1. Title: Flight Management System

2.Project Statement:

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.

Scopes:

In scope:

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.

The customer can:

Create his user account.

Login into the application.

Check for available flights.

Make a booking.

View the bookings made.

Cancel or modify a booking.

The administrator can:

Login into the application.

Add flight, schedule and route details.

View the flight, schedule and route details.

Cancel or modify the flight, schedule and route details.

Out scope:

The following functionalities have not been covered under the application:

The application does not cover boarding pass generation and seating plans.

Third party applications like email & SMS integrations.

Payments are not yet accepted by the application.

3. **Modules to be implemented**

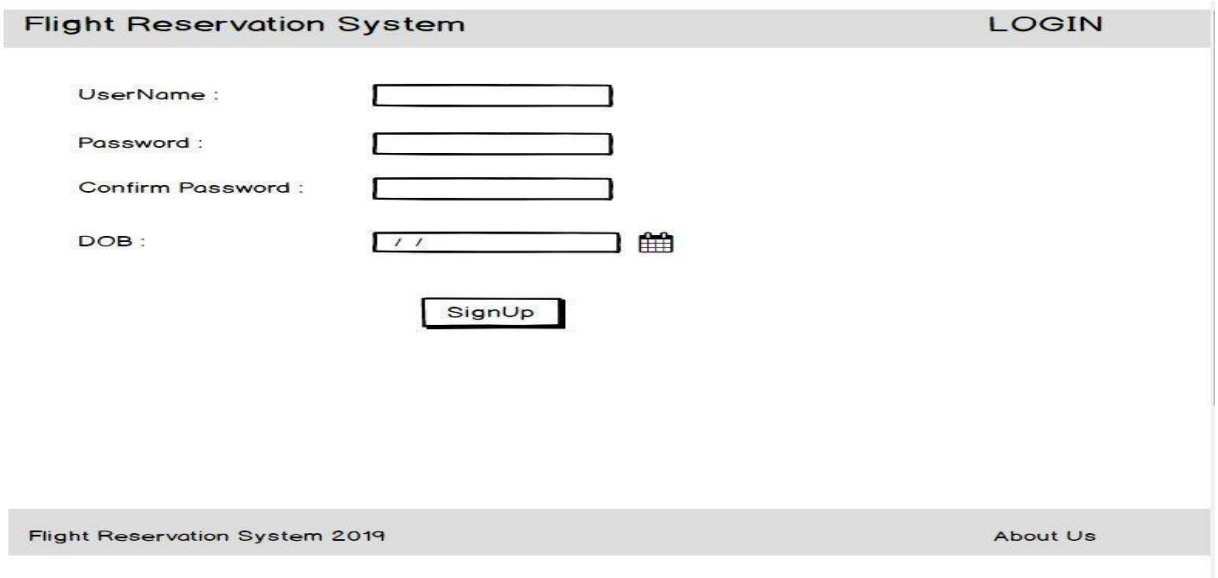
1. User Authentication and Registration
2. Booking
3. Passenger
4. Flight
5. Schedule flight
6. Airport / Airport Schedule

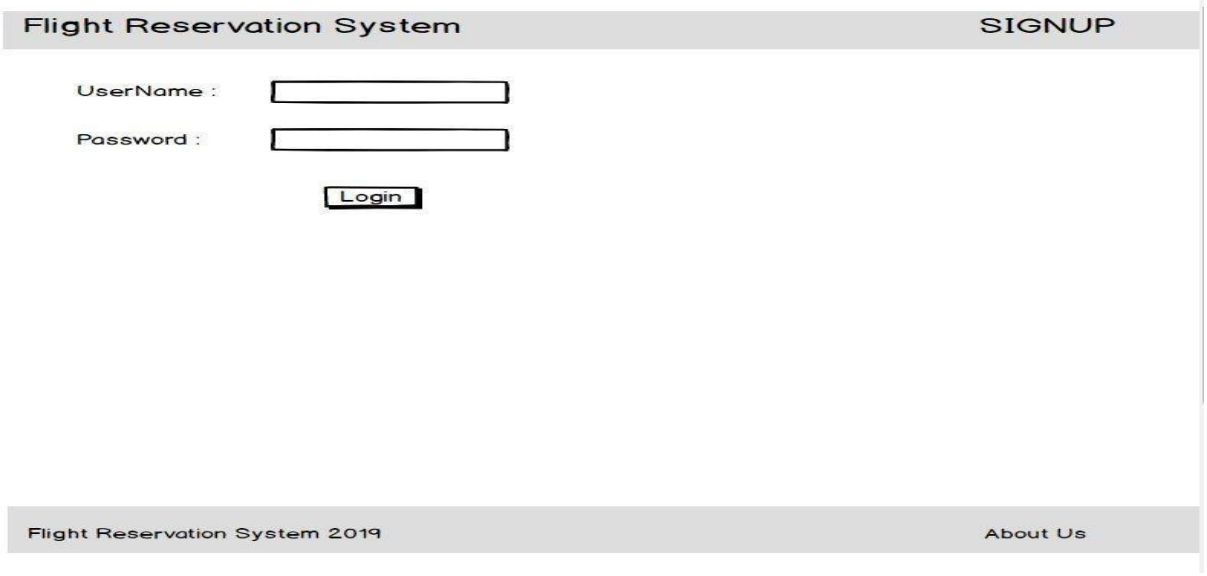
4. **Week-wise module implementation and high-level requirements with output screenshots**

Milestone 1: Weeks 1-3

Module 1: User Authentication and Registration

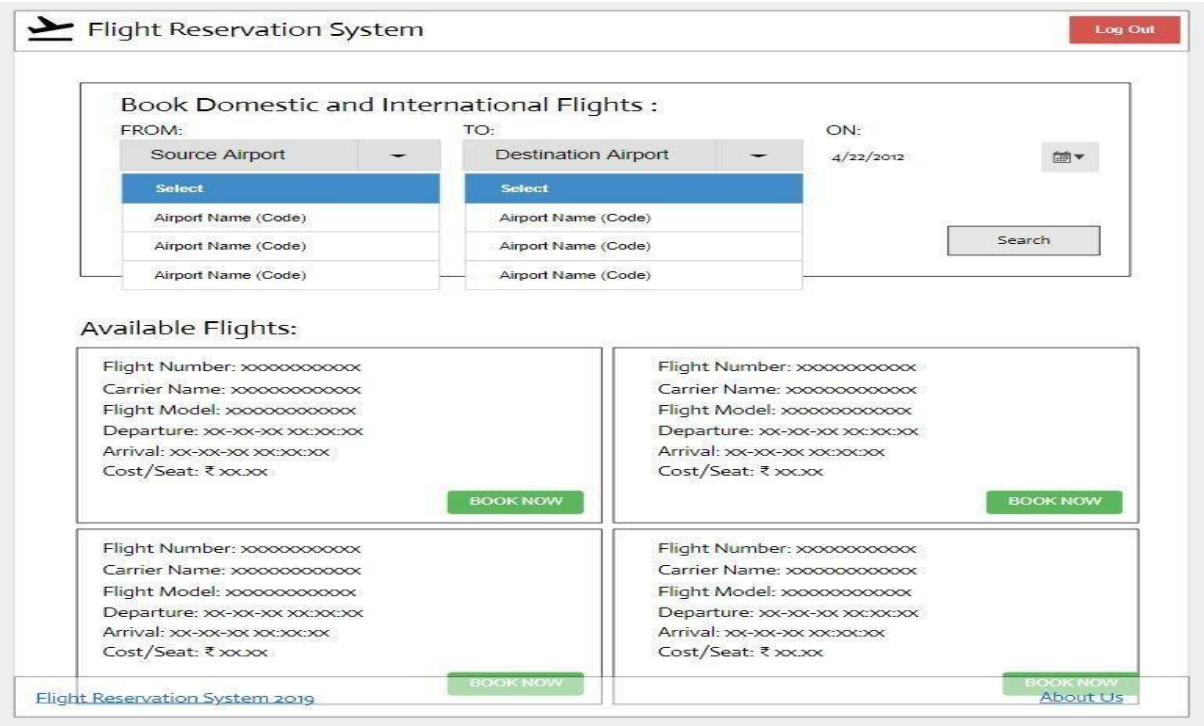
- Implement user registration functionality.
- Develop user login mechanism.
- Integrate email verification and password reset features.
- Output screenshoot:





