;
  }[];
  paymentMethod: string;
}

interface BookingSummary {
  bookingId: number;
  status: "Confirmed" | "Cancelled";
  totalCost: number;
  bookingDate: string;
  tickets: Ticket[];
}

interface Ticket {
  ticketId: number;
  flightNumber: string;
  seatNumber: string;
  travelerName: string;
}
```

3. Module 1: Authentication & Profile (Public)

3.1. Login Screen

- **Route:** /login
- **UI Components:**
  - Email Input (Validation: Email format)
  - Password Input
  - "Login" Button
- **Logic:**
  - On success: Store JWT token.
  - **Critical:** Check user.role.
    - If PASSENGER -> Redirect to /home
    - If ADMIN -> Redirect to /admin/dashboard

3.2. Registration Screen

- **Route:** /register
- **UI Components:**
  - First Name, Last Name
  - Email, Password, Confirm Password
  - Phone Number (Allow adding multiple fields dynamically)
- **Logic:** Creates a new User record with default role PASSENGER.

4. Module 2: Passenger Portal (Booking Flow)

4.1. Home / Flight Search

- **Route:** / or /home
- **UI Components:**
  - **Search Widget:**
    - Origin (Dropdown of Airport Codes)
    - Destination (Dropdown of Airport Codes)
    - Date (Date Picker)
  - "Search Flights" Button.

4.2. Search Results

- **Route:** /flights?origin=CAI&dest=DXB&date=...
- **UI Components:**
  - **Flight Card (List Item):**
    - Displays: Flight Number, Dep/Arr Time, Duration, Price.
    - Action: "Select Flight" Button.
- **Logic:** Fetch list of Flight objects matching criteria.

4.3. Seat Selection (Interactive)

- **Route:** /book/:flightId/seats
- **UI Components:**
  - **Visual Seat Map:** A grid representing the aircraft layout.

        - **Legend:** Available (Green), Taken (Gray), Selected (Blue).
    - **Logic:**
        - Fetch all Seat records for the aircraft type.
        - Fetch booked Ticket records for this specific flight.
        - Disable seats that are already booked.
        - Allow user to toggle selection of multiple seats.

4.4. Traveler Details

- **Route:** /book/:flightId/details
- **UI Components:**
    - **Dynamic Form:** For *each* selected seat, show a form:
        - "Traveler Name for Seat 1A" (Inputs: First Name, Last Name).
        - "Same as User?" Checkbox (Auto-fills with logged-in user's name).
- **Logic:** Collects the TravelerFirstName and TravelerLastName for the Ticket entity.

4.5. Payment & Confirmation

- **Route:** /book/:flightId/payment
- **UI Components:**
    - **Booking Summary:** Flight details + Total Price (BaseFare * Seat Count).
    - **Payment Method:** Radio buttons (Credit Card, PayPal).
    - "Pay & Confirm" Button.
- **Logic:**
    1. Send BookingRequest payload to backend.
    2. Backend creates Booking, processing Payment, and creating Tickets.
    3. On success, redirect to "Success" page showing BookingID.

4.6. My Bookings

- **Route:** /my-bookings
- **UI Components:**
    - List of past and upcoming bookings.
    - Status Badges (Green for "Confirmed", Red for "Cancelled").
    - "View Ticket" Button (Shows ticket details).
    - "Cancel Booking" Button (Only if status is Scheduled).

5. Module 3: Admin Dashboard (Management)

- **Access Control:** Restricted to users with role = 'ADMIN'.

5.1. Admin Home (Reports)

- **Route:** /admin/dashboard
- **UI Components:**
    - **KPI Cards:**
        - Total Revenue (Sum of Payment.Amount)

        - Total Bookings (Count of Booking)
        - Active Flights (Count of Flight where status = 'Scheduled')
    - **Occupancy Chart:** Simple bar chart showing seats sold vs. total seats per route.

5.2. Flight Management

- **Route:** /admin/flights
- **UI Components:**
    - **Data Grid:** Table of all flights (ID, Number, Route, Date, Status).
    - **Actions:**
        - "Add Flight" Modal: Select Route, Aircraft, and Date.
        - "Update Status" Dropdown: Change status (e.g., set to "Delayed").

5.3. Booking Management

- **Route:** /admin/bookings
- **UI Components:**
    - Search Bar: Find booking by User Email or Booking ID.
    - Booking Detail View: Show passenger info and payment status.
    - **Admin Action:** "Force Cancel" button (for customer service scenarios).

6. Technical Notes for Developer

1. **State Management:** Use React Context or Redux to store the User object and the current BookingInProgress state (so data isn't lost between Search -> Seat -> Payment steps).
2. **API Handling:** All API calls should handle 401 (Unauthorized) by redirecting to Login.
3. **Validation:** Frontend must validate:
    - Email format.
    - At least 1 seat selected before proceeding.
    - Traveler names cannot be empty.
4. **Dates:** Use a library like date-fns or dayjs to format ISO strings (e.g., 2025-11-15T10:00:00) into user-friendly formats (e.g., "Nov 15, 10:00 AM").