UNIVERSITY OF CALGARY

44/50

Very Good

ENSF 480 Term Project Design Document

Group 19

Members:

Chun Lok Chan - 30129645

Jordan Torske - 30142873

Mohamad Hussein - 30145507

Logan Nightingale - 30141334

Part A: System Analysis

A.1 Description

A flight reservation web application designed to streamline the flight reservation process for various stakeholders, including users (tourism agents and airline passengers), airline agents, flight attendants, and system administrators. The system is developed to facilitate management of flight information, seat selection, payments, cancellations, and overall flight operations.

User Module:

a. Browse Available Flights
b. Select Desired Flight
c. Browse Seat Map
d. Select Seat
e. Select Insurance
f. Make Payment
g. Receive Ticket and Receipt
h. Cancel Flight

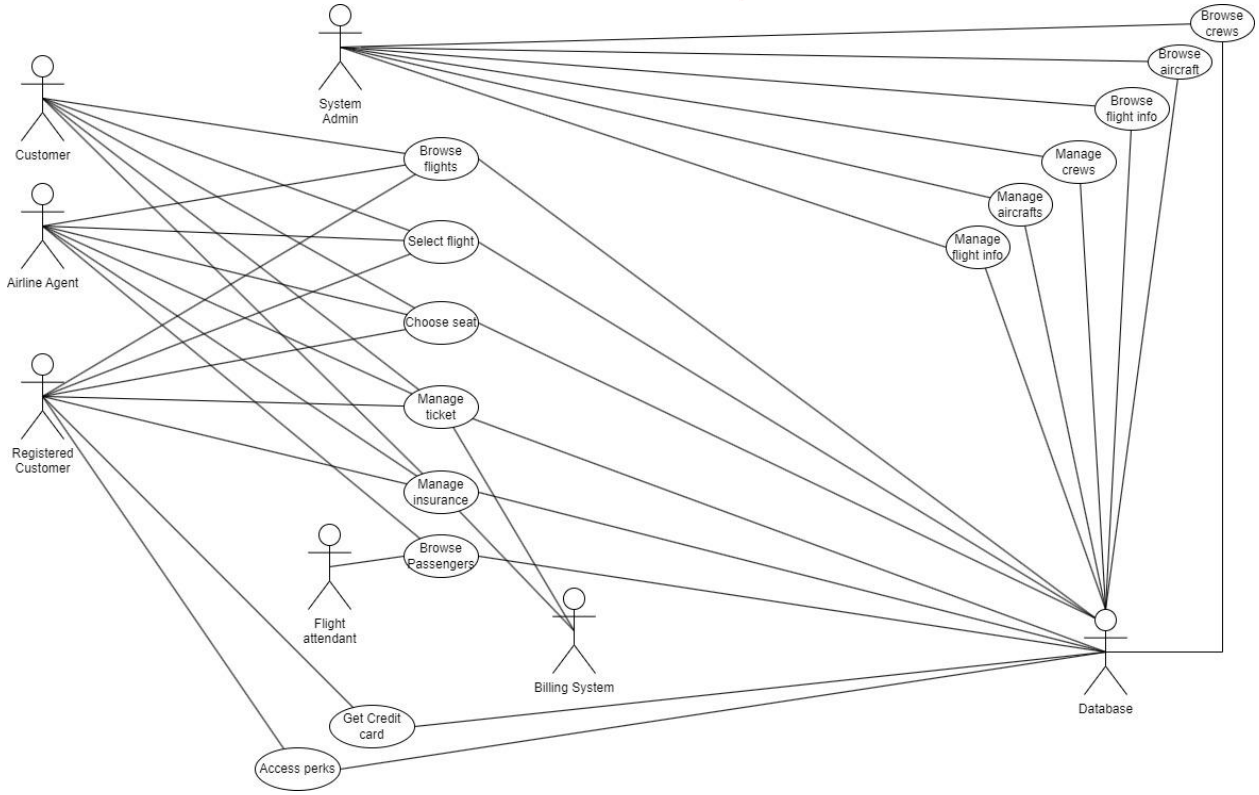
Airline Agent and Flight Attendant Module:

a. Browse List of Passengers
b. All function in User Moule

System Admin Module:

a. Manage Flight Information
b. Manage Crew Information
c. Manage Aircraft Information
d. Manage Flight Destinations
e. Manage Users

A.2 Use-Case diagram:



A.3 Use-Case scenarios:

Browse flights

The user logs in and selects destination B. The system then searches the database for all flights to that destination and returns a list for the user to browse. The user is presented with information about each flight such as origin, departure time and arrival time as well as the number of available ordinary, comfort and business class seats.

Identified candidate objects:

- Flight
- Destination
- Origin
- Time
- Seat
- User

Select flight

Having browsed the flight list and found a flight that suits their needs, the user selects it and is presented with the info about their selected flight and asked to confirm that they would like to proceed with the booking of this flight.

Identified candidate objects:

- Flight

Choose seat

After the user has selected their desired flight, the user is then asked to choose the type of seat they would like (regular, comfort or business class). Once this selection has been made, the database retrieves the information about all available seats of that type on that flight. This information is then presented graphically and the user is asked to choose a seat. Once they have made their selection, the database logs that seat as occupied and the user then proceeds to finalize their ticket purchase.

Identified candidate objects:

- Flight

- Seat
- User
- Ticket

Manage ticket

The user logs into the web application and selects “Manage tickets.” The system pulls the list of tickets for that specific customer from the database, returning a list that the user can manage. The user selects their ticket for flight #001. The application pulls all the relevant information for managing that ticket from the database.

Identified candidate objects:

- Flight
- Ticket
- User
- Passport

Manage insurance

The user has successfully logged in and is in the process of purchasing a ticket. After a user has selected their seat, the application pulls current insurance options for flight #001 tickets. The user then selects the “decline insurance” option, and the databse stores this selection along with the rest of the flight ticket information upon ticket purchase completion.

Identified candidate objects:

- Flight
- Insurance
- User

Browse Passengers

After a flight attendant successfully logs in, and is granted authentication, they can select to view the passenger list for flight #001 which they are registered to serve on. The database pulls the passenger list for flight #001, and displays it to the flight attendant.

Identified candidate objects:

- Flight
- Passenger
- Crew Members

Manage flight info

There is a new flight to be added that has departure location at Alberta and arrival at Quebec. System Admin logs in to the system and receives authentification. The admin presses ‘new flight’ and adds required information. Departure as Alberta, Arrival as Quebec, flight number, scheduled date … The system admin presses apply and the flight is added to the database.

Identified candidate objects:

- Flight
- admin

Manage aircrafts

System admin needs to change flight aircraft from #001 to #002. System admin authenticates in the system. He finds the specific flight using departure time and the flight aircraft number #001. System shows all info that can be edited. Fields flight number is changed from #001 to #002. System shows an apply (or changed) button, . Save is pressed and system shows the item changed and the field edited. The database is then updated to the changed aircraft.