Module 2: Airports and Flights

- Design a database schema for storing airport and flight information.
- Develop an interface for airports to set their flight availability.
- Implement backend functionality to manage and update airport and flight availability.
- **Output screenshot:**



Flight Number: xxxxxxxxxxxx
Carrier Name: xxxxxxxxxxxx
Flight Model: xxxxxxxxxxxx
Departure: xx-xx-xx xx:xx:xx
Arrival: xx-xx-xx xx:xx:xx
Cost/Seat: ₹ xx.xx

Flight Number: xxxxxxxxxxxx
Carrier Name: xxxxxxxxxxxx
Flight Model: xxxxxxxxxxxx
Departure: xx-xx-xx xx:xx:xx
Arrival: xx-xx-xx xx:xx:xx
Cost/Seat: ₹ xx.xx

Flight Number: xxxxxxxxxxxx
Carrier Name: xxxxxxxxxxxx
Flight Model: xxxxxxxxxxxx
Departure: xx-xx-xx xx:xx:xx
Arrival: xx-xx-xx xx:xx:xx
Cost/Seat: ₹ xx.xx

Flight Number: xxxxxxxxxxxx
Carrier Name: xxxxxxxxxxxx
Flight Model: xxxxxxxxxxxx
Departure: xx-xx-xx xx:xx:xx
Arrival: xx-xx-xx xx:xx:xx
Cost/Seat: ₹ xx.xx

Milestone 2: Weeks 4-5

Module 3: Flight booking system

- Design and develop the interface for flight booking.
- Implement functionality for users to select flights, preferred time slots, and source and destination airports.
- Integrate the scheduling interface with the backend for storing booking details.
- **Output screenshot**

Flight Reservation System LOGOUT

Add Schedule Flights:

Schedule Flight Id :

Source Airport :

Destination Airport :

Departure Time : / /

Arrival Time : / /

Ticket Cost :

Add

Flight Reservation System 2019 About Us

Flight Reservation System

Log Out

Booking Details:

Passenger Details:

	Passenger Name	Age	Gender	ID Number
	Passenger Name	Age	Gender	ID Number
	Passenger Name	Age	Gender	ID Number
	Passenger Name	Age	Gender	ID Number

*Maximum 4 passengers per booking

Billing Address: Address

Contact Number: Contact Number

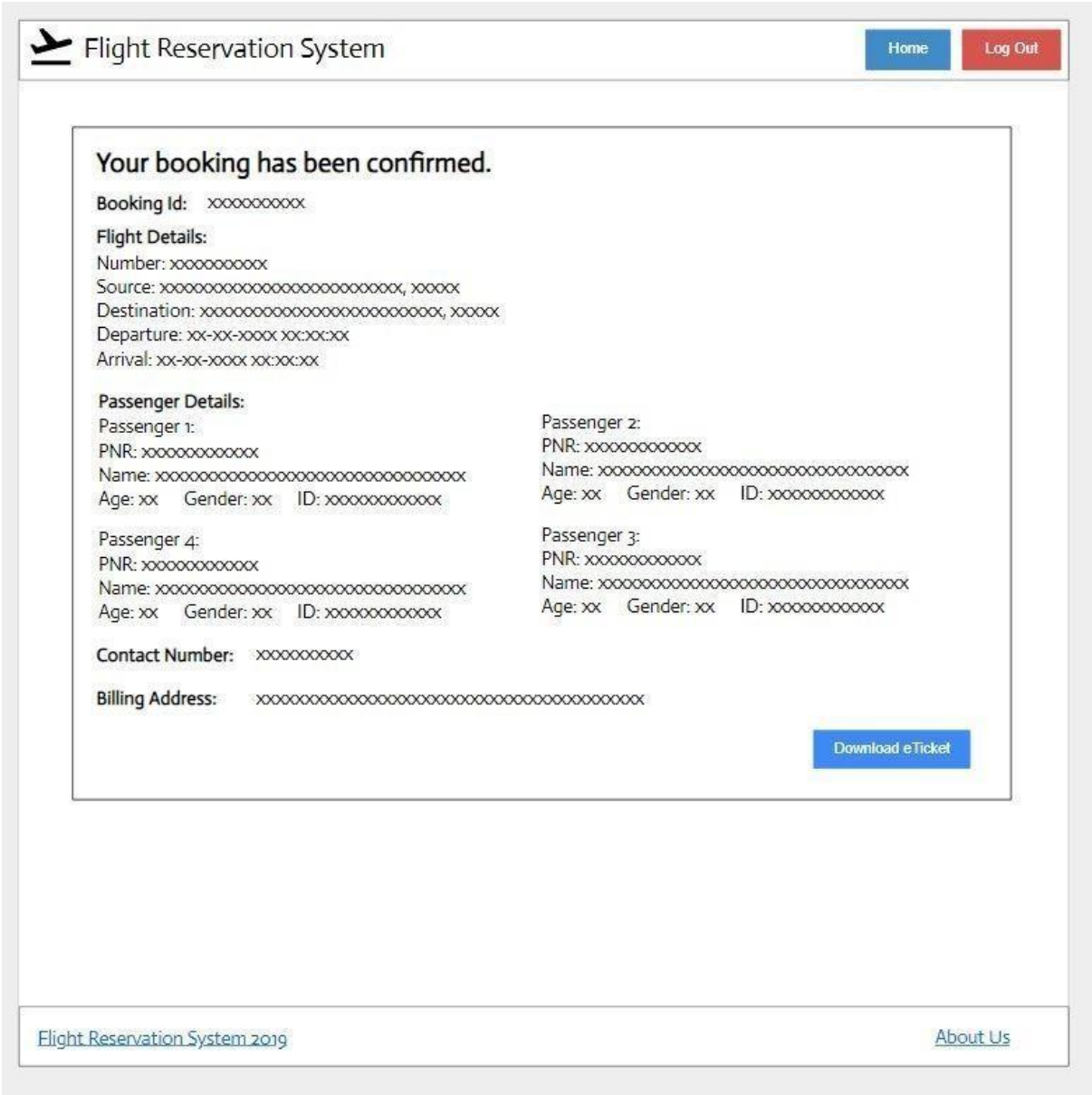
*Booking Confirmation will be sent to the above mentioned contact number.

I agree to the terms and conditions.

