ELEC 6312: Model Driven Software Engineering

Project Title

Airline Reservation System

Group No 7 -DELIVERABLE -3

Submitted By

Avika Lohany	40152495
Mamaleshwar Kovi Gowri Kumar	40126008
Rupinder Kaur	40151935
Satya Srinivas Aritakula	40120245
Vijay Gopal Reddy Badduri	40151211

Submitted To
Dr. Hamou Wahab-Lhadj

Concordia University

Faculty of Engineering and Computer Science

Department of Electrical and Computer Engineering

Table of Contents

List of Figures

1.PROJECT OVERVIEW

1.1 DESCRIPTION OF THE SYSTEM

The Airline Reservation System is a business-critical application which allows passengers to make reservations for most airlines based on the availability of several airline tickets for both domestic and international routes at the convenience of the user. The system includes airline fare tariffs, reservations, airline schedules, ticket records and the transaction history of a customer. The Airline Reservation System enables the user to search for an airline ticket based on their preference of destination, timing, and ticket fare. The Airline Reservation System includes two portals of which one portal will be accessible by the administrator and the other portal will be accessible to the customer registered onto the airline reservation system.

1.2 POTENTIAL CUSTOMERS

The Airline Reservation System is suitable for users who have the need to travel but do not want to go through the hassle of booking their flight tickets from the airline office. The system is completely online enabling the user to book a flight ticket based on their convenience of timing, destination, and ticket fare. As the system includes two portals for both Admin and passenger, so a passenger will be provided with a unique Customer ID upon registration and authentication through the system.

1.3 LIST OF PRIMARY FEATURES

The main features of our application are safety of usage, information hiding, flexibility and the ability to support multiple users at once. This is further understood from the points below:

- **User Registration***:* This feature allows the user to register with the airline reservation system as an admin or a traveler. The system will then verify the identification provided by the user

- **Flight Registration***:* This feature is limited to the administrator, allowing them to include, cancel or modify flights or flight details based on the airline schedule

- **Flight Reservation***:* This feature allows the user to search for an available flight ticket based on their preference for the timing, ticket fare and the destination. This feature allows the user to purchase the ticket by providing their billing details and finally confirming the ticket purchase

- **Traveller Itinerary***:* This feature is unique to each user and it displays the user'stravel history including any ongoing trips which the user will still have access to modify

- **Authorization:** This feature allocates certain access of the airline reservation system to the traveler whereas it provides full access to the administrator

1.4 REQUIREMENTS

The table below summarizes the functional and non-functional requirements of the airline booking application.

Story ID	User story	Functional requirement	Nonfunctional requirement
1	As user/admin, I want to login to the system	Authenticate user/admin for every login attempt	Have a hassle- free login and registration process
2	As a developer, I want to store the user details	Store user details (login/registration) in the database	------
3	I want to be able to view filtered results	Return related selections from the database to the user	Be able to query the different options available
4	As a developer, I want to store the selection details for future use	Store the user selection details	------
5	I want to make a payment and complete the transaction	Provide platform to complete the business transactions	Make payment with payment option of desired choice i.e. Credit/Debit card etc.

1.4.1 Other non-functional requirements:

- **Availability:** ensuring that most functionalities the application are available 24x7
- **Scalability:** ensuring that a greater number of users can access the application without any load
- **Security:** ensuring that there is no data breaching to the user credentials

1.5 EXPECTED USER INTERFACE

Given below is the expected view of the homepage of the website:

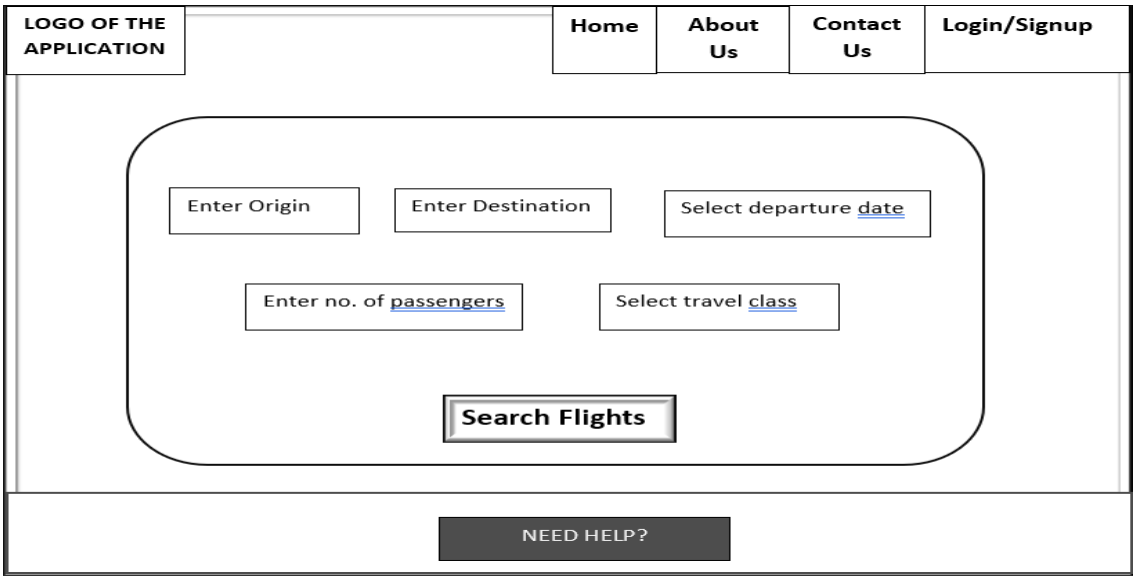

Figure 1 Expected User Interface: Website Homepage

The navigation bar has the following options:

- **Logo of the application:** this contains the icon/logo of the Airline reservation system. It would also be used to direct the control back to the home page from any other webpage for the website

- **Home:** like the logo, this tab will also be sued to bring the control to the homepage

- **About Us:** on clicking here, the user will be directed to a webpage showing the details of the company providing the services for the Airlines reservation system

- **Contact Us:** it navigates the user to the webpage showing the ways by which the he/she can communicate with the airlines company via phone call or email

- **Login/Signup:** To book a flight, a user must register with the company. So, a new user can register, or a registered user can login for more services provided by the website after landing onto the Login/Signup page

To look for the list of all the available flights from one city to other, one can enter the details and check the estimated fare as well. The footer of the homepage contains a link, "**Need help"**, to see the list of frequently asked questions regarding the flight booking or to mail the company for queries, this hyperlink directs the control to another webpage showing the above-mentioned details.

2. DESIGN OF THE APPLICATION

2.1 USE CASE DIAGRAM

From the use case diagram below, the different user interfaces can be understood.