Identified candidate objects:

- Database

Manage crews

Flight crew needs to take a day off and new employee is selected. The system admin accesses the database and authenticates the user. System responds with a good login. System shows the admins options, admin selects edit field. Admin finds item by searching with crew name, flight number, … Admin selects edit field on the item. Admin changes one crew member to another. System checks if valid, system shows success on valid field. Admin logs out.

Identified candidate objects:

- Crew member

Browse flight info

System admin inputs request with date from interface. Then request is generated and passed to the database. The database returns the flight info of the given date, then passed back to display. Return “No result” if no flights on the specific date given.

Identified candidate objects:

- System admin
- Database port
- Flights
- Date
- Display

Browse aircraft

System admin inputs request of aircraft from interface. Then request is generated and passed to the database. Database returns the aircrafts list as well as basic information of them the airline has, and passed back to display. Return “No result” if no aircraft registered.

Identified candidate objects:

- System admin
- Database port
- Aircrafts
- Display

Browse crews

System admin inputs request of crew list from interface, include flight number. Then request is generated and passed to the database. Database returns the list of crew, and passed back to display. Return “No result” if no crews arranged on the specific flight given.

Identified candidate objects:

- System admin
- Database port
- Crew
- Display

Get Credit card

Register client complete form and send to database. Database then check duplicate client exist or not and then create the credit card information. The information are then send it to back to the user and to billing system to register the card.

Identified candidate objects:

- Registered client
- Form
- Database
- Credit card
- Billing System

Access perks

Client fill in forms including their name, address and submit application. Database add the information of registered client and check if duplicate client exist. Then is returns a membership number to client or “Duplicate users” if already exist one.

Identified candidate objects:

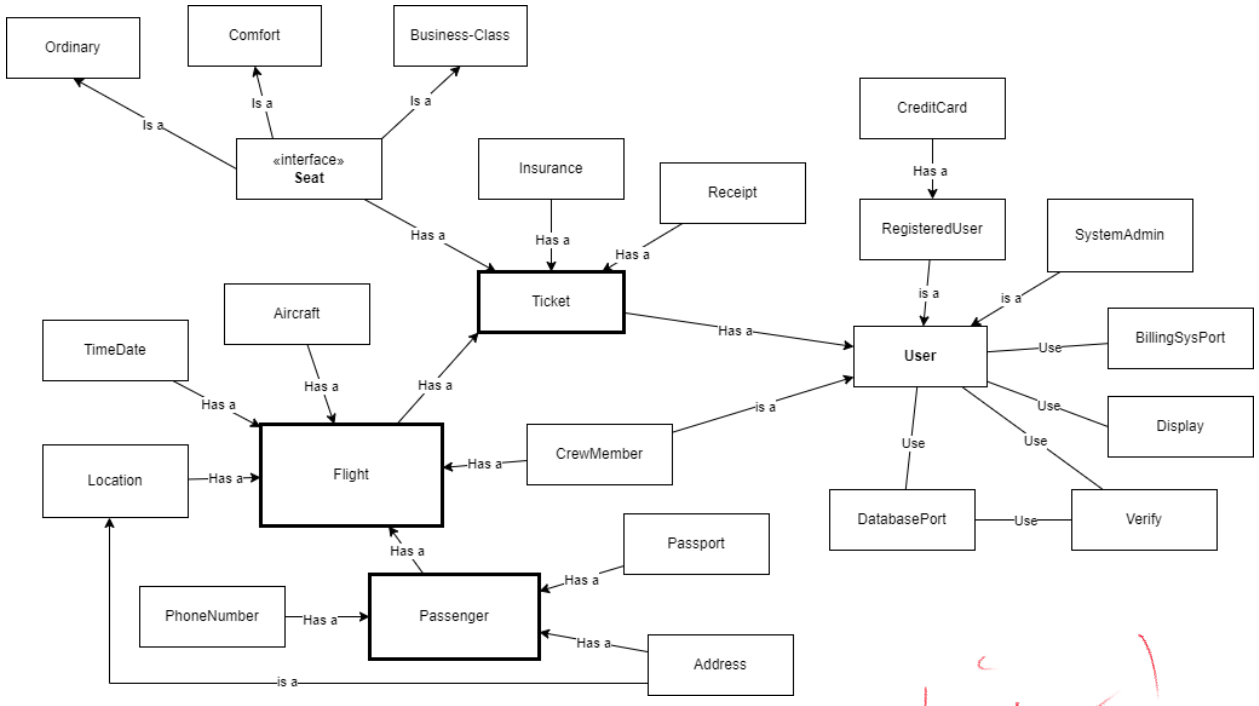
- Client
- Form
- Address
- Database

- Membership number

Objects:

1. Flight
2. Location
3. TimeDate
4. Seat
5. User
6. Ticket
7. Insurance
8. Crew Member
9. Passenger
10. Aircraft
11. Registered user
12. System admin
13. Database port
14. Credit card
15. Billing System port
16. Display
17. Address
18. Phone number
19. Receipt
20. Passport

A.4 Conceptual Model



Part B: Domain Diagrams

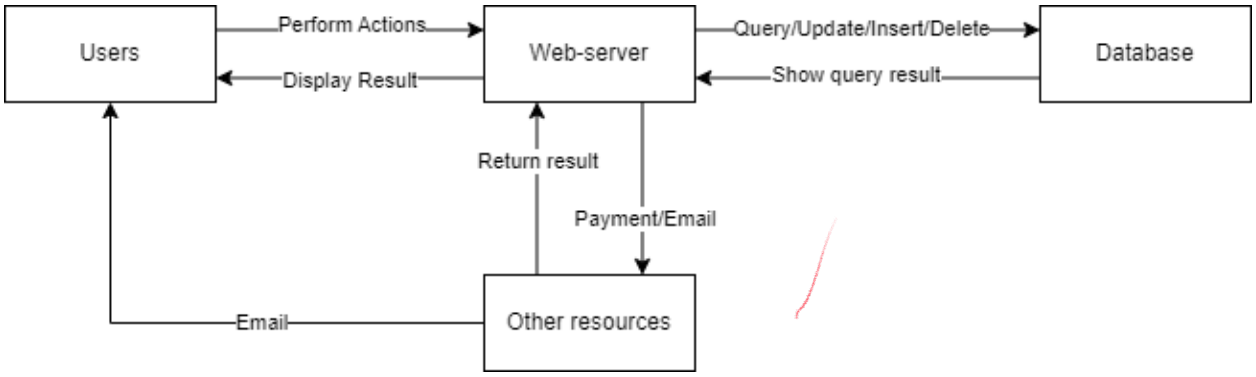
B.1: Highlights of the system's architecture

Client-Side

- Web application allow users to interact with server
- Handles user interactions such as streamlined purchasing of a flight
- Gathers necessary information from the backend display for the user
- Support for email notification of user tickets
- Modular design that promotes maintainability and scalability
- Incoreperates polymorphism for different kinds of users to have unique functionality

Server-Side

- Built using MySQL
- Stores information in regard to flights, passengers, crew members, tickets, and other relevant details.
- Allows for easy access of stored data within the database via queries which are then presented to the user such as available flights to fit their criteria



