🛫 FLIGHT FINDER PROJECT – REQUIREMENTS PHASE DOCUMENTATION

This document outlines the **essential functional and non-functional requirements** for the Flight Finder application, which helps customers find and book flights efficiently. It includes specifications, user stories, system requirements, and APIs to guide the development and testing phases.

1. FUNCTIONAL REQUIREMENTS

1.1 USER MANAGEMENT (ROLES: ADMIN, OPERATOR, CUSTOMER)

Customer can:

Register and log in using email and password

Search for flights and make bookings

View and cancel their bookings

Operator can:

Add, edit, or remove flight details

View all flight bookings

Admin can:

Approve or decline new **Customer** and **Operator** registrations

Manage Operators and Customers

View complete booking statistics and user activities

1.2 FLIGHT SEARCH & BOOKING

Customers can search flights based on:

Origin

Destination

Date

Results will show flight details such as:

Flight name, departure/arrival time, duration, and price

Customers can book a selected flight

1.3 BOOKING MANAGEMENT

Customers can:

View their bookings

Cancel any previously booked flight

Operators and Admins can:

View booking reports

1.4 ADMIN DASHBOARD

Admins can:

Approve or reject Operator/Customer accounts

View analytics for:

Daily bookings

Most used routes

Cancellations

1.5 ERROR HANDLING AND ALERTS

Success/failure messages via popups

Server/API failure messages shown in a user-friendly way

2. NON-FUNCTIONAL REQUIREMENTS

2.1 PERFORMANCE

Flight search API responses under **2 seconds**

2.2 SCALABILITY

System should scale to handle:

Thousands of bookings

Concurrent flight searches

2.3 SECURITY

Password encryption: **bcrypt**

Token-based authentication: **JWT**

Admin-only and operator-only routes protected via **role-based authorization**

2.4 USABILITY

Clean, responsive, and intuitive UI (built in **React JS**)

Cross-browser support

2.5 MAINTAINABILITY

Modular codebase using:

React (Frontend)

Node.js/Express (Backend)

MongoDB (Database)

Proper inline and external documentation

3. USE CASE SCENARIOS

Use Case	Description
UC1	A **Customer** registers, logs in, searches for flights
UC2	A **Customer** books a flight and views/cancels bookings
UC3	An **Operator** adds or updates flight information
UC4	An **Admin** approves new Customer and Operator accounts
UC5	An **Operator** views all bookings and modifies flight data
UC6	An **Admin** analyzes booking trends in dashboard

4. SYSTEM REQUIREMENTS

Frontend: React JS 18+, Bootstrap / Tailwind CSS

Backend: Node.js (v14+), Express

Database: MongoDB Atlas / Local MongoDB

Authentication: JWT, bcrypt

API Tool: Postman / Thunder Client

Dev Tools: VS Code, Git, GitHub

Supported OS: Windows, macOS, Linux

5. API TEMPLATES

5.1 USER (CUSTOMER/OPERATOR) REGISTRATION TEMPLATE

```
POST /api/register
{
  "name": "Jane Smith",
  "email": "jane@example.com",
  "password": "********",
```

```
  "role": "Customer" // or "Operator"
}
```

5.2 FLIGHT SEARCH API

```
GET /api/flights?
origin=Delhi&destination=Mumbai&date=2025-07-21
```

5.3 ADD FLIGHT (OPERATOR ONLY)

```
POST /api/operator/add-flight
{
  "flightName": "AI202",
  "origin": "Delhi",
  "destination": "Mumbai",
  "departureTime": "10:00",
  "arrivalTime": "12:30",
  "price": 4500
}
```

5.4 ADMIN APPROVE USER

```
POST /api/admin/approve-user
{
  "userId": "6473ac238f3a",
  "status": "approved" // or "declined"
}
```