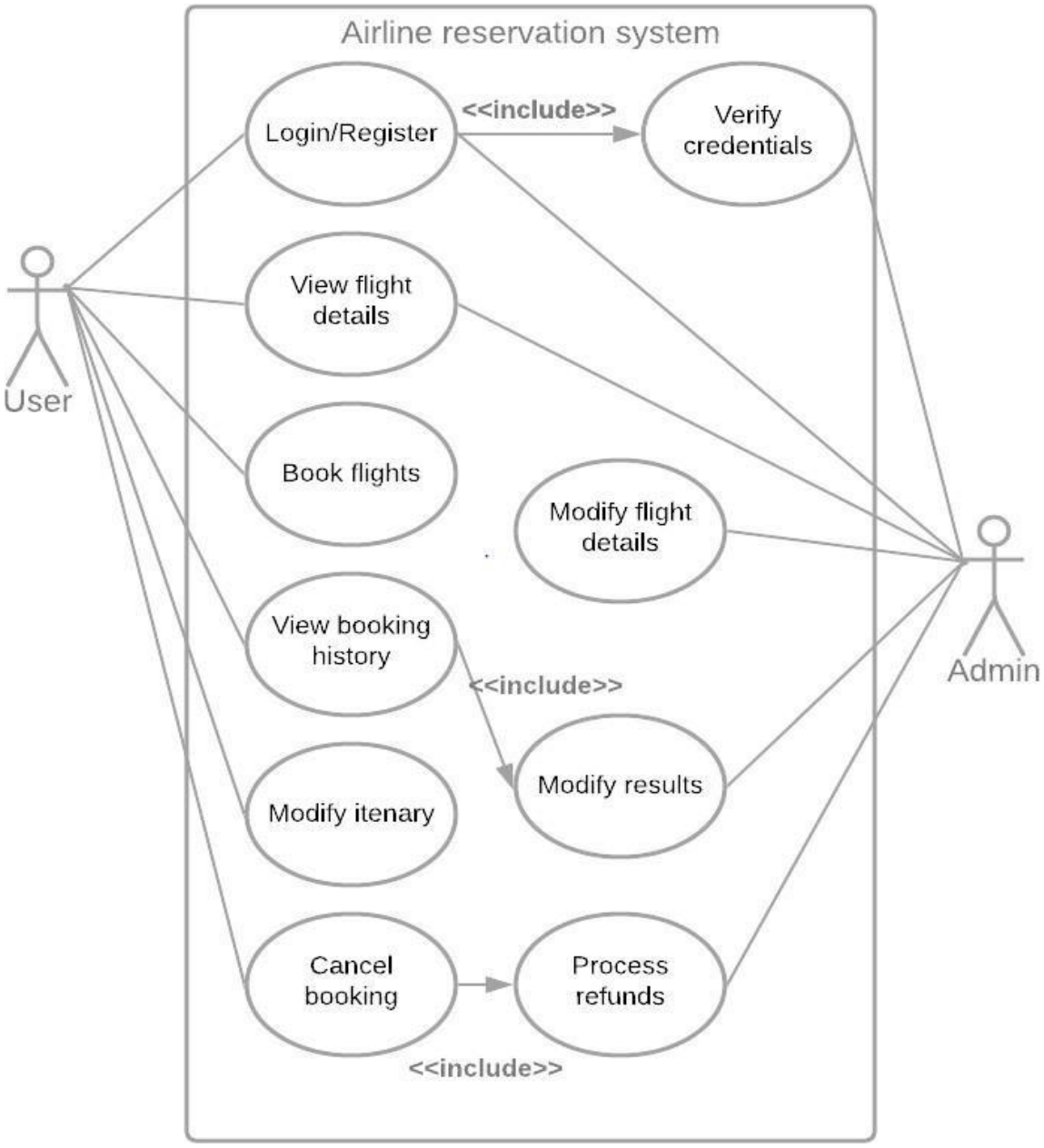

Figure 2 Use Case Diagram

2.2 CLASS DIAGRAM

The class diagram displayed below depicts the structure of the flight booking management system. It shows all the classes required for modeling the application along with the relationship between the objects of these classes.

