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:
   a. Create his user account.
   b. Login into the application.
   c. Check for available flights.
   d. Make a booking.
   e. View the bookings made.
   f. Cancel or modify a booking.

2. The administrator can:
   a. Login into the application.
   b. Add flight, schedule and route details.
   c. View the flight, schedule and route details.
   d. 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.
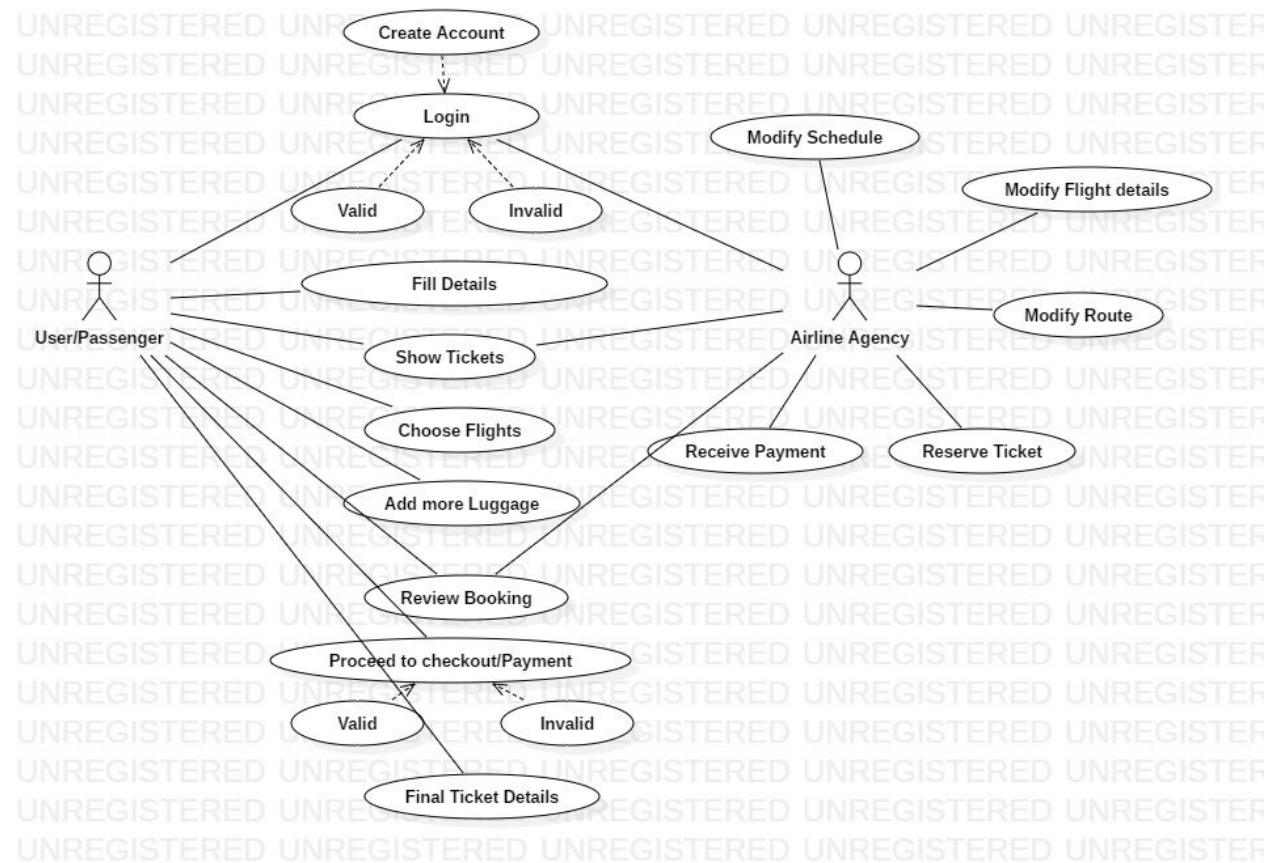
Validation:

1. The 'userPhone' should have an exact 10 digit number and the number should not start with zero.
2. Date and Time should be valid i.e date and time that has already elapsed shouldn't be entered
3. 'noOfPassenger' should always be less than equal to that of available seats.
4. The local part of the email should contain alphanumeric characters only. No special characters are to be present as the first character of the id.
5. The chosen airport's name should be present inside the Airport database.
6. The Unique Identification Number should be of 12 digits.