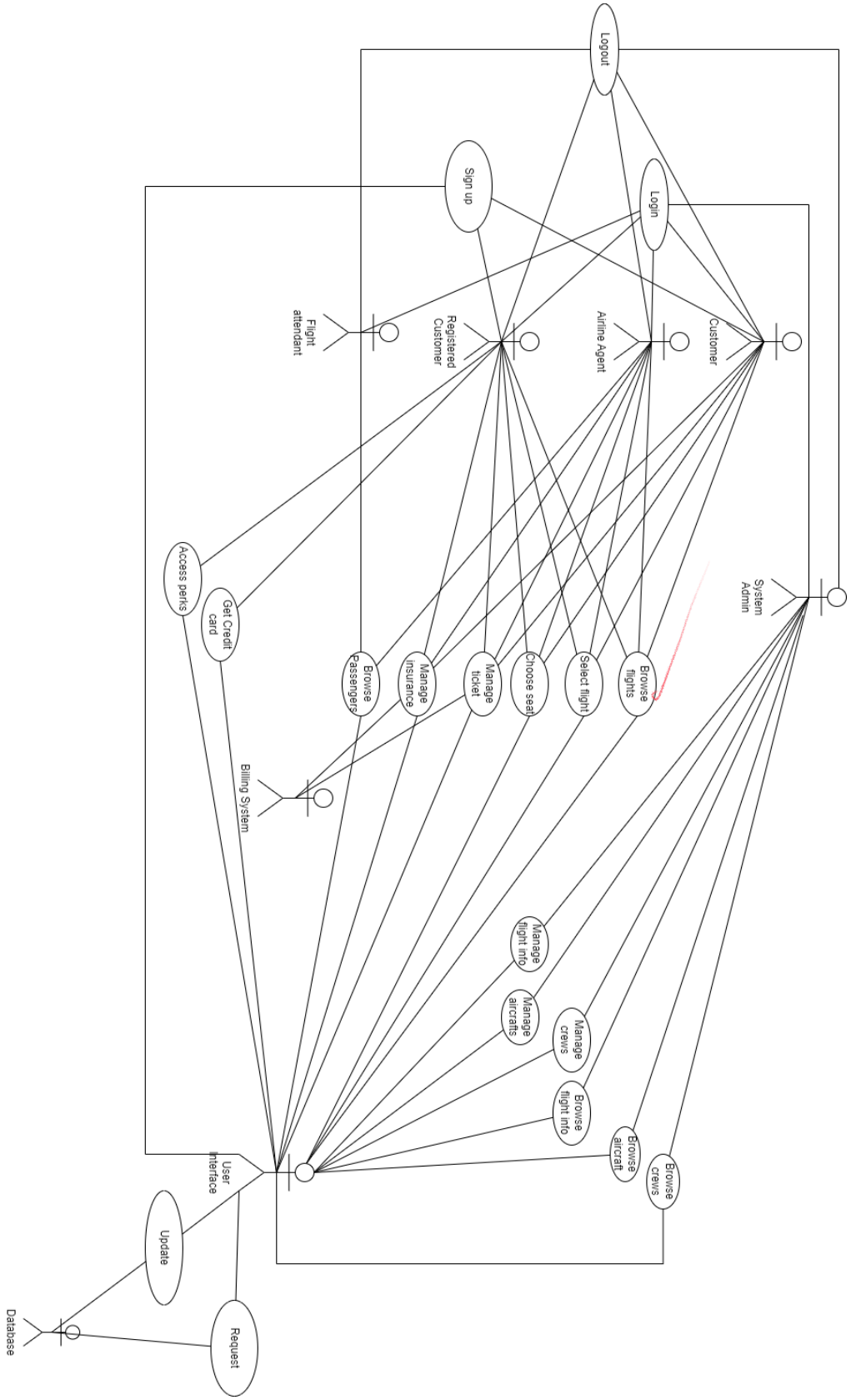
Modular Design

- The program is separated into packages for distinct functionalities to enhance maintainability and scalability
- While each package functions independently, they are still designed to work together seamlessly to achieve an effective system architecture

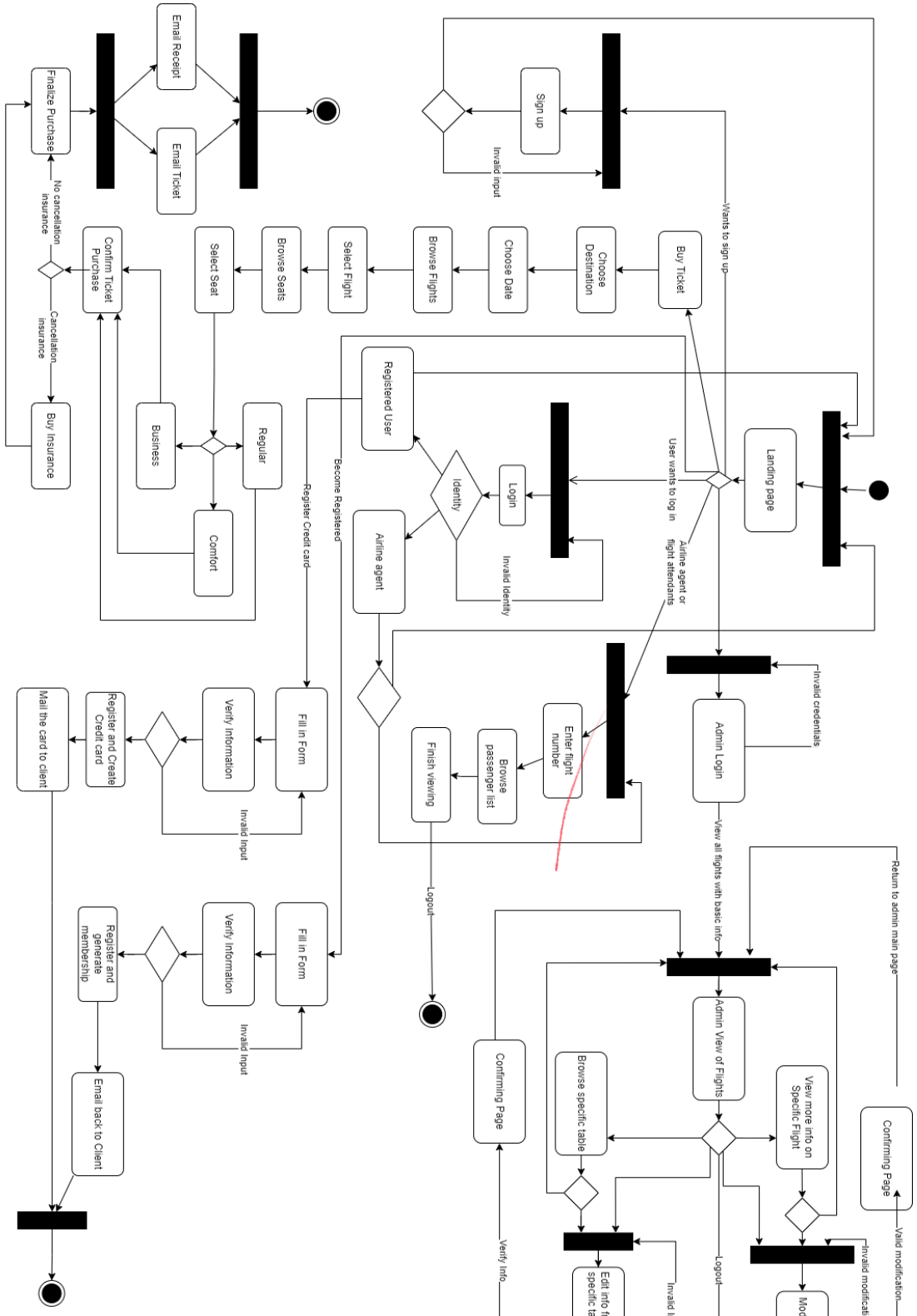
Authentication and Validaiton

- Input validations for things such as username/passwords and emails
- Authentication of different roles within the system ensuring only specific users (such as admins) have special privileges or a different view of the program

