✈️ Flight Booking System – Project Documentation

1. Project Overview

The goal is to design and implement a **responsive flight booking application** similar to Booking.com. The system will enable users to:

- Search flights based on origin, destination, date, and passenger count.
- View available flight options with details such as airline, time, duration, price, and seat availability.
- Book tickets by filling in personal and travel details.
- Get a booking confirmation with unique booking reference and summary.

The app will follow **modern web development practices**, leveraging **latest tools and technologies** for frontend, backend, database, and AI-assisted development.

---

2. System Features

🔹 Frontend (UI/UX)

- Homepage: Flight search form (origin, destination, date, passenger count).
- Search results: List of flights with airline name, time, price, duration, and **“Book Now”** CTA.
- Booking form: Flight summary, passenger details (name, age, passport), seat selection.
- Confirmation screen: Booking reference, passenger summary, and payment status.
- Clean, responsive layout with **mobile-first design**.

🔹 Backend

- REST APIs for flight search, booking, and booking confirmation.
- Integration with **Supabase (Postgres)** for data storage.
- Data validation, error handling, and secure environment variable management.
- Support for mock APIs (for real-world testing with flight data).

🔹 Database (SQL Schema)

- **Users**: UUID, name, email, password_hash, created_at.
- **Flights**: id, flight_number, airline_name, departure_city, departure_airport, arrival_city, arrival_airport, departure_time, arrival_time, duration, price, seats_available.
- **Bookings**: id, user_id (FK), flight_id (FK), booking_reference, passenger details, seat_number, total_price, status (confirmed/pending/cancelled).
- Indexes on critical columns (flight_number, departure_time).
- ON DELETE CASCADE for relational integrity.

---

3. Tech Stack

🖥️ Frontend

- **React.js** – Component-based UI development.
- **Tailwind CSS** – Utility-first CSS for responsive, mobile-friendly design.
- **React Router** – Navigation and routing.
- **Shadcn/UI** – Pre-built accessible UI components.

⚙️ Backend

- **Node.js (Express.js)** – RESTful API development.

- **Supabase** – Database + Authentication (Postgres-based).
- **Dotenv** – Secure environment variable handling.
- **Cors Middleware** – Allow frontend-backend communication.

🛢️ Database

- **PostgreSQL (via Supabase)** – Relational database with SQL schema.
- Indexes & constraints for query optimization and integrity.

🧪 Tools & AI Assistance

- **Cursor AI** – Assisted code generation for frontend.
- **Replit AI** – Backend prototyping and debugging.
- **SQL Editor AI** – Auto-generating schemas and queries.
- **Claude (Mock API)** – Simulating real flight data for testing.

---

4. API Endpoints

1. **Search Flights**
   - `GET /flights?origin=DEL&destination=BOM&date=2025-09-10`
   - Returns list of available flights with details.
2. **Get Flight Details**
   - `GET /flights/:id`
   - Returns details for a single flight.
3. **Create Booking**

- `POST /bookings`
- Request: flight_id, user_id, passenger details.
- Response: booking reference, confirmation status.

4. **Get Booking Details**
    - `GET /bookings/:id`
    - Returns booking summary (flight info + passenger info).

---

5. Development Workflow

1. **Design & Prototyping**:
    - Figma for wireframes and UI mockups.
    - Tailwind CSS + Shadcn for clean responsive layouts.
2. **Frontend Development**:
    - React components + state management.
    - API integration with backend.
3. **Backend Development**:
    - Express.js routes + controllers.
    - Supabase integration.
    - Validation, authentication, and error handling.
4. **Database Setup**:
    - SQL schema creation.
    - Sample data population (10+ sample flights).

5. **Testing**:
   - Unit tests with Jest (backend).
   - Integration tests with React Testing Library.
   - API testing with Postman.
6. **Deployment**:
   - Frontend: Vercel / Netlify.
   - Backend: Render / Railway / Supabase Functions.
   - Database: Supabase (managed Postgres).

---

6. Future Enhancements

- Payment gateway integration (Stripe/PayPal).
- Real-time seat availability via WebSockets.
- AI-powered flight recommendations.
- Multi-language support.
- Progressive Web App (PWA) version.