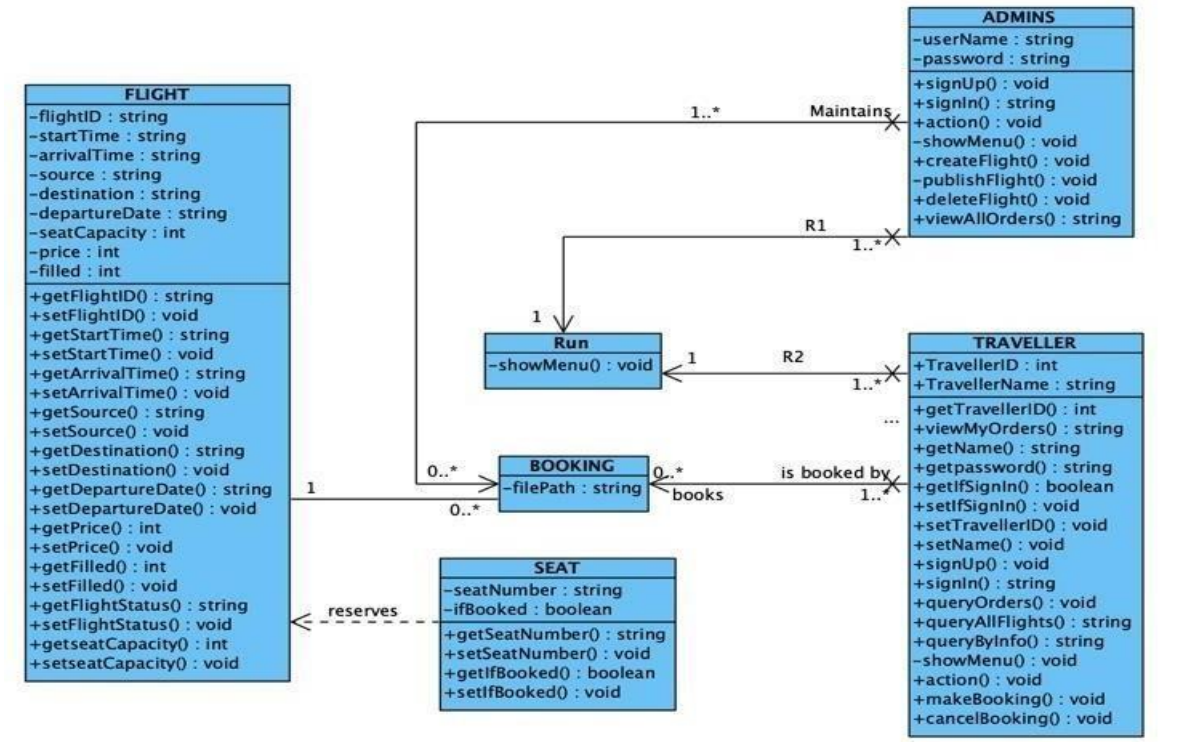

Figure 3 Class Diagram

2.3 OCL CONSTRAINTS

OCL constraints add precision to the UML models created for an application.
Following are some of the OCL constraints written and later implemented in the source code of the Flight reservation system:

1. The flights must have unique flight numbers

   Context Flight

   Inv: allInstances->forAll(f1, f2: Flight | f1<>f2 implies f1.flightID<>f2.flightID)

2. Cancelling a flight by the traveller

   Context Traveller::cancelBooking(String Travellername): void

   Pre: self.Booking.Flight->includes(Travellername)
   Post: self.Booking.Flight->excludes(Travellername)

3. Each flight has a limited number of seats i.e. 60.

   Context Flight

   Inv: self.seatCapacity-> size()<=60

4. No same origin and destination location for a flight.

   Context Flight

   Inv: allInstances->forall(f1:Flight| f1.source<>f1.destination)
   Context Flight

   Inv: self.source->reject(f:String| f=self.destination)

5. Only admins can publish the flight.

   Context Admins

   Pre: self.Booking.Flight.FlightStatus =’UNPUBLISHED’

   Post:self.Booking.Flight.FlightStatus =’AVAILABLE’

6. Admin can only create flights which do not already exist.

   Context Admins::createFlight(): void

   Pre: (self.Booking.Flight.flightStatus = ’UNPUBLISHED’)

   AND(self.Booking.Flight.allInstances->forAll(f1,f2:Flight|f1<>f2implies

   f1.flightID<>f2.flightID)AND

   (self.Booking.Flight.getsource()<>self.Booking.Flight.getdestination())

   Post: self.Booking.Flight->flightStatus = ’AVAILABLE’

7. Flight arrival time should not be after departure time.

   Context Booking

   Inv: self.Flight->forAll(f:Flight| f.startTime<f.arrivalTime)

8. Every user should have a unique user ID

   Context Traveller

   Inv: allInstances->forAll(t1,t2: Travellers| t1<>t2 implies (a1.travellerID <>a2.travellerID)

