Course: ENSF 614 – Fall 2023
Term Project
Group 19

Student Names: Jenn Bushey
Robby Parmar
Harshil Patel
Benjamin Reid

Submission Date: December 3, 2023

Design Phase

Part One - System Analysis

1. System Description

The Flight Web Application is a comprehensive and user-friendly system designed to facilitate the seamless booking and management of flights for various categories of users, including tourism agents, airline agents, registered users, and system administrators. Developed to streamline the entire flight reservation process, the system incorporates advanced features to meet the diverse needs of its users.

Actors:

- User:
    - User:
        - Users are airline customers who have the ability to browse, select, and book flights, and manage their bookings.
    - Registered User:
        - Registered users are airline customers who have registered for membership within the flight reservation system. They have the ability to browse, select, and book flights, as well as manage their bookings. Registered users may also enjoy additional benefits such as receiving monthly promotion news, using the airport longest with a discounted price, receiving a free companion ticket once per year, and applying for the company's credit card.
- Agent:
    - Tourism Agent:
        - Tourism agents are individuals or agencies that specialize in assisting customers with travel plans, including flight reservations. They act as intermediaries between travelers and airline companies, providing guidance and support in choosing suitable flights and destinations.
    - Airline Agent:
        - Airline agents are representatives of the airline company. They play a crucial role in assisting customers, managing bookings, and ensuring a smooth travel experience. Airline agents may include those involved in reservations, customer service, and flight operations.
    - Flight Attendant:
        - Flight attendants are members of the cabin crew responsible for ensuring the safety and comfort of passengers during flights. While not directly involved in the booking process, they may access information about passengers and flights for in-flight safety..

- System Admin:
    - System administrators are responsible for managing and maintaining the overall functionality and security of the flight reservation system. They have access to administrative tools to configure databases, manage users, and oversee the system's integrity.
- External Systems:
    - Email Server:
        - The email server is an external system responsible for sending and receiving emails. It plays a vital role in delivering essential communications to users, such as e-tickets, payment receipts, and cancellation confirmations.
    - Payment System:
        - The payment system is an external service or component that handles financial transactions. It processes payments made by users for flight reservations, ensuring secure and efficient transactions using credit cards or other payment methods.
- Database
    - The database contains essential information about flights, including crew, aircraft, destinations, schedules, availability. It serves as a core component for users and agents to browse and select flights based on their preferences.
    - The database stores information about registered users, including their names, email addresses, and other relevant details.

2. System Use-Case Diagram

3. System Scenarios[1]

3.1. Use Case: Booking a Flight

Description:
Meet John, a registered user excited about her upcoming vacation. He logs in, selects his dream destination, and browses through the available flights. After choosing the perfect one, he picks his favorite seat on the graphical seat map, opts for ticket cancellation insurance, and makes a secure online payment using his credit card. Within moments, he receives his ticket and payment receipt via email.

Actors:

- User
- Agent
- External Systems
- Database

Preconditions:

- Registered User is logged into the system.
- Guest users are not logged into the system.
- Flights are available for the selected date and destination.

Main Flow:

1. Actor initiates the "Book Flight" action.
2. The system displays a list of available flights.
    - Users and agents can view details such as flight numbers, departure times, and prices.
    - Users and agents can filter the flight by origin or destination.
3. Actor selects a desired flight
4. The system presents a graphical representation of available seats for the selected flight.
5. The system presents the option to add cancellation insurance and the Actor decides whether to include this insurance.
6. The system displays the checkout screen. The system calculates the total cost, including the selected seat and optional insurance and displays this information on the screen.
7. Actor proceeds to make a payment, including taxes.
    - Actor enters credit card details for payment.
    - The system applies GST and PST if applicable.
8. Upon successful payment, the system generates an e-ticket and sends it to the actor via email.
9. The system database is updated.

Postconditions:

[1] Candidate objects are underlined.

- Actor has successfully booked a flight.
- The selected seat is reserved for the actor on the chosen flight.
- Actor receives an e-ticket and payment receipt.
- The system is updated with the latest booking information.

3.2. Use Case: Canceling a Flight

Description:
Life is unpredictable, and plans change. John, a registered user, needs to cancel his flight. No worries! He logs in, navigates to the "Cancel Flight" section, selects his flight, and confirms the cancellation. The system processes the request and sends him a cancellation confirmation via email.

Actors:
- User
- Agent
- External Systems
- Database

Preconditions:
- User has a booked flight.

Main Flow:
1. User selects the “Cancel Flight” button.
2. The system displays a list of the user's booked flights.
3. User selects the specific flight to cancel.
4. The system confirms the cancellation request.
    - The system verifies the cancellation request and checks if any refund is applicable based on the cancellation policy.
5. The system cancels the booking and removes the booking from the database.
    - The system updates the booking status to "canceled" and processes any applicable refund to the user's payment method.
6. User receives a cancellation confirmation via email.