Confirm Booking

Flight Reservation System 2019

About Us



Milestone 3: Weeks 6-7

Module 4: Schedule Management

- Design database schema for storing flight schedule details.
- Implement backend functionality for scheduling bookings and preventing conflicts.
- Develop an interface for flights and passengers to view and manage schedules.
- **Output Screenshot:**

Flight Reservation System LOGOUT

Modify Schedule Flights:

Enter Schedule Flight Id : Search

Details :

Source Airport :

Destination Airport :

Departure Time : / /

Arrival Time : / /

Ticket Cost :

Modify

Flight Reservation System 2019 About Us

Flight Reservation System LOGOUT

Search Schedule Flights :

Enter Schedule Flight Id : Search

Details :

Source Airport :

Destination Airport :

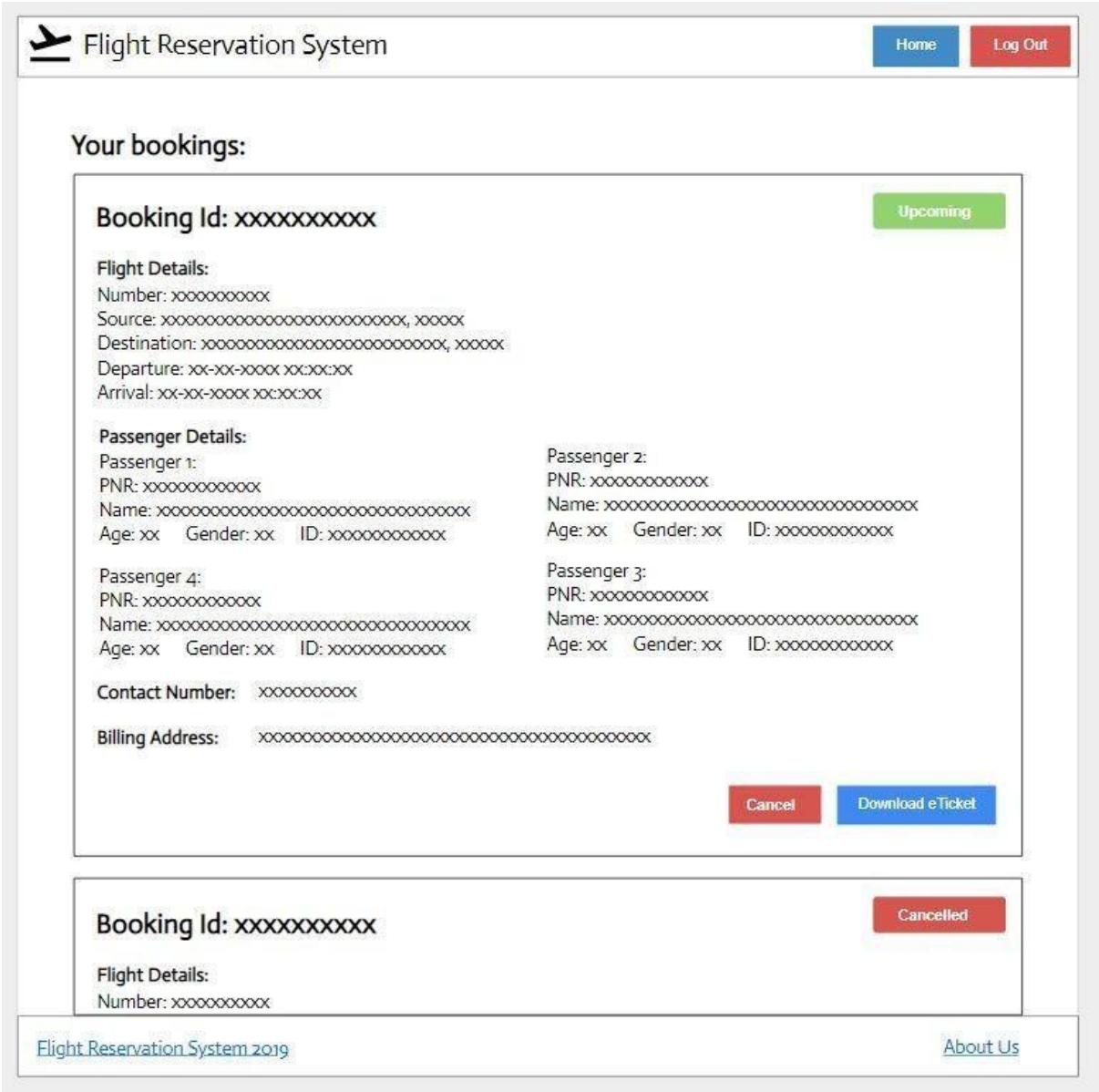
Departure Time : / /

Arrival Time : / /

Ticket Cost :

Flight Reservation System 2019 About Us

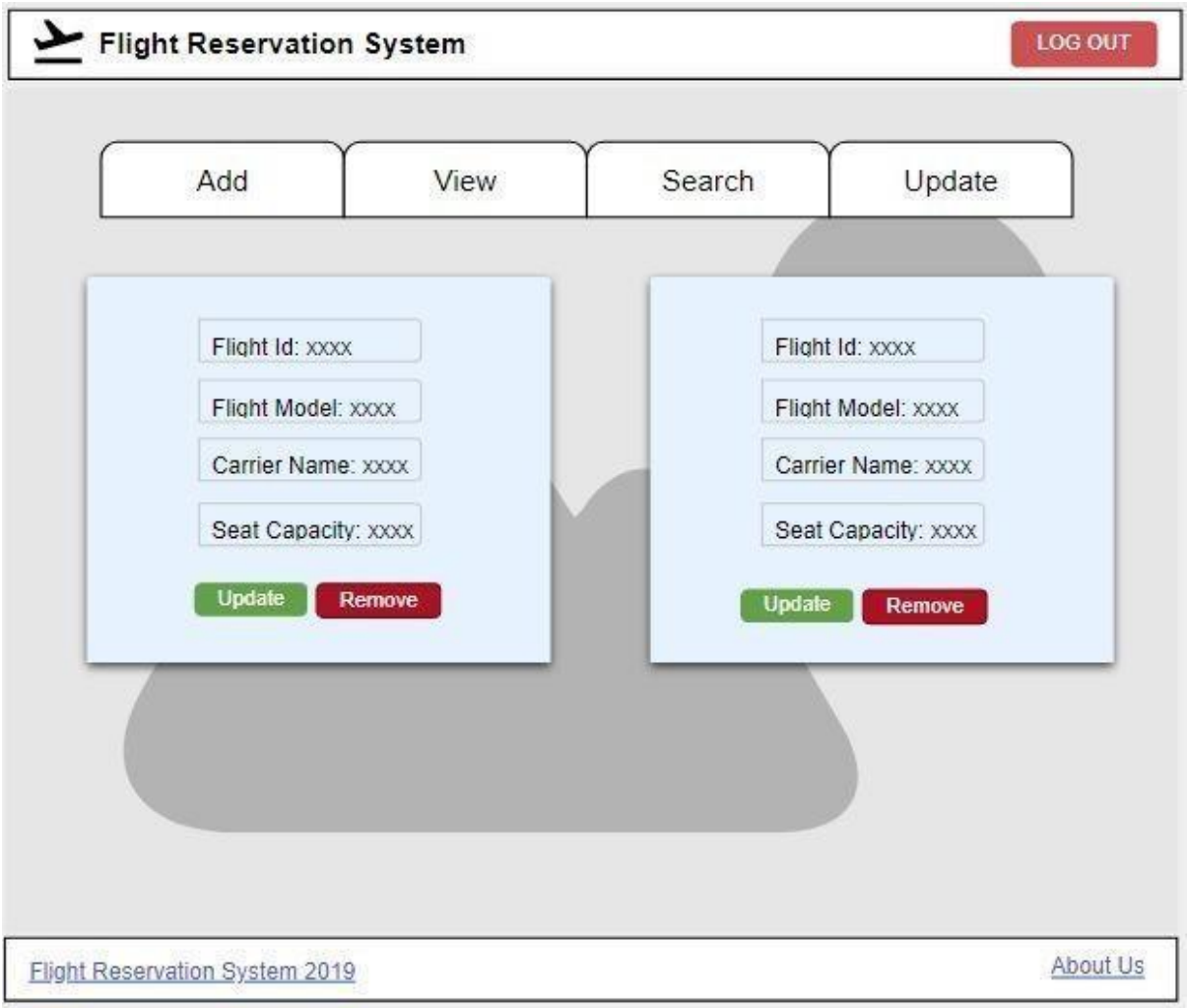

Flight Reservation System
LOG OUT
Welcome User!!!
Book A Flight
View Available Flights
Cancel A Booking
Flight Reservation System 2019
About Us