9. Traveller can only access the bookings belonging to them.

   Context Traveller::viewMyOrders()

   Inv: self.Booking.filepath->select(self.TravellerName)

10. The same traveller cannot have more than one booking for the same flight.

    Context Traveller

    Inv:self.Traveller.Flight->

    forAll(f1,f2:Flight|(f1<>f2 AND f1.flightNumber=f2.flightNumber) implies

    f1.departureDate <>f2.departureDate AND

    f1.departureTime <>f2.departureTime)

11. The flight booking is only added if it does not already exist.

    Contex Flight::add(Booking:b): void

    pre: self.Flight->excludes(b)

    post: self.Flight->includes(b)

12. The flight booking can only be cancelled if it does not already exist.

    Context Flight::cancel (Booking: b): void

    pre: self.Flight->includes(b)

    post: self.Flight->excludes(b)

2.4 STATE DIAGRAMS

State diagrams reflect action of an object of a class based on any stimuli i.e. state or event.

Following are some of the possible state diagrams showing the behavior of the objects of the classes created for the Flight reservation system.

2.4.1 State diagram for Run class:

This is how the transition will be from one state to another in the class containing the main ().

In the main (), a user is displayed with various options to choose from and use various functionalities of the application.

i. Upon selecting value zero, the user exists from the menu in main().

ii. If the user chooses to sign up as an admin, he can select option 1. They are requested to enterthe name and password. This record is then written inside the admin. CSV file and the user goes back choose another option from the menu.

iii. If a user wishes to sign up as a Traveller, he selects option 2 and requested to enter the name and desired password. The record from people.CSV file is searched to check is there is a repetition with the same name and password combination or not. If not, assign a new row in the CSV file with name, password and a TravellerID.

iv. To sign in as admin, enter the name and password and get authenticated after checkingthe admin.CSV file for the combination of name and password entered by the user. If the user is an authentic admin, then he is prompted to a menu with functionalities assigned to an admin only. If there is no record inside the file, the user goes back to menu.

v. To sign in as a Traveller, enter the name and password and get authenticated after checking the people.CSV file for the combination of name and password entered by the user. If the user is an authentic Traveller, then he is prompted to a menu with functionalities assigned to a Traveller only. If there is no record inside the file, the user goes back to menu.

vi. To view a list of flights based on source and destination without signing in, the user is displayed the content of file containing the list of all published flights. If there is no record inside the file, the user goes back to menu.

vii. If the user does not select any value from 0 to 5, then he is prompted to select a valid option.