Postconditions:
- User's flight is canceled.
- Cancellation confirmation email is received by the user.

3.3. Use Case: Airline Agents Print List of Registered Users

Description:
Before the aircraft takes off, Liam, a flight attendant meticulously prepares the passenger list, reviewing details such as names, seat assignments, and any special requests or requirements. This allows him to be well-informed about the composition of passengers on the flight, ensuring a seamless and safe in-flight experience.

Actors:
- Airline Agent
- Database

Preconditions:
- Agent is logged into the system.

Main Flow:
1. Agent initiates the "Passenger Manifest" action.
2. The system prompts the Agent to input a Flight Number.
3. The system displays a list of passengers for the active flight.
4. Agent reviews passenger details including names and seat numbers.

Postconditions:
- Agent has successfully browsed the passenger list for the active flight.
- Relevant passenger details are available to the agent.

3.4. Use Case: Registering User

Description:
A customer has decided to register with the airline. The main menu is opened and the user clicks “Login”. The login GUI is opened. The user proceeds to click “Register”. The register GUI is opened. The user proceeds to write in all information required for registration. A controller processes the input and calls classes from another package to enter the newly registered information to the database. The registered user interface is opened.

Actor:
- User
- Database

Preconditions:
- User is not currently a registered member.

Main Flow:
1. User initiates the application.

- User starts the application and the main menu is opened.

2. User clicks the “Login” button.
    - User clicks “Login” and a Login GUI is opened.
3. User clicks the “Register” button.
    - User clicks “Register” and a Register GUI is opened.
4. System prompts the user for username, password, and email.
    - User enters all information required for registration.
5. User clicks the “Enter” Button.
    - System retrieves all the information and proceeds to control the flow of logic for saving the user’s information into the database.
    - Controller calls classes to save the user’s information into the database.
    - Controller opens the registered user interface once successfully saved.

Postconditions:

- Registered User is successfully registered as a member.
- The system is updated with the user's email and password

3.5. Use Case: Manage Flight Information

Description:
In the bustling world of airline operations, our trusty System Admin, Emma, takes on a superhero role, seamlessly managing the intricacies that keep everything soaring smoothly. Starting with a casual flight check, they effortlessly browse through available flights, ensuring each departure is well-documented and ready for eager travelers. A quick pivot to crew management has our System Admin orchestrating the flight crews like a maestro, making sure everyone is in sync for a flawless performance. Next up, a glance at the fleet – aircraft details are checked and adjusted with the precision of a seasoned pilot. With a nod to destinations, our admin maps out the airline's route, tweaking, adding, or removing destinations as needed. Flight details are meticulously reviewed and updated, ensuring every piece of information is as crisp as the morning air at takeoff. The admin's final touch involves curating a list of the airline's registered users, a community connected through the shared love of travel.

Actor:

- System Admin
- Database

Preconditions:

- System Admin is logged into the system.

Main Flow:

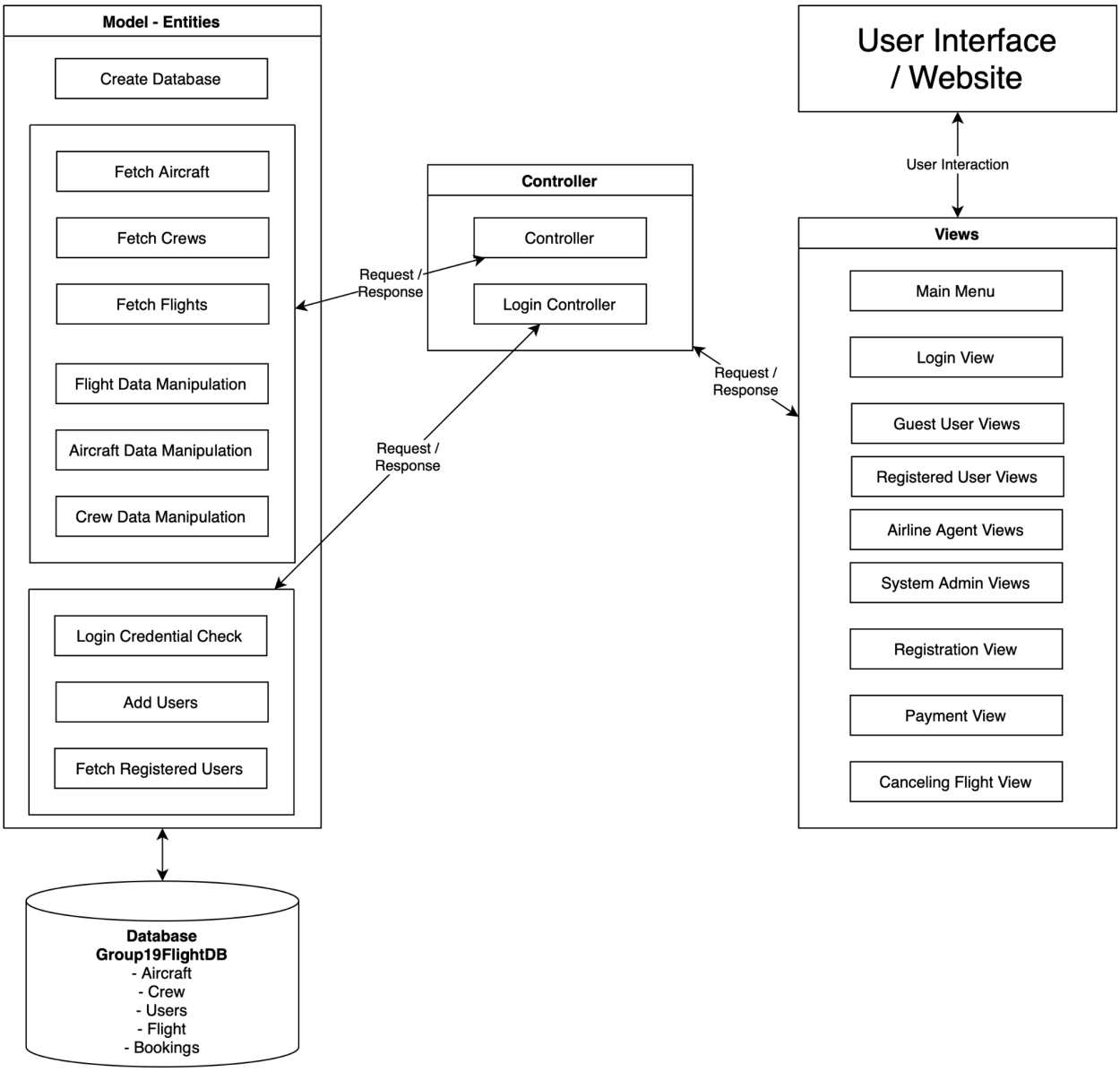
1. System Admin initiates the "Browse Flight" action.
2. The system displays a comprehensive list of flights, allowing the System Admin to review information such as flight numbers, aircraft, crew and schedules.

3. System Admin initiates the "Browse Crew" action.
4. The system displays a list of crew members, including their names, ID number and assigned flights.
5. System Admin initiates the "Browse Aircraft" action.
6. The system displays information about each aircraft including aircraft model.
7. System Admin initiates the "Manage Crew" action.
8. System Admin can choose to insert, update, or delete a crew member from the roster.
9. The system confirms the insert, update, or delete was a success.
10. The System Admin can verify the action was completed by selecting the “Browse Crew” action from within the "Manage Crew" view
11. System Admin initiates the "Manage Aircraft" action.
12. System Admin can choose to insert, update, or delete an aircraft from the roster.
13. The system confirms the insert, update, or delete was a success.
14. The System Admin can verify the action was completed by selecting the “Browse Aircraft” action from within the "Manage Aircraft" view
15. System Admin initiates the "Manage Flights" action.
16. System Admin can choose to insert, update, or delete a flight including departure location and time; arrival location and time; aircraft; and crew.
17. The system confirms the insert, update, or delete was a success.
18. The System Admin can verify the action was completed by selecting the “Browse Flights” action from within the "Manage Flights" view
19. System Admin initiates the "Display Registered Users" action.
20. The system generates a printable list containing user details such as names, email addresses, and companion voucher status.
21. System Admin initiates the “Manage Promos” action.
22. The system accepts text input from the System Admin and will send an email to all registered users.

Postconditions:

- The System Admin has successfully browsed and managed various aspects of the airline's operations.
- Relevant databases are updated with the latest information.

4. System Conceptual Model


Model - Entities
Create Database
Fetch Aircraft
Fetch Crews
Fetch Flights
Flight Data Manipulation
Aircraft Data Manipulation
Crew Data Manipulation
Login Credential Check
Add Users
Fetch Registered Users
Database
Group19FlightDB
- Aircraft
- Crew
- Users
- Flight
- Bookings
Controller
Controller
Login Controller
Request / Response
Request / Response
Request / Response
User Interface / Website
User Interaction
Views
Main Menu
Login View
Guest User Views
Registered User Views
Airline Agent Views
System Admin Views
Registration View
Payment View
Canceling Flight View

Part Two - Domain Diagrams

1. Highlights of the System Architecture

Our Flight Application is based on the Model View Controller architecture using a monolith database for all Aircraft, Crew, Users, Flight, and Booking information.