Milestone 4: Weeks 8-10

Module 5: Passenger Information Management

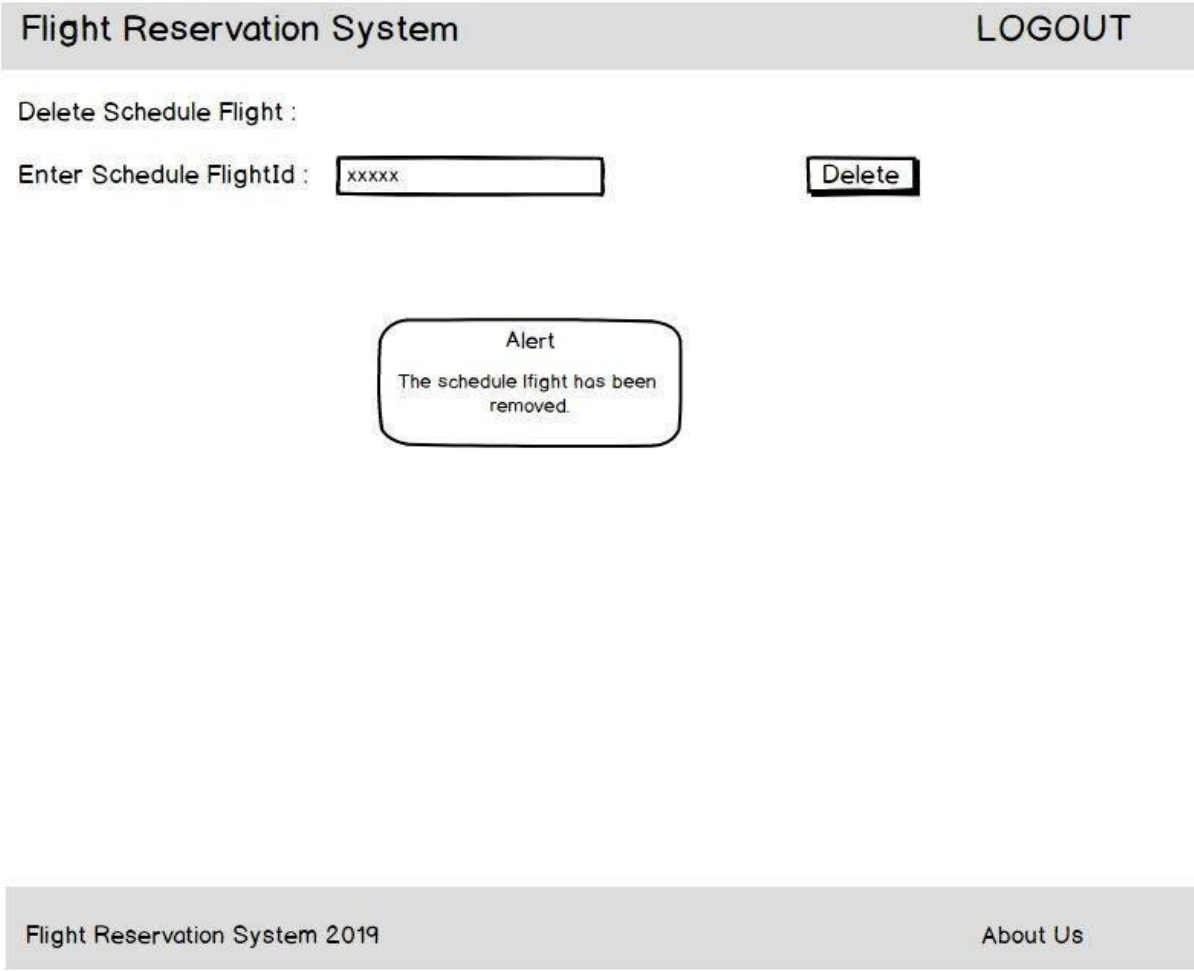
- Design database schema for storing passenger information securely.
- Develop functionality for passengers to provide and update their bookings.
- Implement backend logic for accessing and managing passenger information securely.
- **Output Screenshot:**


Flight Reservation System
LOG OUT
Add
View
Search
Update
Flight Id: xxxx
Flight Model: xxxx
Carrier Name: xxxx
Seat Capacity: xxxx
Update
Remove
Flight Id: xxxx
Flight Model: xxxx
Carrier Name: xxxx
Seat Capacity: xxxx
Update
Remove
Flight Reservation System 2019
About Us


Flight Reservation System
LOG OUT
Add
View
Search
Update
Flight Id: xxxx
Flight Id: xxxxx
Flight Model: xxxxx
Carrier Name: xxxxx
Seat Capacity: xxxxxx
Update
Remove
Flight Reservation System 2019
About Us

Module 6: Notification System

- Integrate email and/or SMS notification services.
- Develop triggers and notifications for appointment reminders, new bookings, and cancellations.
- Implement a system for handling and tracking notifications.
- **Output screenshot:**



Module 7: Admin Dashboard

- Design the layout and functionality of the admin dashboard.
- Implement user management features for admins to add, edit, and delete users.
- Develop modules for managing passengers, schedules, and passenger information.
- **Output Screenshot:**

Evaluation Criteria:

Milestone 1 Evaluation (Week 3):

- Completion of user authentication and registration.
- Successful implementation of flight availability management functionalities.

Milestone 2 Evaluation (Week 5):

- The flight scheduling interface is fully developed and integrated with the backend.

Milestone 3 Evaluation (Week 7):

