**Project Problem Statement: Flight Management Portal**

**1. Background**

Air travel has become the most convenient mode of transport, but managing flight bookings, schedules, passengers, and staff is complex. Airlines often face challenges like:

* Managing **multiple flights, routes, and schedules**
* Handling **real-time seat availability**
* Ensuring **secure ticket booking and cancellation**
* Providing **easy search options** for passengers (flight by date, route, airline)
* Testing and validating the entire workflow

To address these challenges, the **Flight Management Portal** project will be developed using **Core Java (backend & console UI), PostgreSQL (database management), and Selenium (automation testing)**.

**2. Problem Statement**

The goal is to **develop a robust, modular, and testable Flight Management Portal** that:

* Allows **Passengers** to register, search flights, book/cancel tickets, and check booking history
* Allows **Admins** to manage flights, schedules, and passenger records
* Uses **PostgreSQL** for reliable data storage
* Uses **Selenium** for automated functional and regression testing

**3. Objectives**

The **main objectives** of the Flight Management Portal are:

1. Implement **Core Java** for business logic (OOP, collections, JDBC, multithreading).
2. Store/manage flight, passenger, and booking data in **PostgreSQL**.
3. Ensure **real-time seat availability** and **concurrency handling** during bookings.
4. Provide **role-based access** (Passenger vs. Admin).
5. Automate critical workflows (login, booking, cancellation, flight search) using **Selenium**.
6. Maintain **scalability** for adding more routes and airlines in the future.

**4. System Requirements**

**Functional Requirements**

* **Passenger Module**
  + Register & Login
  + Search Flights (by date, source, destination, airline)
  + Book Tickets (choose seat count)
  + Cancel Booking
  + View Booking History
* **Admin Module**
  + Login with admin role
  + Add/Update/Delete Flights
  + Manage Flight Schedules (time, route, seats, status)
  + View Passenger Bookings
  + Generate Reports (e.g., flight occupancy, cancellations)
* **Booking Module**
  + Show available seats for a flight in real time
  + Prevent overbooking with concurrency control
  + Generate Booking ID + confirmation details
* **Testing Module (Selenium)**
  + Automated UI test cases for:
    - Login (Passenger & Admin)
    - Flight search
    - Booking & cancellation
    - Admin operations (add/update flights)

**Non-Functional Requirements**

* **Performance:** Must handle concurrent bookings
* **Security:** Protect user login & booking data
* **Usability:** Menu-driven console UI (simple navigation)
* **Reliability:** Accurate seat tracking, no duplicate bookings

**5. Database Schema (PostgreSQL)**

**Tables**

**1. Users (Passengers/Admins)**

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15),

role VARCHAR(10) CHECK (role IN ('PASSENGER','ADMIN')) DEFAULT 'PASSENGER'

);

**2. Flights**

CREATE TABLE flights (

flight\_id SERIAL PRIMARY KEY,

airline VARCHAR(50) NOT NULL,

source VARCHAR(50) NOT NULL,

destination VARCHAR(50) NOT NULL,

total\_seats INT NOT NULL

);

**3. Schedules**

CREATE TABLE schedules (

schedule\_id SERIAL PRIMARY KEY,

flight\_id INT REFERENCES flights(flight\_id) ON DELETE CASCADE,

departure\_time TIMESTAMP NOT NULL,

arrival\_time TIMESTAMP NOT NULL,

available\_seats INT

);

**4. Bookings**

CREATE TABLE bookings (

booking\_id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,

schedule\_id INT REFERENCES schedules(schedule\_id) ON DELETE CASCADE,

seats\_booked INT NOT NULL,

booking\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

status VARCHAR(20) CHECK (status IN ('CONFIRMED','CANCELLED')) DEFAULT 'CONFIRMED'

);