Flight Booking System — Project Requirement Document (V0.dev)

---

1) Product Overview

A **responsive flight booking app** inspired by Booking.com that allows users to: - Search for flights by origin, destination, date, and passenger count. - Browse flight results with airline, time, price, and duration. - Book flights with passenger details and seat selection. - Get booking confirmation with a reference number and itinerary.

---

2) Core Features (MVP)

Frontend (Web)

- Homepage with **flight search form** (origin, destination, date, passenger count).
- Search results page: list of flights (airline, time, price, duration) with **"Book Now"** CTA.
- Booking form: flight summary, user info (name, age, passport), seat selection.
- Confirmation screen: booking reference, passenger and flight summary.
- Responsive UI with **clean layout**, **Tailwind CSS**, and **mobile-friendly components**.

Tech Stack: - **React (Next.js)** for components, routing, and SSR for SEO. - **Tailwind CSS** + shadcn/ui for styling. - **Lucide Icons** for visuals. - **Cursor AI** for assisted code generation.

Backend (Server)

- **Node.js (Express.js)** for REST API endpoints.
- APIs for searching flights, creating bookings, and fetching booking details.
- Replit AI for backend scaffolding/testing (optional).
- Integration with flight data providers or mock APIs.

Endpoints (initial): - `GET /flights` → search flights using query params (origin, destination, date). - `GET /flights/:id` → fetch flight details. - `POST /bookings` → create booking (flight_id, passenger info). - `GET /bookings/:id` → fetch booking details.

Middleware/Features: - Input validation (Zod/class-validator). - Error handling with proper HTTP status codes. - Async/await for all DB operations. - Enable **CORS** for frontend integration. - Use **dotenv** for environment variables.

Database

- **PostgreSQL** (Supabase or RDS) for persistent data.
- **Prisma ORM** for schema and migrations.

Schema: - **Users**: id (UUID), full_name, email (unique), password_hash, created_at. - **Flights**: id, flight_number, airline_name, departure_city, arrival_city, departure_time, arrival_time, duration_minutes, price, seats_available, created_at. - **Bookings**: id, user_id (FK), flight_id (FK), booking_reference (unique), passenger_name, passenger_age, passport_number, seat_number, total_price, status (confirmed/pending/cancelled), created_at. - **Payments (future)**: id, booking_id, provider, status, amount, created_at.

Indexes: flight_number, departure_time for search optimization.

Constraints: - ON DELETE CASCADE for FK relations. - Unique booking reference.

Seed Data: - Add ~10 sample flights + sample users for development/testing.

API & Data Flow

1. User searches → frontend calls `/flights` API → results returned from DB/mock data.
2. User selects → frontend calls `/bookings` API → booking saved with reference.
3. Booking confirmation → display + persist reference ID.

Mock Data & Testing

- Create a script to seed **10 mock flights** in DB.
- Include mock API integration for real flight data (Amadeus/Skyscanner API in future).
- Provide **clear folder structure** (routes, controllers, services, db).
- Document steps to run & test locally.

---

3) Roadmap

Phase 1 (MVP) - Search + results page - Booking form + confirmation - SQL schema with Supabase/Postgres - Basic Express.js backend APIs

Phase 2 - Real-time flight data API integration - Payment gateway (Razorpay/Stripe) - Notifications (email/SMS) - Seat selection logic

Phase 3 - Redis caching for search queries - OpenSearch integration for faceted search - Microservices split (search, booking, payments) - Containerization (Docker, Kubernetes)

Phase 4 - AI Recommendations (cheapest/fastest flights) - Analytics Dashboard (bookings, revenue) - Progressive delivery (feature flags, A/B tests)

4) Tools & Tech Stack Summary

- **Frontend**: React (Next.js), Tailwind CSS, shadcn/ui, Lucide Icons.
- **Backend**: Node.js (Express.js), REST APIs, Prisma ORM.
- **Database**: PostgreSQL (Supabase for managed dev env).
- **Infra (future)**: Docker, Kubernetes, Terraform, AWS/GCP.
- **Payments**: Razorpay/Stripe.
- **Comms**: AWS SES (email), Twilio/SNS (SMS).
- **AI tools**: Cursor AI, Replit AI, Claude for mock APIs.

---

5) Deliverables

- End-to-end **responsive flight booking app**.
- Documented **API contract** (Swagger/OpenAPI).
- **SQL schema** + mock seed data.
- **Clear README** for local setup, API usage, and testing.

This document covers the baseline MVP and future roadmap for building a scalable flight booking system similar to Booking.com, leveraging modern 2025-ready tech stacks and AI-assisted tooling.