**FLIGHT MANAGEMENT SYSTEM**

### **PREPARED FOR**

JEE Cloud Training Program

Capgemini Technology Services

### **PREPARED BY**

Devang Singh

Navya Likhita

Surya

**Introduction:**

The Flight Management System is a Java-based booking solution for flight tickets. It consolidates data provided by different airline carriers and hence provides the user details and rates in real-time. Travellers may want to make changes in their bookings. The application allows them to book, cancel, view and update their bookings with ease. Other than this, it eases the management of bookings too. All the bookings, flights, schedules and routes can be viewed, added and modified on a single application by the administrator.

Following is the functionality provided by the system:

There are two categories of people who would access the system: customer and administrator. Each of these would have some exclusive privileges.

1. The customer can:
   1. Create his user account.
   2. Login into the application.
   3. Check for available flights.
   4. Make a booking.
   5. View the bookings made.
   6. Cancel or modify a booking.
2. The administrator can:
   1. Login into the application.
   2. Add flight, schedule and route details.
   3. View the flight, schedule and route details.
   4. Cancel or modify the flight, schedule and route details.

**Use Case view:**

The following diagram represents the interaction of the customer and the administrator with the system. It also shows the various use cases in which both types of users are involved. It provides a high-level view of the working of the system. It is also providing a requirement analysis for the system.

**Class and Methods Description:**

However, we have made a few assumptions with respect to the application, which are:

1. Administrator and customer are both Users. They are differentiated by a variable ‘userType’ in the User class.
2. Every passenger needs to enter a Unique Identification Number while booking is being made. For simplicity, we assume it to be a 12-digit Aadhaar Number.
3. All flights are direct flights.
4. No flight gets cancelled.

The application will consist of the following classes, the utility of which have been described below:

1. **User:** This class stores the details of any type of user: Customer or Administrator. The User Name is a String containing multiple words. The User Password are is a

Attributes:

userId: Long

userName: String

userPassword: String

userPhone: Long

userEmail: String

userType: String.

Methods: -

*logIn()*: Generates a Login Token for the User if the credentials are correct. Based on the user type the field ‘userType’ is assigned corresponding value.

The value of the field ‘userType’ decides the actions to be provided to the user.

*logOut()*: Removes all the session related information of a user.

The following methods will only be allowed for user type: Customer. :-

*addBooking(Long, String, Short, List<Passenger>, DateTime, Double, Flight):Booking* :-

Creates a new booking.

*modifyBooking(Long, String, Short, List<Passenger>, DateTime, Double, Flight): Booking* :- Modifies a previous booking identifiable by the ‘bookingId’ and ‘userId’. All information related to the booking except the booking id can be modified.

*viewBooking(Long): List<Booking>* :-

Retrieves a list of bookings made by the user.

*deleteBooking(Long): void* :-

Deletes a previous booking identifiable by the ‘bookingId’ and ‘userId’.

The following methods will only be allowed for user type: Administrator. :-

*addFlight(Long, String, String, Integer): Flight :-*

Adds a new flight which can be scheduled.

*modifyFlight(Long, String, String, Integer): Flight :-*

Modify the details of a flight.

*viewFlight(): List<Flight> :-*

View the details of all flights.

deleteFlight(Flight): void :-

Removes a flight.

*scheduleFlight(Flight, Schedule, int): ScheduledFlight* :-

Schedules a flight alongwith its timings, locations and capacity

*modifyScheduledFlight(Flight, Schedule, int): ScheduledFlight* :- Modifies the details of an added flight retrievable by Flight Number. The flight schedule and capacity can also be modified here.

*viewScheduledFlight(): List<ScheduledFlight>* :-

Shows all the details and status of all flights.

*deleteScheduledFlight(): void* :-

Removes a flight from the available flights.

1. **Passenger**: This class stores all the details of a passenger.

Attributes:

pnrNumber: Long

passengerName: String

passengerAge: Short

passengerUIN: Long

Luggage: Double

Methods: -

1. **Booking**: This class stores the details of a booking made by a particular userId. Every booking stores a list of passengers travelling in it as well as the flight details.

Attributes:

bookingId: Long

userId: Long

noOfPassengers: Short

passengerList: List<Passenger>

bookingDate: DateTime

ticketCost: Double

flight: Flight

Methods:

*addPassenger(Passenger): Passenger :-*

Adds a Passenger to the current booking. The Passengers are stored in the passengerList attribute of a booking. A passenger will be added only if a valid name is entered, the age of the passenger is less than 6, the UIN is a 12 digit number and the luggage is a floating point number less than 15 kg per passenger.

*viewPassenger(): List<Passenger>* :-

Returns a list of all the passengers present in a booking.

*updatePassenger(Passenger): Passenger* :-

Updates the details of added passengers to a booking.

*deletePassenger(): void* :-

Removes a passenger from a booking.

1. **ScheduledFlight**: This class stores a flight that is scheduled along with its schedule and the vacancy.

Attributes:

flight: Flight

schedule: Schedule

availableSeats: Short

Methods:

*updateSchedule(Schedule): Schedule* :-

Updates the schedule of the current flight.

*deleteSchedule(Schedule): void* :-

Removes the schedule from a flight. Thereby, making a flight unscheduled.

1. **Flight**: This class stores all the details of a flight.

Attributes:

flightNumber: Long

flightModel: String

carrierName: String

capacity: Short

Methods: -

1. **Schedule**: This class stores a flight schedule.

Attributes:

arrivalTime: DateTime

departureTime: DateTime

sourceAirport: Airport

destinationAirport: Airport

Methods:

*updateSourceAirport(Airport): Airport :*-

Changes the source airport.

*updateDestinationAirport(Airport): Airport* :-

Changes the destination airport.

1. **Airport**: This class stores the details of an airport.

Attributes:

airportName: String

airportCode: String

airportLocation: String

Methods: -

**Class Design View:**

The following diagram shows the attributes and methods of each class. Also, it represents the constraints imposed on the system. It represents the static view of the application and can be directly mapped to an Object-Oriented Programming language.