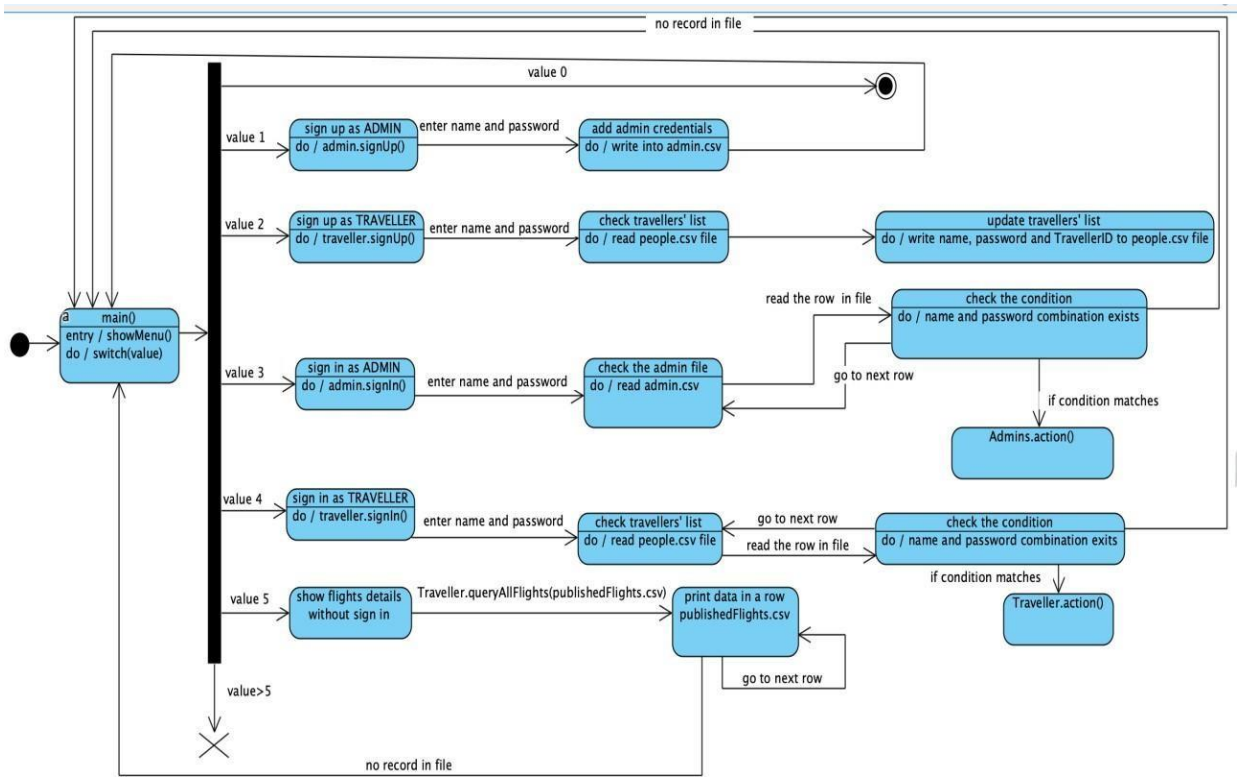

Figure 4 State Diagram for Run Class

2.4.2 State diagram for controls that an Admin has:

Being an admin of the application, a user has various functionalities. After signing in as an admin, a user is displayed a menu with multiple values to be entered.

i. To exit and not choose any value from the menu, enter value 0 and reach the final state.

ii. To generate a new flight, the admin enters the FlightID. If there already exists a flight with that ID,the admin is asked to enter another flight ID. If not, an object of flight is created with all the details entered by the admin and the state is set to UNPUBLISHED.

iii. If the admin wished to display a list of all the orders, he checks the record in orders.CSV file. If there is no record in the file, he is prompted with a message that there is no flight. Or else, the details of all the flights in the file is displayed to him.

iv. In order to delete a particular flight details from the record, admin enters the FlightID of the respective flight and checks if it exists in the publishedFlights.CSV file. If yes, then thedetails are removed. If not, then after checking the file till the end, the admin reaches the final state.

v. In order to publish the flights to be available for booking, admin is displayed with a list of flights from publishFlights.CSV file. He writes the details of a flight he wants to publish,

and the status of the flight is changed to AVAILABLE.

vi. If an admin does not choose any option out of the one specified in the menu, he is prompted to select a valid option.