B.2: Updated use case diagram

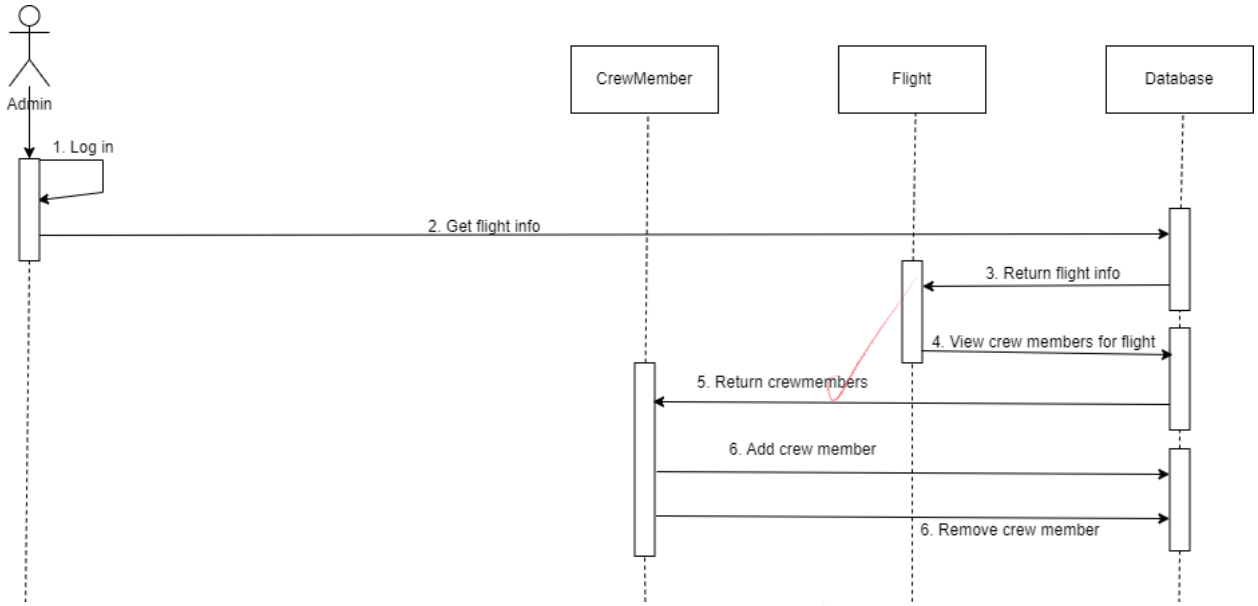


B.3: Systems activity diagram

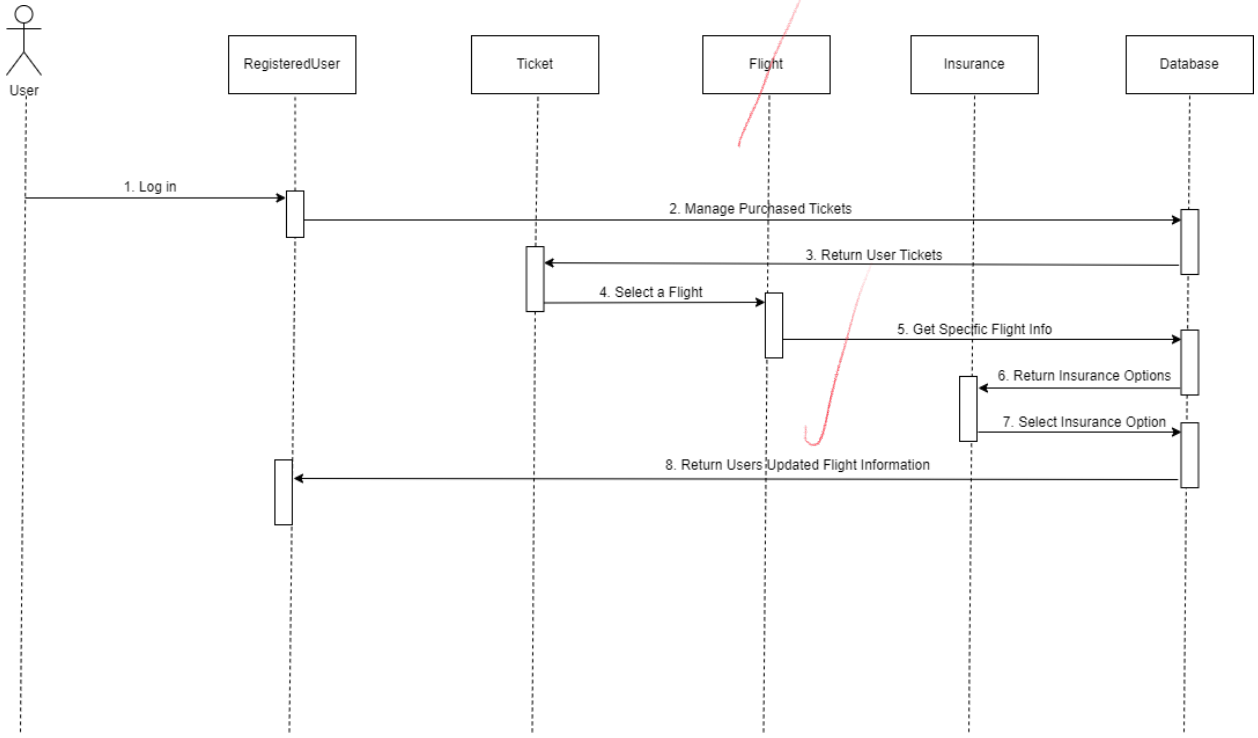


B.4: Sequence diagram

