TEST PLAN DOCUMENT – FLIGHT FINDER PROJECT

Objective:
To validate that the Flight Finder application meets all functional and non-functional requirements.

1. TEST SCOPE

Functional Testing (UI, APIs, Admin Panel, Booking)

Non-Functional Testing (Performance, Usability, Security)

2. TEST TYPES

Test Type	Description
Unit Testing	Component-wise testing (React, Node.js functions)
Integration Testing	Frontend-Backend data flow (API calls, DB updates)
System Testing	End-to-end testing of the entire application
Regression Testing	Post-deployment checks after fixes or updates
UAT	Performed by customers/operators for real use cases

3. TEST SCENARIOS AND CASES

Test ID	Scenario	Steps	Expected Result
TC01	Customer Registration	Fill form → Submit	New user created, success alert
TC02	Operator Flight Entry	Login as operator → Add flight → View flight list	Flight added & visible
TC03	Admin Approval	Login as admin → Approve pending user	User becomes active
TC04	Flight Search	Enter origin, destination, date → Search	Available flights listed
TC05	Booking a Flight	Select flight → Confirm booking	Booking confirmation message

4. TOOLS

Postman: API testing

Jest / React Testing Library: Frontend testing

Mocha + Chai: Backend unit tests

Manual Testing: End-to-end validation

✅ ER DIAGRAM – FLIGHT FINDER

Here's a simple Entity Relationship structure:

```
[Customer]
   | customer_id (PK)
   | name
   | email
   | password
   | role (Customer/Operator)
       |
       |---<books>---[Booking]---<relates to>---[Flight]
                              |
                             flight_id (FK)
                             customer_id (FK)
                             booking_date
                             status

[Flight]
   | flight_id (PK)
   | flight_name
   | origin
   | destination
   | departure_time
   | arrival_time
   | price
   | added_by (Operator)

[Admin]
   | admin_id (PK)
```

```
    | email
    | password
```

✅ DEPLOYMENT STEPS DOCUMENT

Goal: Deploy the Flight Finder application in a production or staging environment.

1. FRONTEND (REACT.JS)

Run: `npm run build`

Deploy build folder to:

Firebase Hosting

Netlify / Vercel

Apache / Nginx if using VPS

2. BACKEND (NODE.JS WITH EXPRESS)

Install Node.js v14+

Clone backend repository

Install dependencies: `npm install`

Create `.env` with:

Run server: `npm start` or `node server.js`

For production: Use `pm2` or `forever`

3. DATABASE (MONGODB ATLAS)

Set up a free MongoDB cluster at MongoDB Atlas

Create DB and collections: `users`, `flights`, `bookings`

Add IP whitelist & connection URI to backend `.env`

4. DOMAIN & SSL (OPTIONAL)

Point custom domain (GoDaddy, Namecheap) to hosting platform.

Use Let’s Encrypt or hosting provider’s built-in SSL.

```
PORT=5000
MONGO_URI=your_connection_string
JWT_SECRET=your_jwt_secret
```