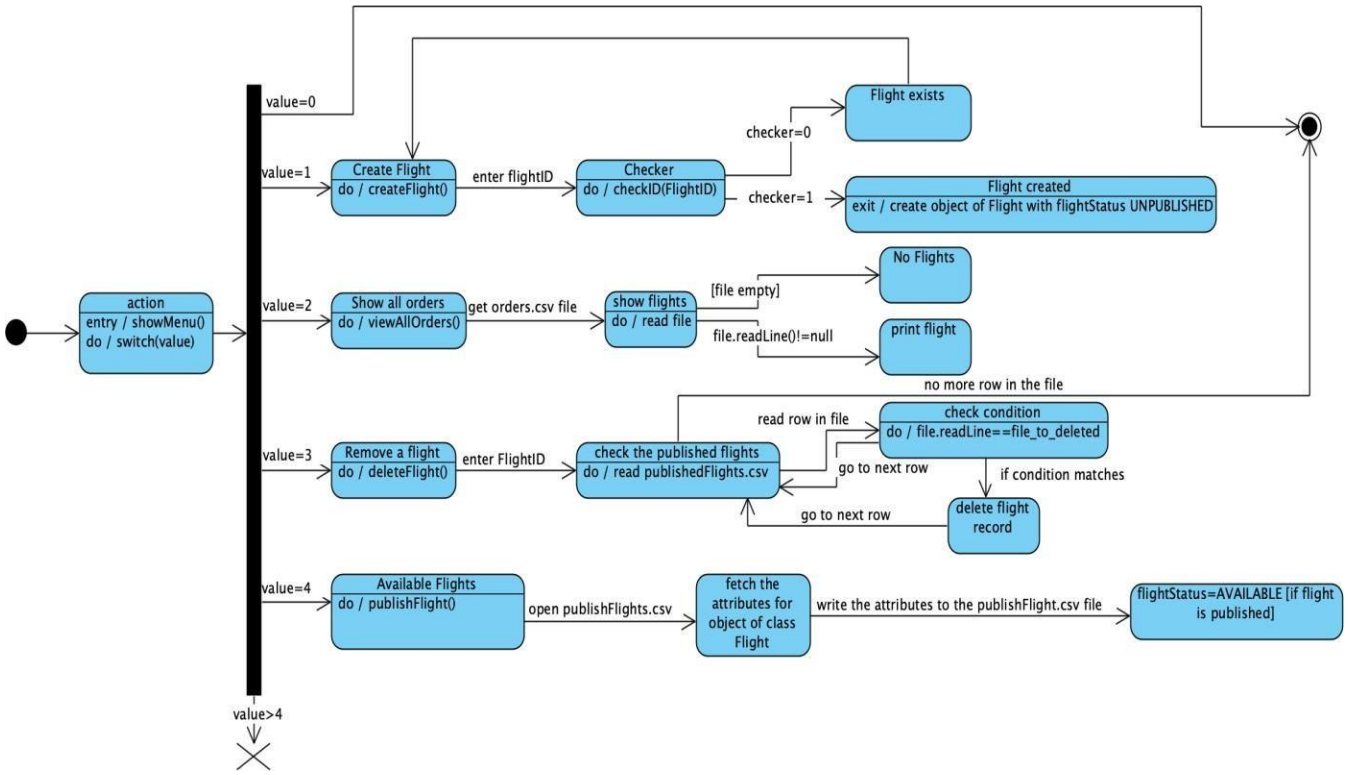

Figure 5 State Diagram for Admin Class

2.4.3 State diagram depicting controls that a Traveller has:

After signing in as a Traveller, the user is displayed a menu after getting triggered by the action().

i. In this menu, if the user selects option 0, he exits the menu and thus reaches the final state.

ii. Upon choosing option 1, he can query the flights based on the source and destination he enters. If he enters the same place as source and destination, he is prompted to review and enter the locations for source and destination. But, if the source and destination are different, then the application checks in the flights file if there is a flight available with the respective source and destination or not. And prompts message if such a flight with required source and destination does not exist.

iii. Then, there is another option to reserve a flight based on the Flight ID entered by the user. This is done by the make Booking (). The user enters the card details for payment to confirm the booking. A user whose booking is confirmed can either cancel/unsubscribe a

booking and could also view and query all the booking made by him.

iv. Upon choosing the option 3 in the menu a user can delete the flight booked by him. The details of the booking are removed from the record.

v. Another functionality that a user can have as a Traveller is that he can query all the bookings made by him. A list of all the flight booking done by the user is displayed from the people.CSV file. He can then exit from the current option and reach the final state.

vi. If a user does not choose any option out of the one specified in the menu, he is prompted to select a valid option.