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.
5. Number of airports is fixed and stored in database.
6. All the flights are considered to be domestic.

The application will consist of the following classes, the utility of which have been described below as per their respective layers:

DTO Layer:

1. **User**: This class stores the user type (admin or the customer) and all user information.
    Attributes:
        userType: String.
        userId: BigInteger
        userName: String
        userPassword: Password
        userPhone: BigInteger
        userEmail: String
    Methods: –

2. **Passenger**: This class stores all the details of the travelling passenger.

     Attributes:
          pnrNumber: BigInteger
          passengerName: String
          passengerAge: Integer
          passengerUIN: BigInteger
          Luggage: Double
     Methods: –

3. **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: BigInteger
          userId: User
          bookingDate: Date
          passengerList: List<Passenger>
          ticketCost: BigDouble
          flight: Flight
          noOfPassengers: Integer
     Methods: –

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

     Attributes:
          flight: Flight
          availableSeats: Integer
          schedule: Schedule
     Methods: –

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

     Attributes:
          flightNumber: BigInteger
          flightModel: String
          carrierName: String
          seatCapacity: Integer
     Methods: –

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

        Attributes:
                sourceAirport: Airport
                destinationAirport: Airport
                arrivalTime: DateTime
                departureTime: DateTime
        Methods: -

7. **Airport**: This class stores the details of an airport.
        Attributes:
                airportName: String
                airportCode: String
                airportLocation: String
        Methods: -

Service Layer:

8. **UserServiceImpl:**
        Attributes: -
        Methods:
                addUser(User):User :-
                Adds a new user.

                viewUser(BigInteger):User :-
                Shows the details of a user identifiable by the user id.

                viewUser(): List<User> :-
                Shows the details of all users.

                updateUser(User):User :-
                Updates the details of a user.

                deleteUser(BigInteger):void
                Removes a user as per the user id.

                validateUser(User): void :-
                Validates the attributes of a user.

9. **BookingServiceImpl:**

Attributes: -
Methods:

addBooking(Booking):Booking :- Creates a new booking.

modifyBooking(Booking): Booking :- Modifies a previous booking. All information related to the booking except the booking id can be modified.

viewBooking(BigInteger): List<Booking> :- Retrieves a booking made by the user based on the booking id.

viewBooking(): List<Booking> :- Retrieves a list of all the bookings made.

deleteBooking(BigInteger): void :- Deletes a previous booking identifiable by the 'bookingId'.

validateBooking(Booking): void :- Validates the attributes of a booking.

validatePassenger(Passenger): void :- Validates the attributes of a passenger.

10. FlightServiceImpl:

Attributes: -
Methods:

addFlight(Flight): Flight :- Adds a new flight which can be scheduled.

modifyFlight(Flight): Flight :- Modify the details of a flight.

viewFlight(BigInteger): Flight :- Shows the details of a flight specified by the flight number.

viewFlight(): List<Flight> :- View the details of all flights.

deleteFlight(BigInteger): void :- Removes a flight.

validateFlight(Flight): void :- Validates the attributes of a flight.

11. **ScheduleFlightServicesImpl:**

Attributes: -
Methods:

> *scheduleFlight(ScheduledFlight): ScheduledFlight* :-
> Schedules a flight alongwith its timings, locations and capacity

> viewScheduledFlights(Airport, Airport, LocalDate): List<Scheduled Flight> :-
> Returns a list of flights between two airports on a specified date.

> viewScheduledFlights(BigInteger):Flight :-
> Returns a list of a scheduled flight identifiable by flight number.