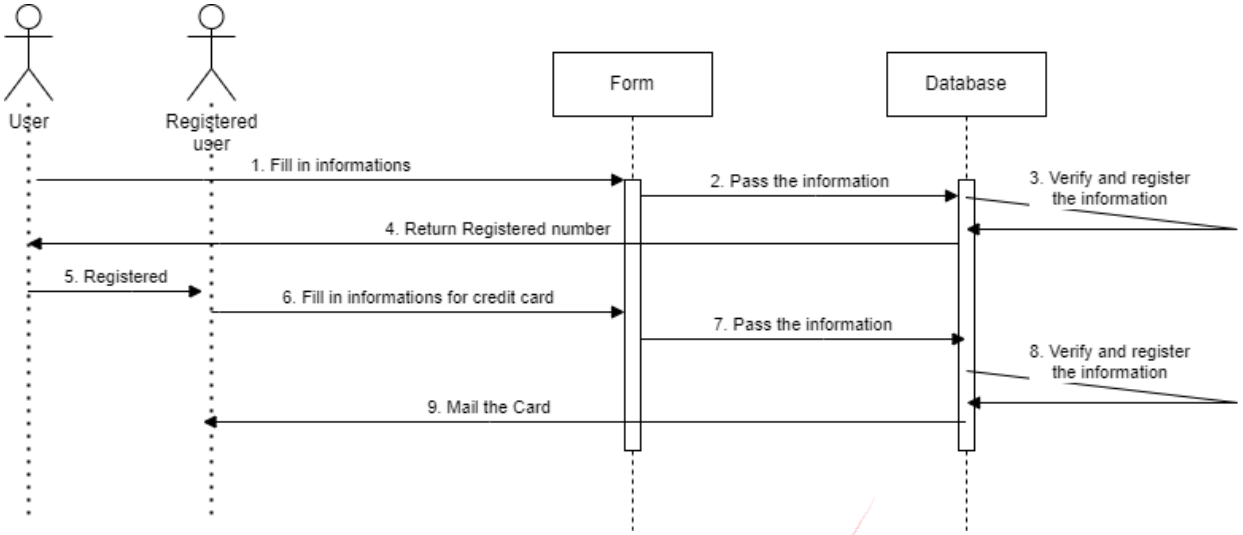
1. Manage crew member



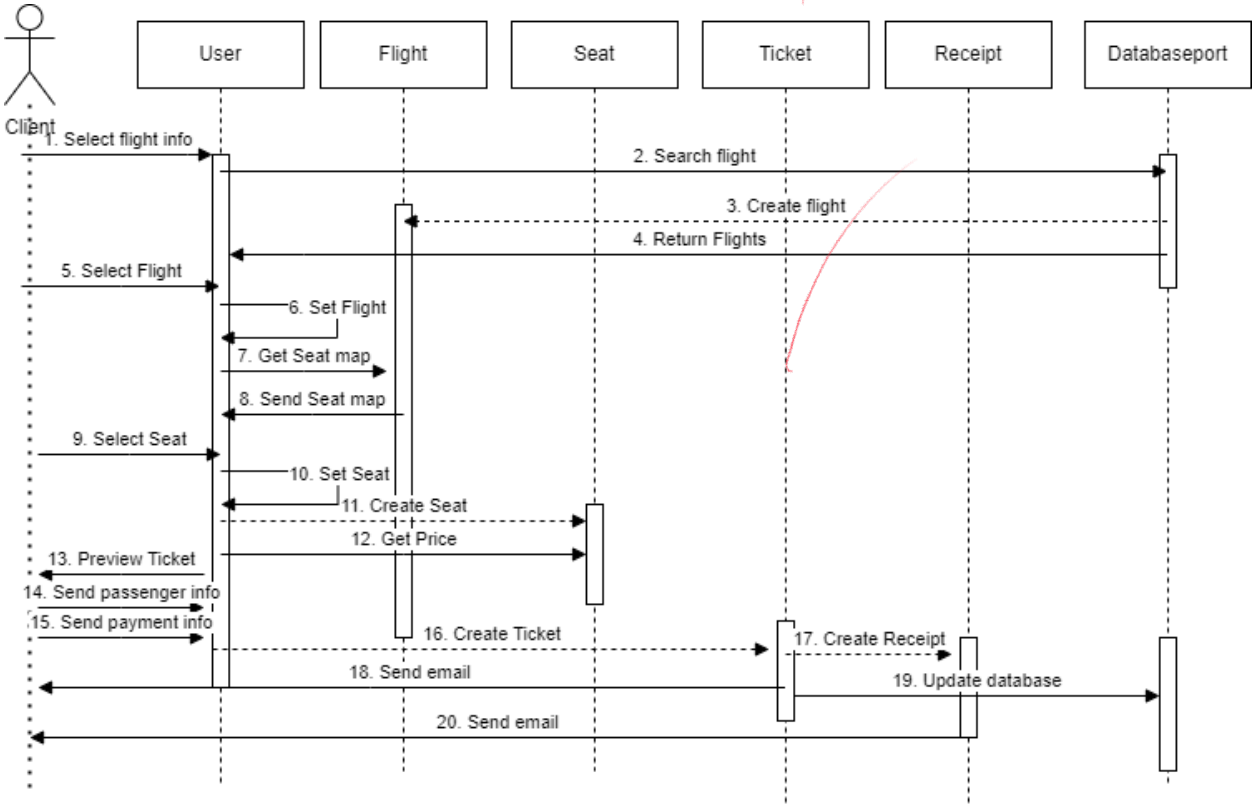
2. Manage insurance