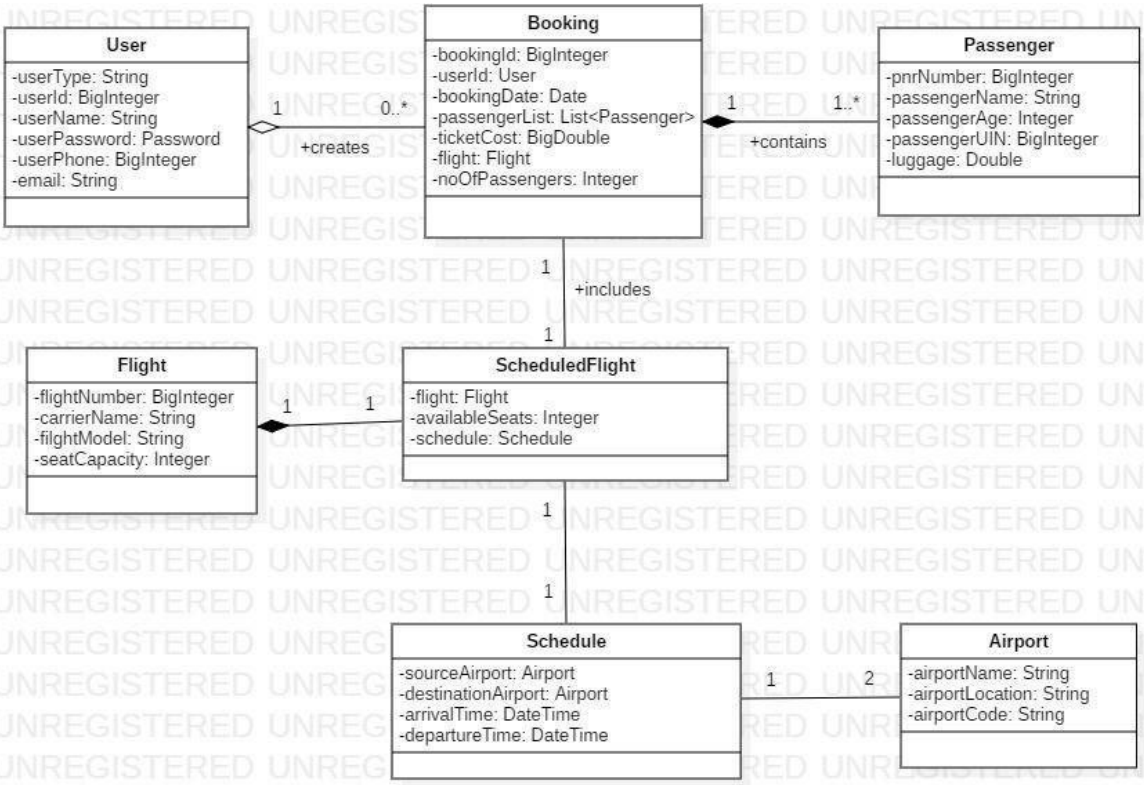
- The flight management system is operational, allowing for scheduling and viewing flight bookings.

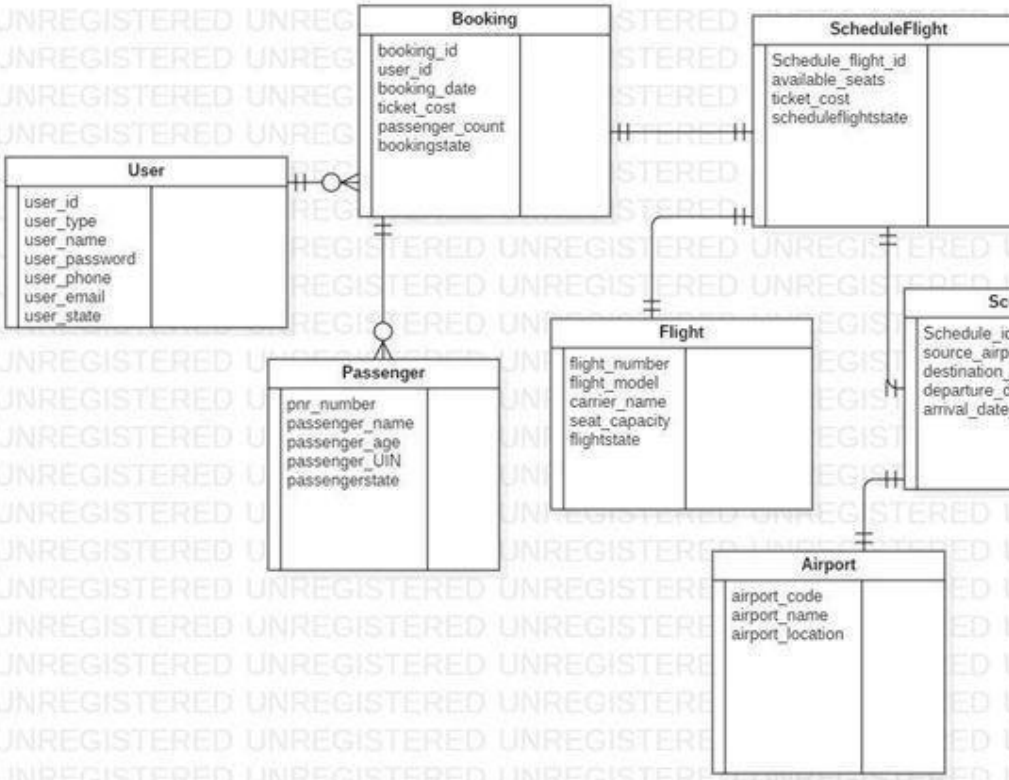
Milestone 4 Evaluation (Week 10):

- Passenger information management, notification system, and admin dashboard are fully implemented and functional.

