## **FLIGHT BOOKING SYSTEM OODP**

## **1. Background**

In today's fast-paced digital world, the airline industry depends heavily on robust, scalable, and secure flight booking systems. These systems must efficiently handle:

* Millions of simultaneous users,
* Real-time search, booking, and payment transactions,
* Secure user authentication,
* Notifications for users at each booking stage,
* Handling of failures (login, payment, booking),
* Dynamic payment method support,
* State management of bookings (Pending → Confirmed → Canceled → Refunded).

Building such a system requires a solid **Object-Oriented Design** approach with the application of **design patterns** like **Singleton**, **Strategy**, **State**, and **Observer** to ensure maintainability, scalability, and fault tolerance.

## **2. Objective**

The objective of this project is to design and implement a **complete flight booking system** that:

* Allows users and administrators to securely **login** and **logout**.
* Provides flight search and booking functionalities.
* Supports multiple **payment methods** through the **Strategy Pattern**.
* Manages booking statuses using the **State Pattern**.
* Handles **login verification failures**, **payment failures**, and **booking failures** gracefully.
* Sends real-time **notifications** (confirmation or failure) via the **Observer Pattern**.
* Maintains centralized management of flights and bookings using the **Singleton Pattern**.
* Ensures the system is highly extendable for future features like coupons, loyalty programs, and more payment integrations.

## **3. Design**

### **3.1 System Design**

This system will follow a **three-tier architecture**:

* **Presentation Layer**: Web/mobile frontend for users and admins (HTML/CSS, JavaScript, Flutter or React Native).
* **Application Layer**: Backend logic and RESTful API services (Java Spring Boot, Node.js, or Django).
* **Data Layer**: Database for storing all persistent data (MySQL, PostgreSQL, or MongoDB).

#### **Key Modules:**

* **User Management Module**: Handles registration, login, and authentication.
* **Flight Management Module**: Allows admin to add/update/delete flights.
* **Booking Module**: Lets users book, cancel, and modify tickets.
* **Payment Module**: Integrates third-party payment gateways (e.g., Stripe, PayPal).
* **Notification Module**: Sends confirmation emails or SMS after booking/payment.

| **Actor** | **Use Case** | **Description** |
| --- | --- | --- |
| Customer | Register/Login | Allows users to create an account or log in |
| Customer | Search Flights | Searches for flights based on date, location, and class |
| Customer | View Flight Details | Displays time, duration, seat availability, and price |
| Customer | Book Flight | Fills passenger details and selects seats |
| Customer | Make Payment | Pays for the selected flight using online payment |
| Customer | Receive Booking Confirmation | Gets confirmation via email/SMS |
| Customer | View/Cancel/Modify Bookings | Accesses past and upcoming bookings |
| Admin | Login | Authenticates to access admin features |
| Admin | Add/Edit/Delete Flights | Manages flight schedules |
| Admin | View User Bookings | Monitors individual or total bookings |
| Admin | Generate Booking/Revenue Reports | Views statistics and performance |
| Payment Gateway | Process Payment | Processes user transaction securely |

### **3.1.2 Sequence Diagram**

**Scenario: Flight Booking by Customer**

Customer --> System: Login

System --> Database: Authenticate user

Database --> System: Success

System --> Customer: Dashboard

Customer --> System: Search flights (from, to, date)

System --> Database: Query available flights

Database --> System: Return flight list

System --> Customer: Display flight list

Customer --> System: Select flight and enter passenger details

Customer --> System: Initiate payment

System --> Payment Gateway: Process payment

Payment Gateway --> System: Payment success/failure

System --> Database: Save booking and payment details

System --> Customer: Display confirmation and send email