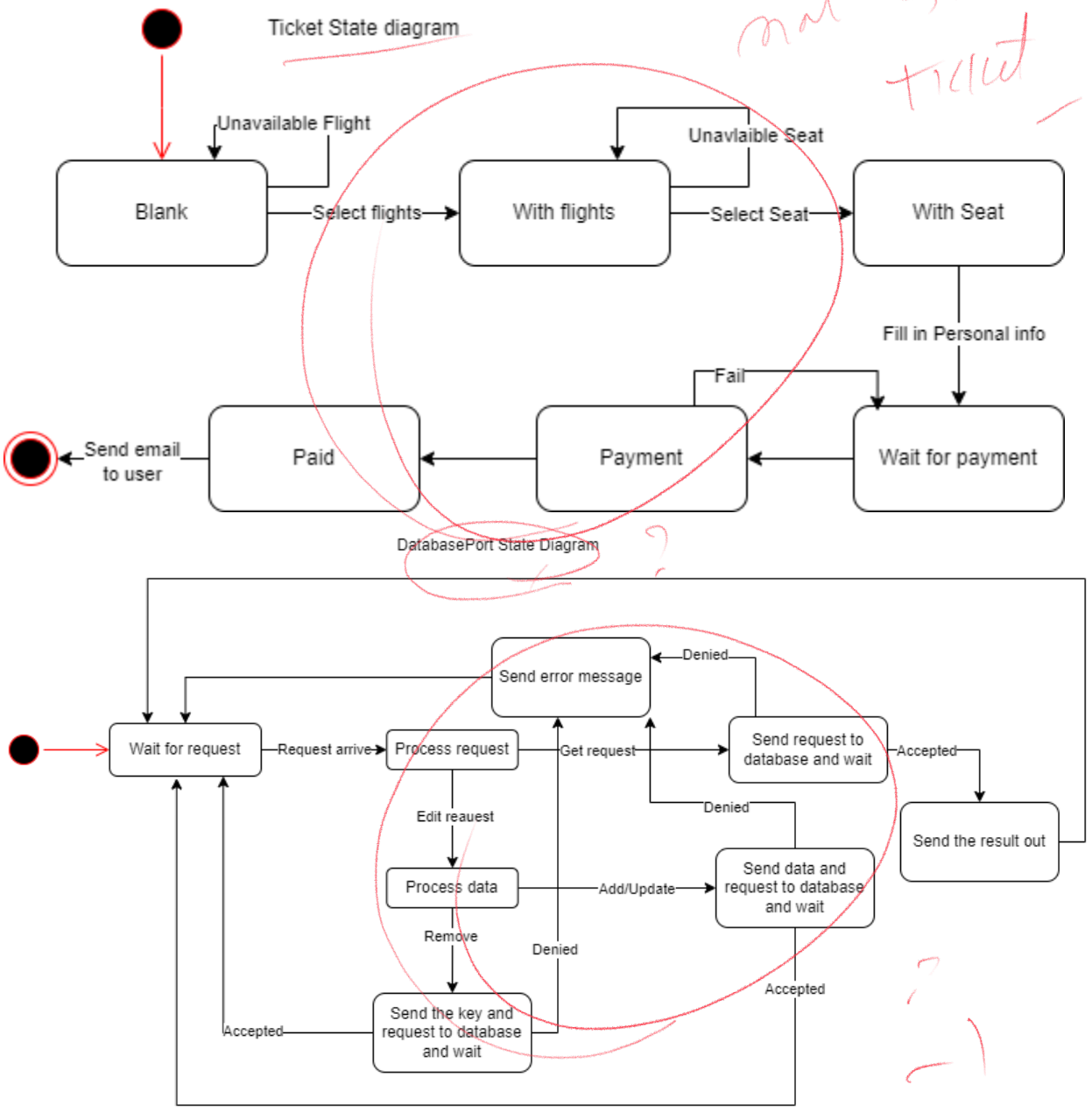
3. Register credit card



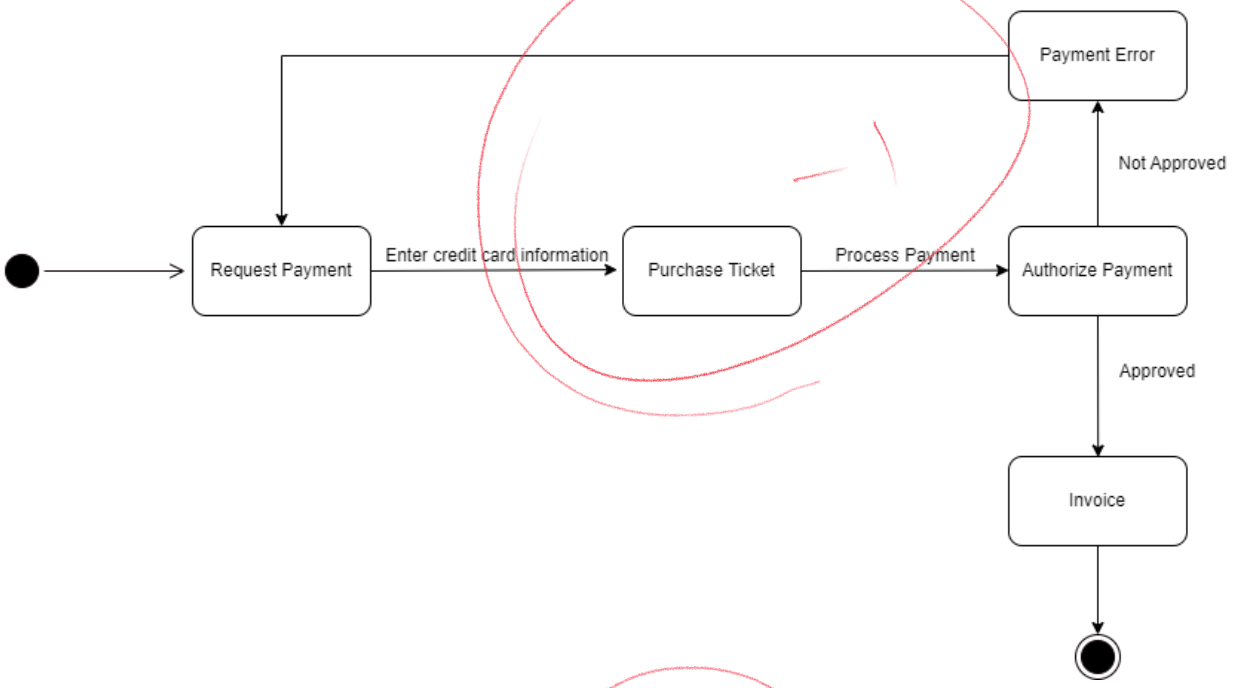
4. Buy ticket



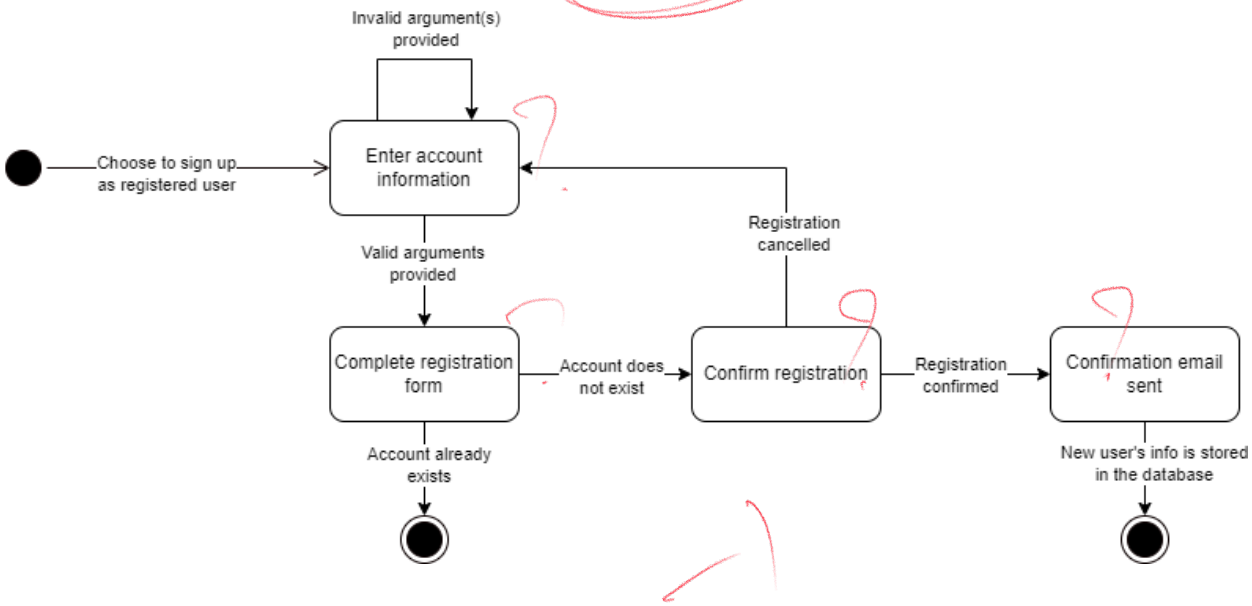
B.5: State Diagram