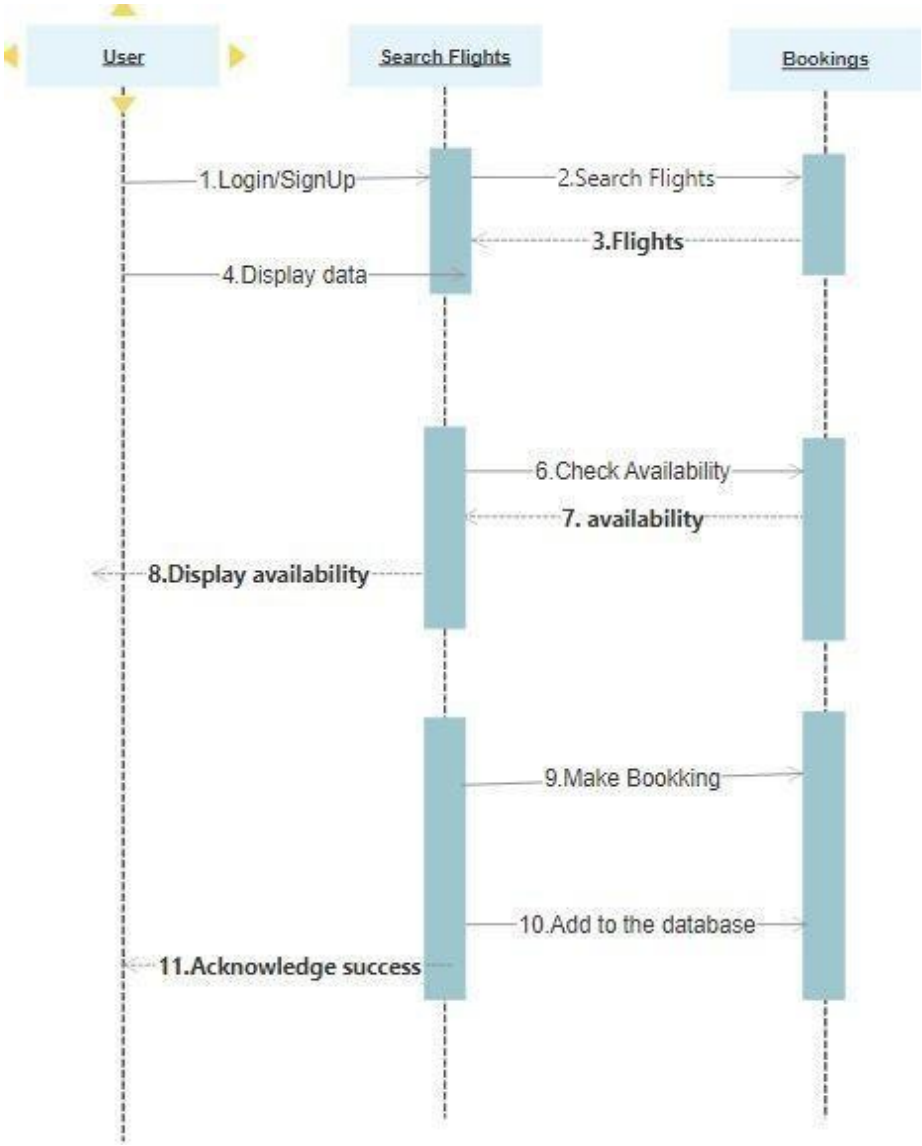
5. Design diagrams

Class Diagram: Entity classes

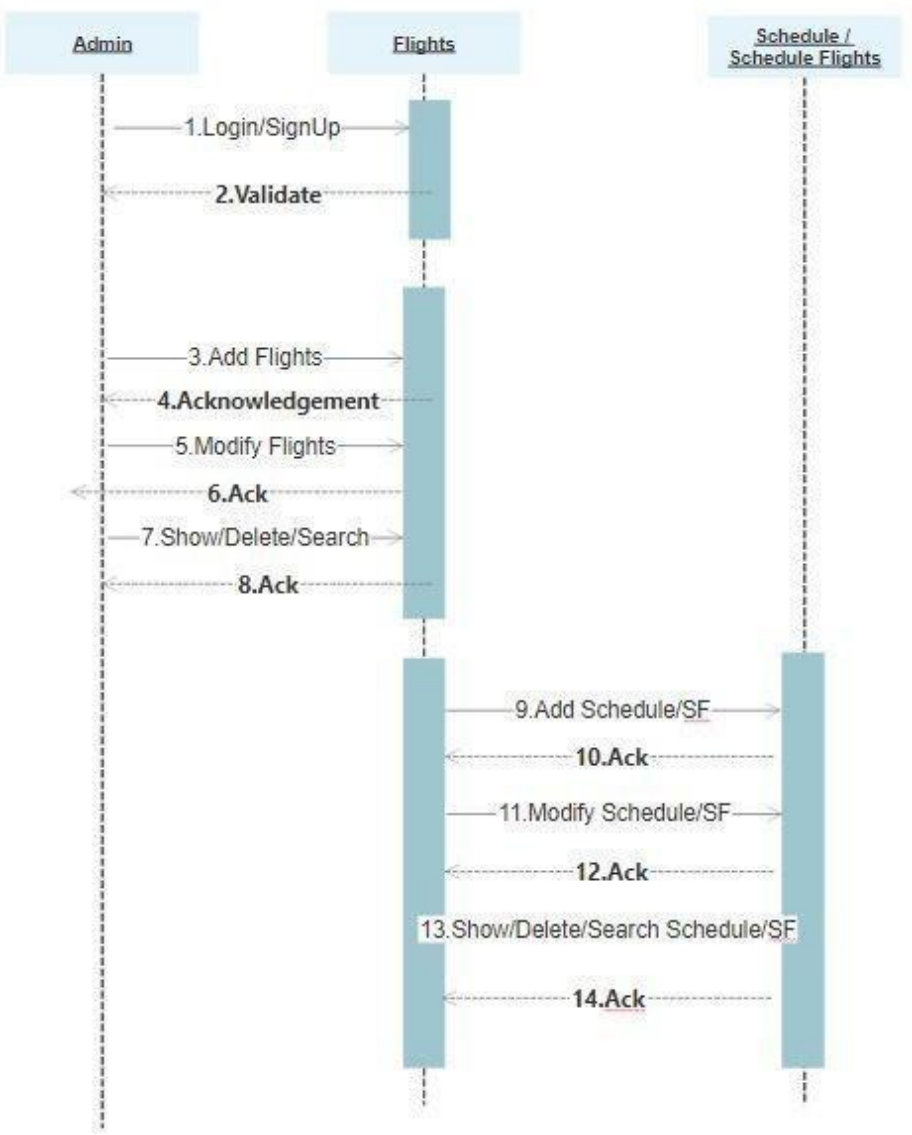




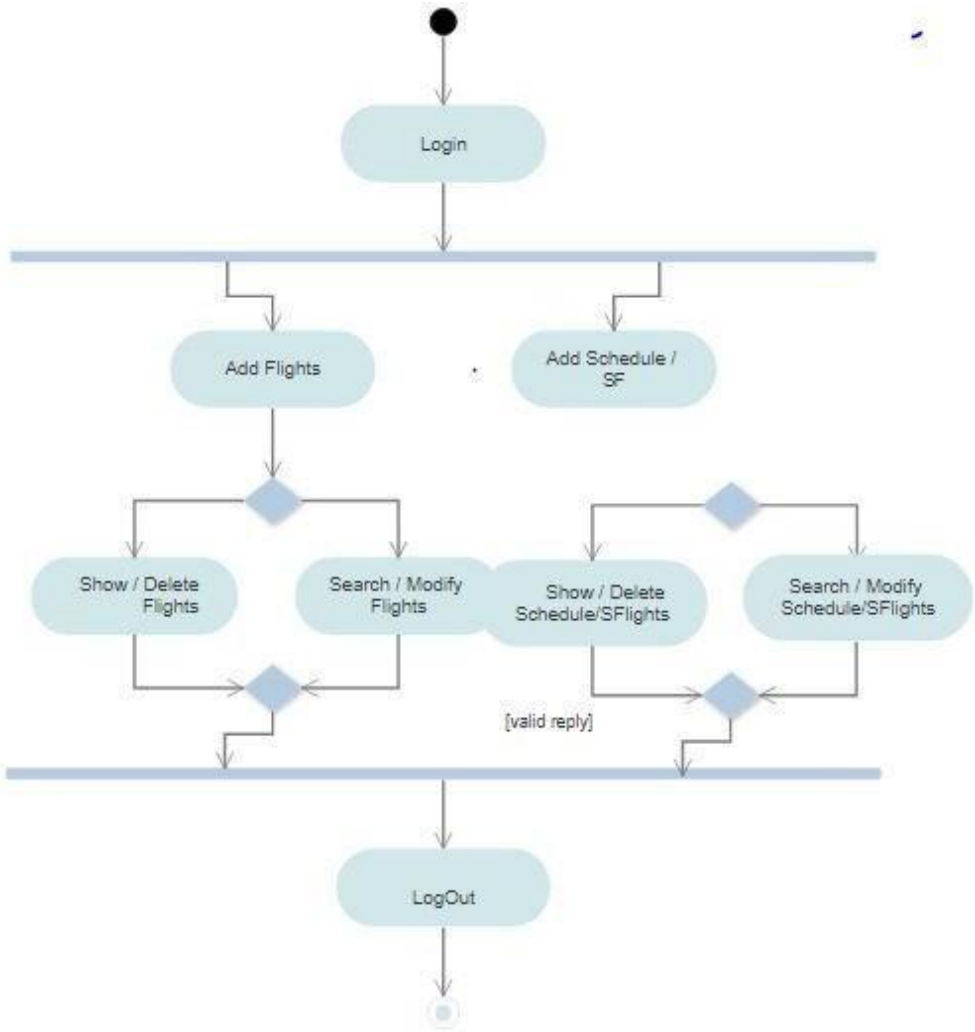
Sequence Diagram for User:



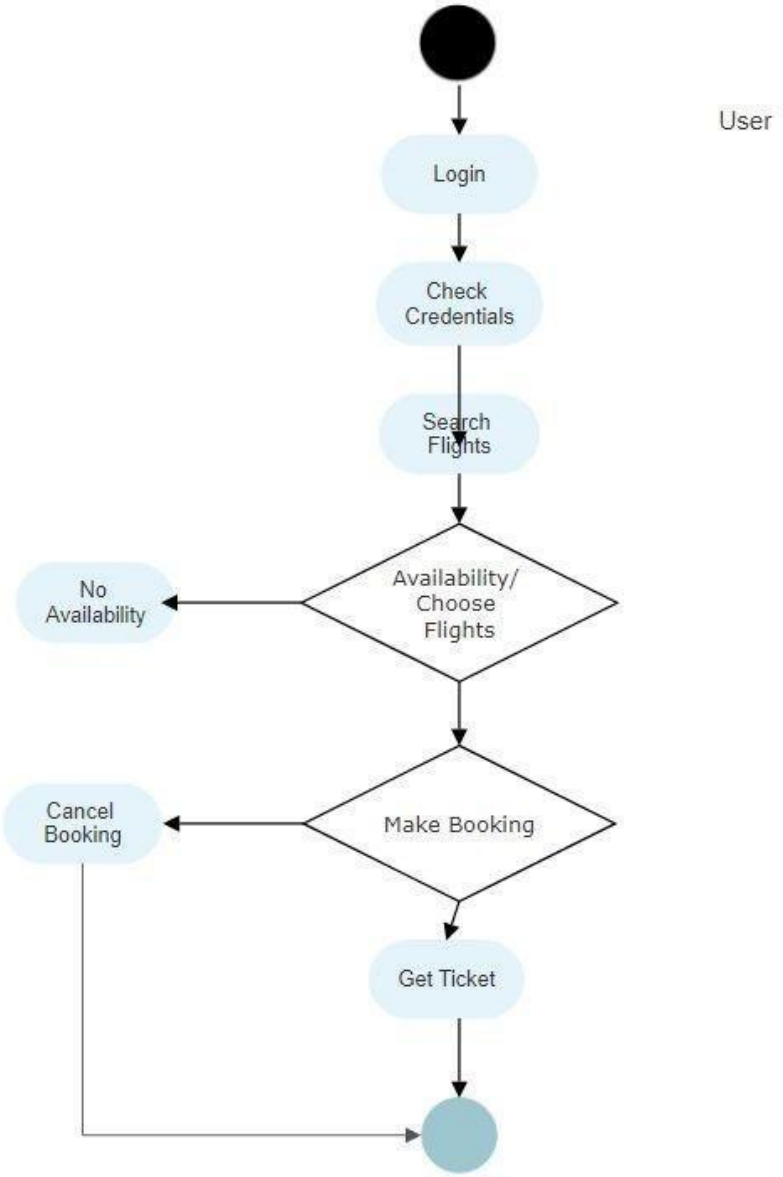
Sequence Diagram for Admin:



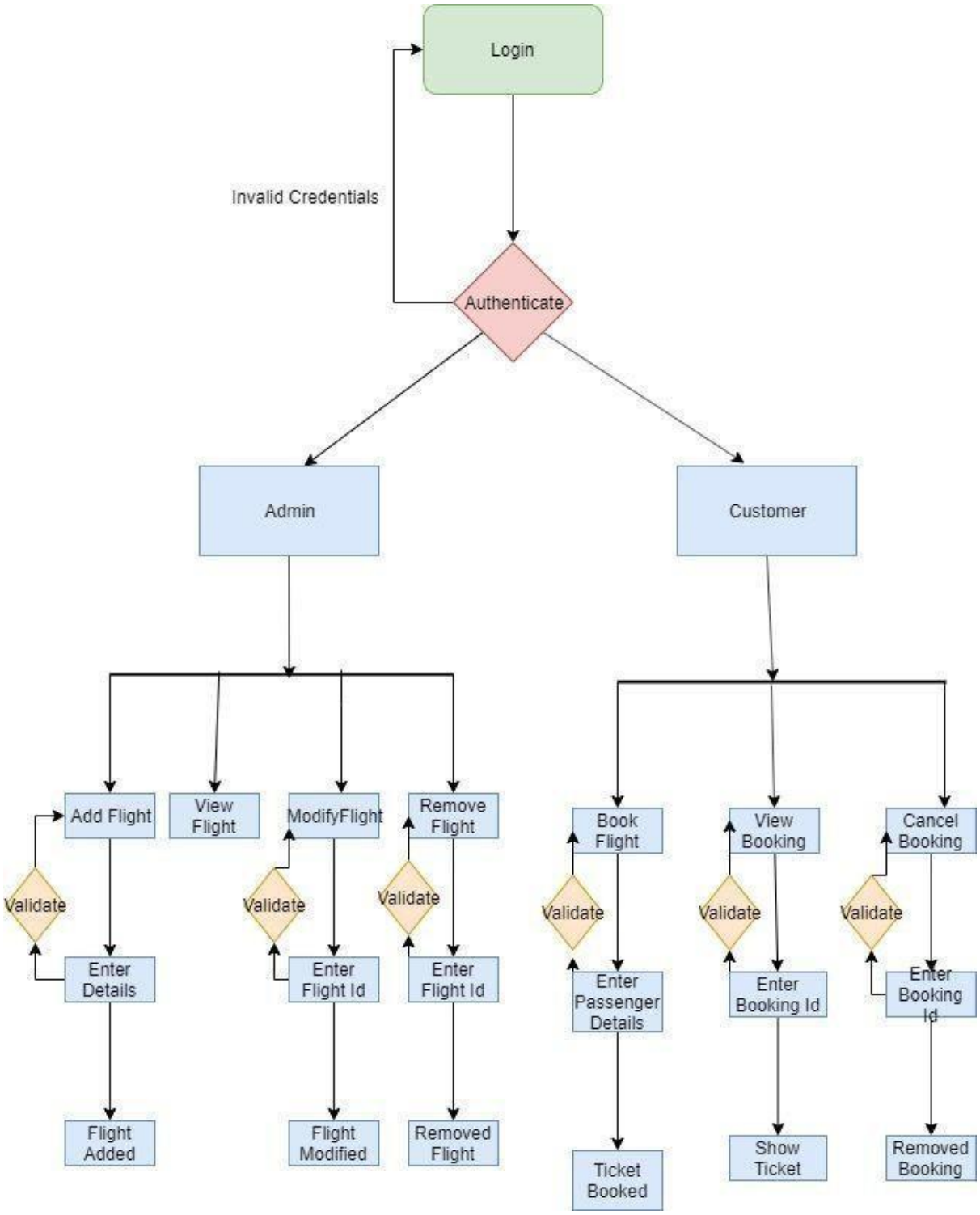
Activity Diagram for Admin:



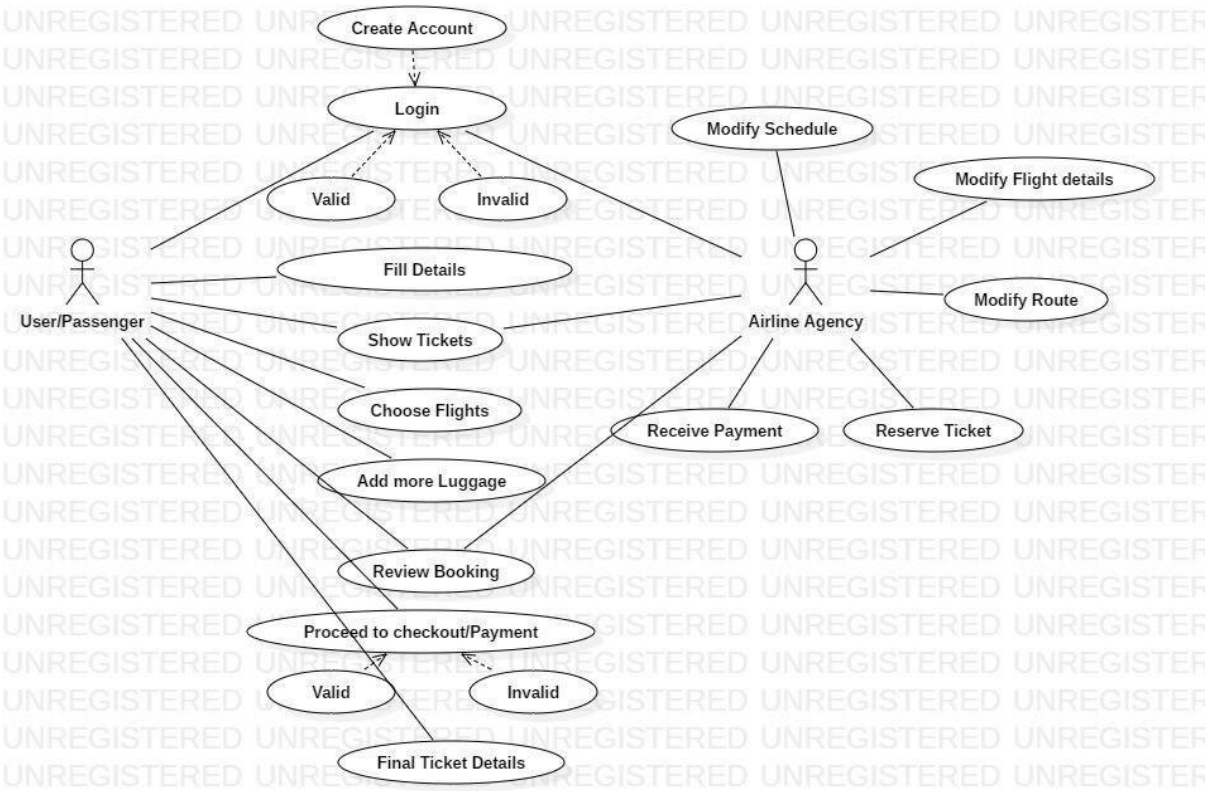
Activity Diagram for User:


User
Login
Check Credentials
Search Flights
Availability/ Choose Flights
No Availability
Make Booking
Cancel Booking
Get Ticket

Flow Chart:



Use Case Diagram:



Class and Method Description:

DTO Layer:

6. **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:BigI
   nteger
   userEmail:
   String

Methods: -

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

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

8. **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
noOfPassenger
s: Integer
Methods: -

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

Attributes:
flight: Flight
availableSeats
: Integer
schedule:
Schedule

Methods: -

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

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

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

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

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

Service Layer:

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.

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.

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.

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.

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:

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):Us
er :- Updates the
details of a user.

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

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'.

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.

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.

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.

Validations:

13. The ‘userPhone’ should have an exact 10 digit number and the number should not start with zero.
14. Date and Time should be valid i.e date and time that has already elapsed shouldn’t be entered
15. ‘noOfPassenger’ should always be less than equal to that of available seats.
16. 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.
17. The chosen airport’s name should be present inside the Airport database.
18. The Unique Identification Number should be of 12 digits.

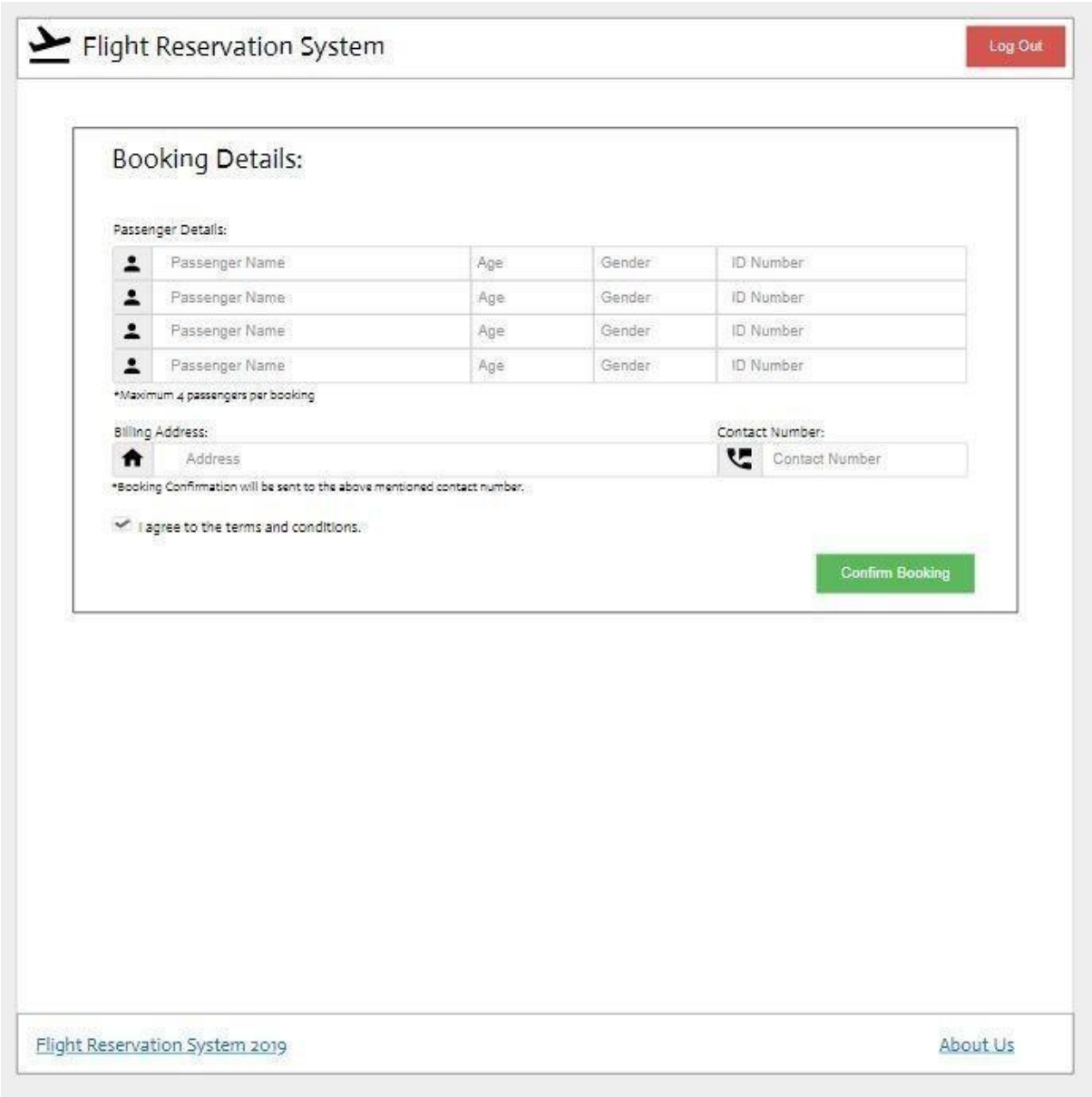
Assumptions:

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

19. Administrator and customer are both Users. They are differentiated by a variable ‘userType’ in the User class.
20. 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.
21. All flights are direct flights.
22. No flight gets cancelled.
23. Number of airports is fixed and stored in database.
24. All the flights are considered to be dom

Output Screenshots:

1. Passenger Dashboard



2. Admin Dashboard


Flight Reservation System
LOG OUT
Welcome Admin!!!
Add A Flight
Schedule A Flight
Search A Flight
Search Scheduled Flight
View Flights Available
View Scheduled Flight
Flight Reservation System 2019
About Us