> *viewScheduledFlight(): List<ScheduledFlight>* :-
> Shows all the details and status of all flights.

> *modifyScheduledFlight(Flight,Schedule, Integer): ScheduledFlight* :-
> Modifies the details of a scheduled flight.

> *deleteScheduledFlight(BigInteger): void* :-
> Removes a flight from the available flights.

> validateScheduledFlight(ScheduledFlight): void :-
> Validates the attributes of a scheduled Flight.

12. **AirportServiceImpl:**

Attributes: -
Methods :

> viewAirport(): List<Airport> :-
> Returns the list of all airports.

> viewAirport(String): Airport :-
> Returns the details of an airport identifiable by the airport code.

DAO Layer:

13. **UserDaoImpl:**

Attributes:

> userList: List<User>

Methods:

    addUser(User):User :-
    Adds a new user.

    viewUser(BigInteger):User :-
    Shows the details of a user identifiable by the user id.

    viewUser(): List<User> :-
    Shows the details of all users.

    updateUser(User):User :-
    Updates the details of a user.

    deleteUser(BigInteger):void
    Removes a user as per the user id.

14. BookingDaoImpl:

    Attributes:
        bookingList: List<Booking>

    Methods:

        addBooking(Booking):Booking :- Creates a new booking.

        modifyBooking(Booking): Booking :- Modifies a previous booking. All information related to the booking except the booking id can be modified.

        viewBooking(BigInteger): List<Booking> :- Retrieves a booking made by the user based on the booking id.

        viewBooking(): List<Booking> :- Retrieves a list of all the bookings made.

        deleteBooking(BigInteger): void :-
        Deletes a previous booking identifiable by the 'bookingId'.

15. FlightDaoImpl:

    Attributes:
        flightList: List<Flight>

Methods:

addFlight(Flight): Flight :-
Adds a new flight which can be scheduled.

modifyFlight(Flight): Flight :-
Modify the details of a flight.

viewFlight(BigInteger): Flight :-
Shows the details of a flight specified by the flight number.

viewFlight(): List<Flight> :-
View the details of all flights.

deleteFlight(BigInteger): void :-
Removes a flight.

16. ScheduledFlightDaoImpl:

Attributes:
scheduledFlightList: List<ScheduledFlight>

Methods:

scheduleFlight(ScheduledFlight): ScheduledFlight :-
Schedules a flight alongwith its timings, locations and capacity

viewScheduledFlights(Airport, Airport, LocalDate): List<Scheduled Flight> :-
Returns a list of flights between two airports on a specified date.

viewScheduledFlights(BigInteger):Flight :-
Returns a list of a scheduled flight identifiable by flight number.

viewScheduledFlight(): List<ScheduledFlight> :-
Shows all the details and status of all flights.

modifyScheduledFlight(Flight,Schedule,int): ScheduledFlight :-
Modifies the details of a scheduled flight.

deleteScheduledFlight(BigInteger): void :-
Removes a flight from the available flights.

17. AirportDaoImpl:

Attributes:
airportList: List<Airport>

Methods:

    viewAirport(): List<Airport> :-
    Returns the list of all airports.

    viewAirport(String): Airport :-
    Returns the details of an airport identifiable by the airport code.

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.

UserDaoImp

+userList: List<User>

+addUser(User): User
+showUser(): List<User>
+showUser(): User
+updateUser(User): User
+deleteUser(BigInteger): void

BookingDaoImpl

+bookingList: List<Booking>

+addBooking(Booking): Booking
+showBooking(): List<Booking>
+showBooking(BigInteger): Booking
+updateBooking(Booking): Booking
+deleteBooking(BigInteger): void

ScheduledFlightDao

+scheduledFlightList: List<ScheduledFlight>

+addScheduledFlight(ScheduledFlight): ScheduledFlight
+showScheduledFlight(Airport, Airport, LocalDate): List<Flight>
+showScheduledFlight(BigInteger): ScheduledFlight
+showScheduledFlight(): List<ScheduledFlight>
+updateScheduledFlight(Flight, Schedule, int): ScheduledFlight
+deleteScheduledFlight(BigInteger): void