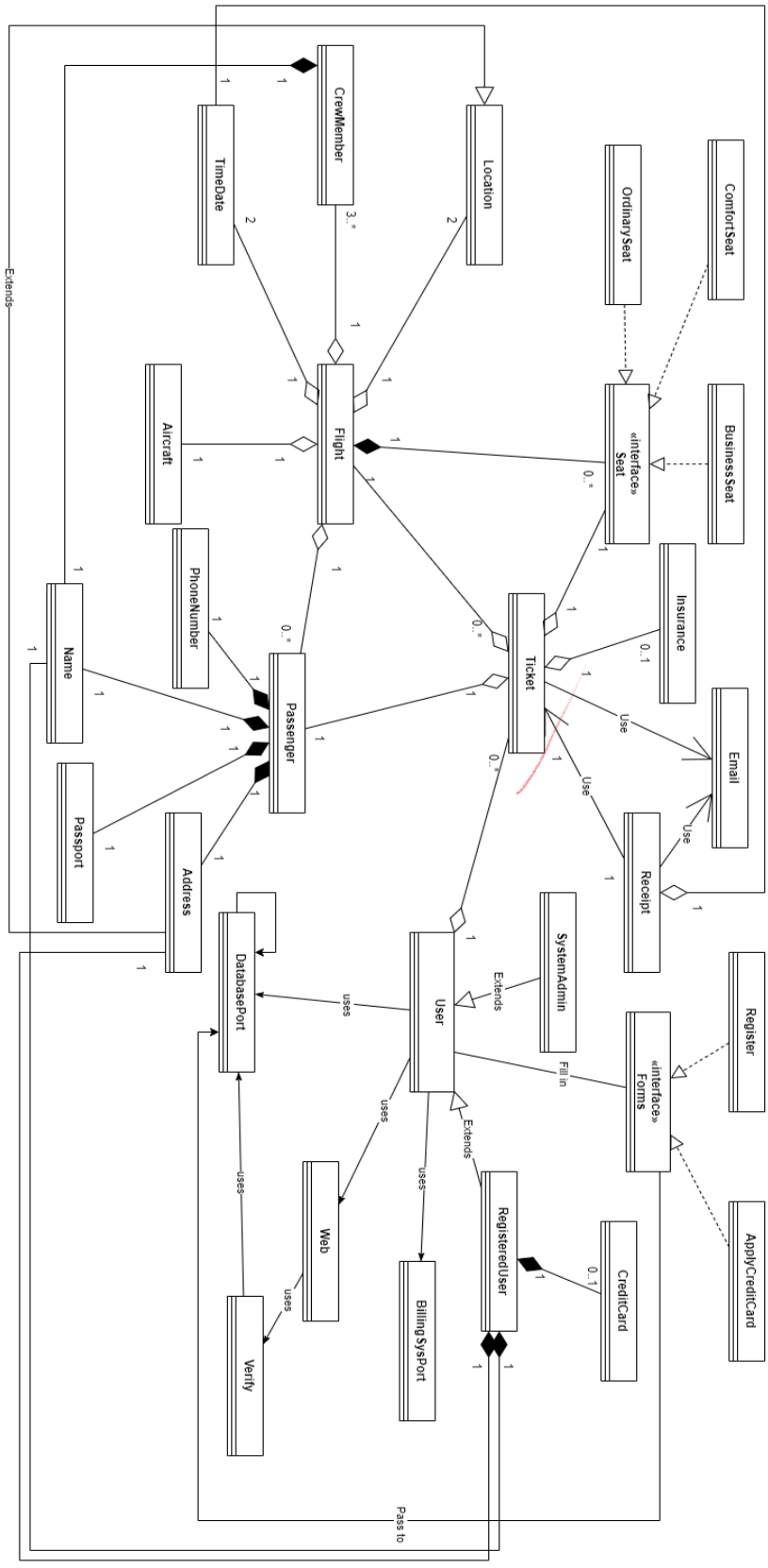
Payment State Diagram



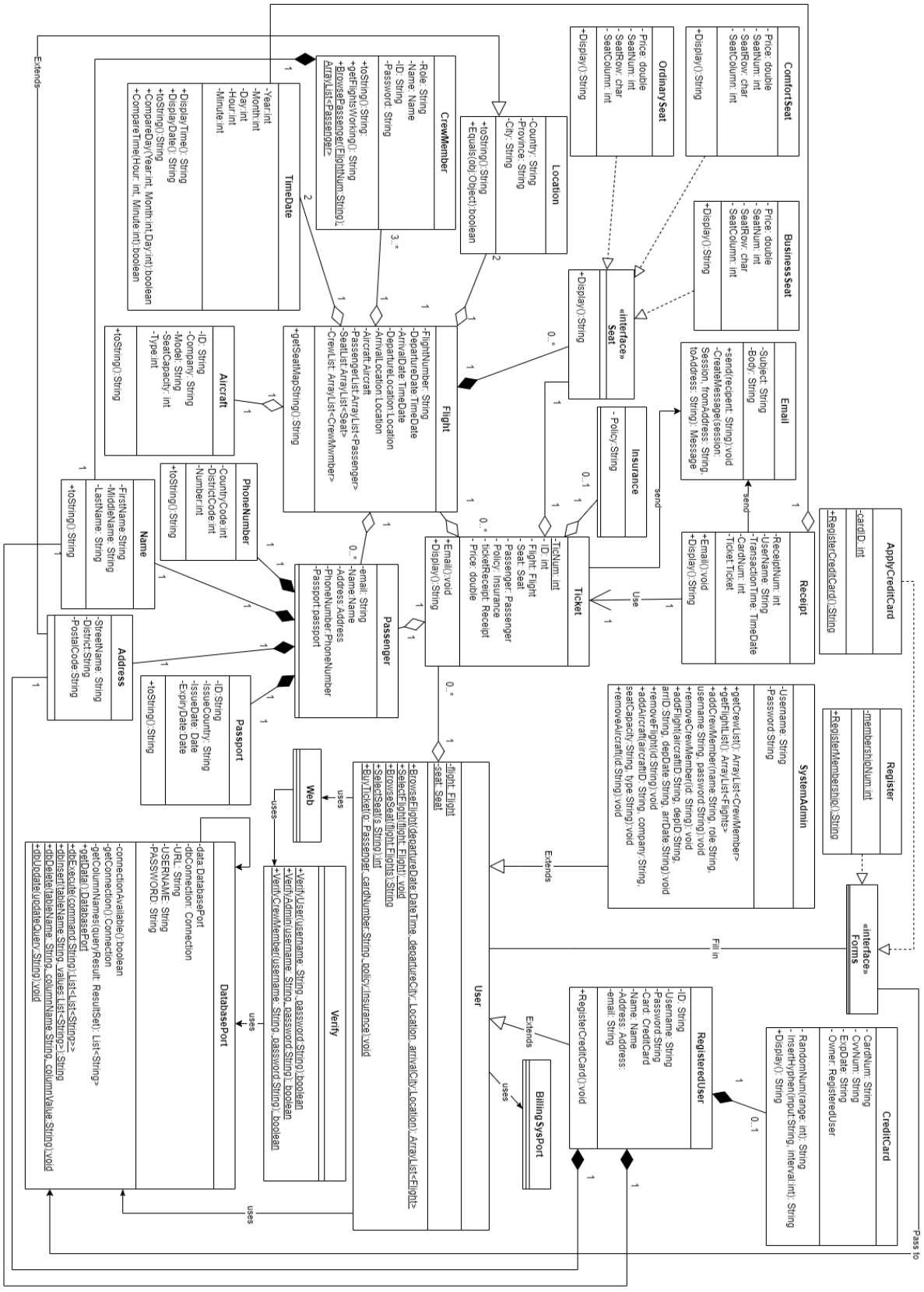
Sign Up As Registered Member State Diagram