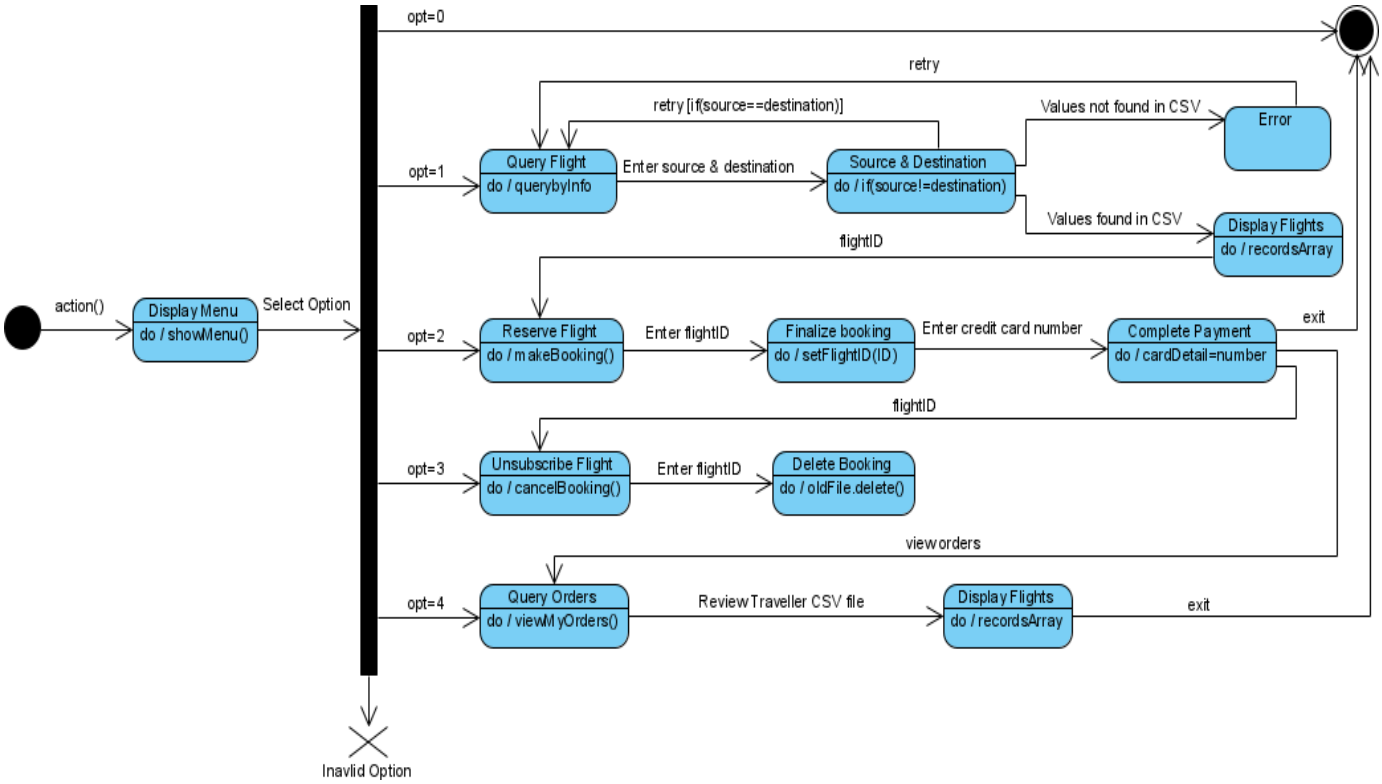

Figure 6 State Diagram for Traveller Class

3. REFERENCES

[1] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/
[2] http://etutorials.org/Programming/UML

4. APPENDIX 1

Zip file for the source code is submitted separately.

5. APPENDIX 2

The CSV files used for the storage of user credentials are:

1.admins.CSV

Name	Password

2. orders.CSV

Name	FlightID	Card	Date

3. people.CSV

Name	Password	ID

4. publishedflights.CSV

ID	Start	End	Source	Destination	Date	Capacity	Price	Filled