+implements

+implements

+implements

UserDao

+addUser(User): User
+showUser(): List<User>
+showUser(): User
+updateUser(User): User
+deleteUser(BigInteger): void

BookingDao

+addBooking(Booking): Booking
+showBooking(): List<Booking>
+showBooking(BigInteger): Booking
+updateBooking(Booking): Booking
+deleteBooking(BigInteger): void

ScheduledFlightDao

+addScheduledFlight(ScheduledFlight): ScheduledFlight
+showScheduledFlight(Airport, Airport, LocalDate): List<Flight>
+showScheduledFlight(BigInteger): ScheduledFlight
+showScheduledFlight(): List<ScheduledFlight>
+updateScheduledFlight(Flight, Schedule, int): ScheduledFlight
+deleteScheduledFlight(BigInteger): void

FlightDaoImpl

+flightList: List<Flight>

+addFlight(Flight): Flight
+showFlight(): List<Flight>
+showFlight(BigInteger): Flight
+updateFlight(Flight): Flight
+deleteFlight(BigInteger): void

AirportDaoImpl

+airportList: List<Airport>

+showAirport(): List<Airport>
+showAirport(String): Airport

+implements

+implements

FlightDao

+addFlight(Flight): Flight
+showFlight(): List<Flight>
+showFlight(BigInteger): Flight
+updateFlight(Flight): Flight
+deleteFlight(BigInteger): void

AirportDao

+showAirport(): List<Airport>
+showAirport(String): Airport

12

UserDaoImp

+userList: List<User>

+addUser(User): User
+showUser(): List<User>
+showUser(): User
+updateUser(User): User
+removeUser(BigInteger): void

BookingDaoImpl

+bookingList: List<Booking>

+addBooking(Booking): Booking
+showBooking(): List<Booking>
+showBooking(BigInteger): Booking
+updateBooking(Booking): Booking
+removeBooking(BigInteger): void

ScheduledFlightDao

+scheduledFlightList: List<ScheduledFlight>

+addScheduledFlight(ScheduledFlight): ScheduledFlight
+showScheduledFlight(Airport, Airport, LocalDate): List<Flight>
+showScheduledFlight(BigInteger): ScheduledFlight
+showScheduledFlight(): List<ScheduledFlight>
+updateScheduledFlight(Flight, Schedule, int): ScheduledFlight
+removeScheduledFlight(BigInteger): void

+implements

+implements

+implements

UserDao

+addUser(User): User
+showUser(): List<User>
+showUser(): User
+updateUser(User): User
+removeUser(BigInteger): void

BookingDao

+addBooking(Booking): Booking
+showBooking(): List<Booking>
+showBooking(BigInteger): Booking
+updateBooking(Booking): Booking
+removeBooking(BigInteger): void

ScheduledFlightDao

+addScheduledFlight(ScheduledFlight): ScheduledFlight
+showScheduledFlight(Airport, Airport, LocalDate): List<Flight>
+showScheduledFlight(BigInteger): ScheduledFlight
+showScheduledFlight(): List<ScheduledFlight>
+updateScheduledFlight(Flight, Schedule, int): ScheduledFlight
+removeScheduledFlight(BigInteger): void

FlightDaoImpl

+flightList: List<Flight>

+addFlight(Flight): Flight
+showFlight(): List<Flight>
+showFlight(BigInteger): Flight
+updateFlight(Flight): Flight
+removeFlight(BigInteger): void

AirportDaoImpl

+airportList: List<Airport>

+showAirport(): List<Airport>
+showAirport(String): Airport

+implements

+implements

FlightDao

+addFlight(Flight): Flight
+showFlight(): List<Flight>
+showFlight(BigInteger): Flight
+updateFlight(Flight): Flight
+removeFlight(BigInteger): void

AirportDao

+showAirport(): List<Airport>
+showAirport(String): Airport

13