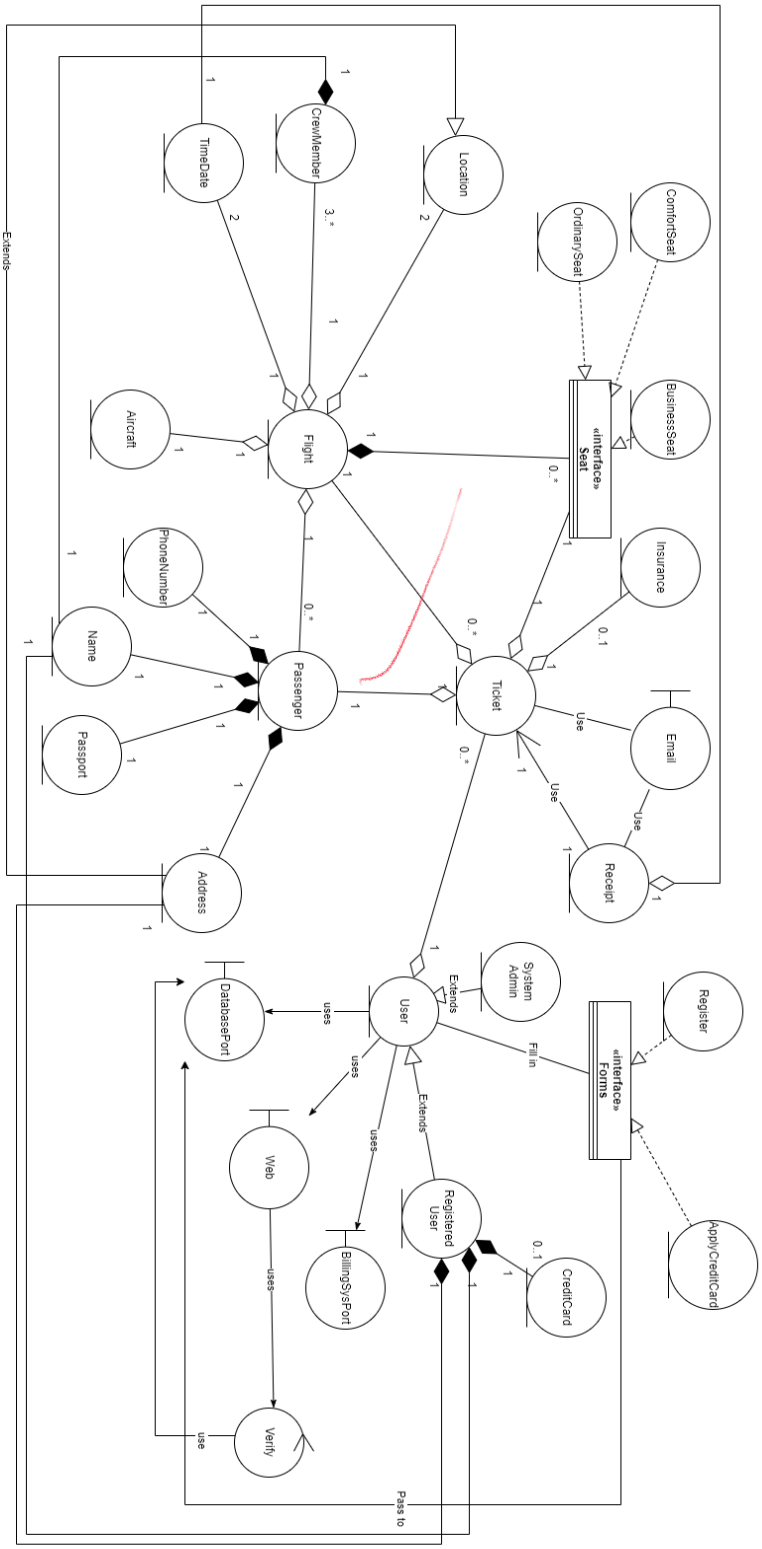
B.6: System's Domain class diagram (Simple)



B.7: System's Domain class diagram

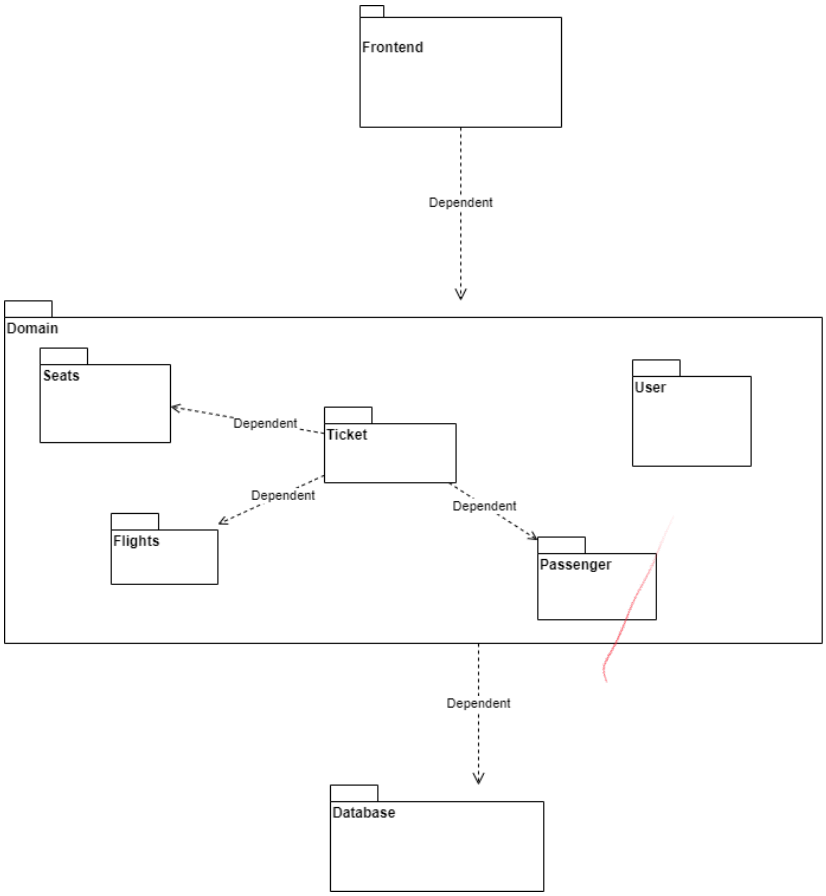


C: System’s Detailed Design-Class Diagram



D:High-Level System's Architecture

D.1: Package Diagram



D.2 Deployment Diagram