The model handles all the data logic. Here we have our logic to read and write from the database.

The controller controls the data flow into a model object and updates the view whenever data changes. This implements the strategy pattern to ensure there is only one controller object in any moment of the application and also implements a strategy pattern to change the views displayed to the user.

The view layer implements the user interface displaying information and sending user inputs to the controller for action.

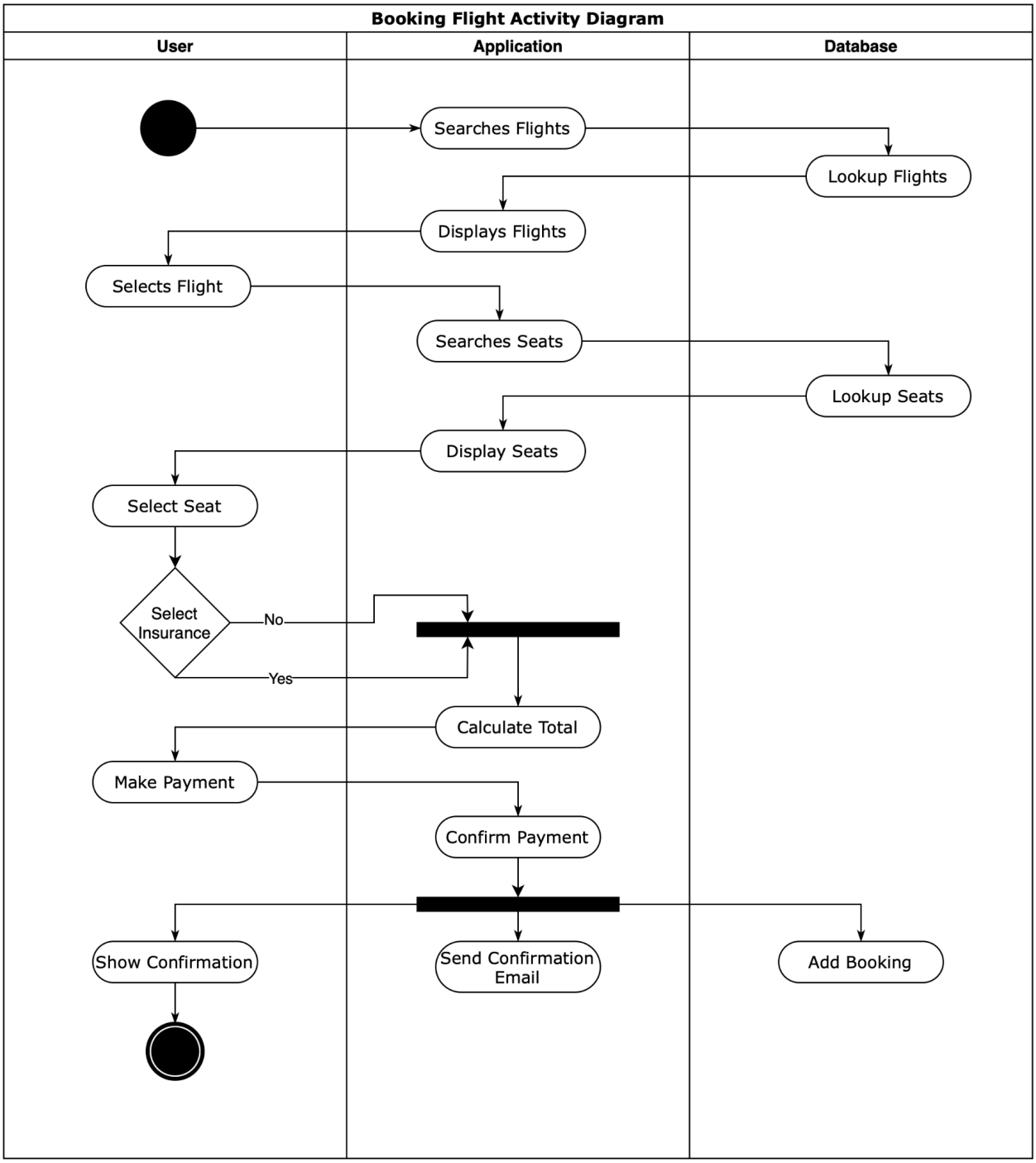
We chose to implement the MVC framework as it supports rapid and parallel development so our team members could be working on implementing separate views or user flows independently.

The MVC framework helps abstract the business logic from the display logic allowing for greater flexibility in each view.

2. Updated Use Case Diagram

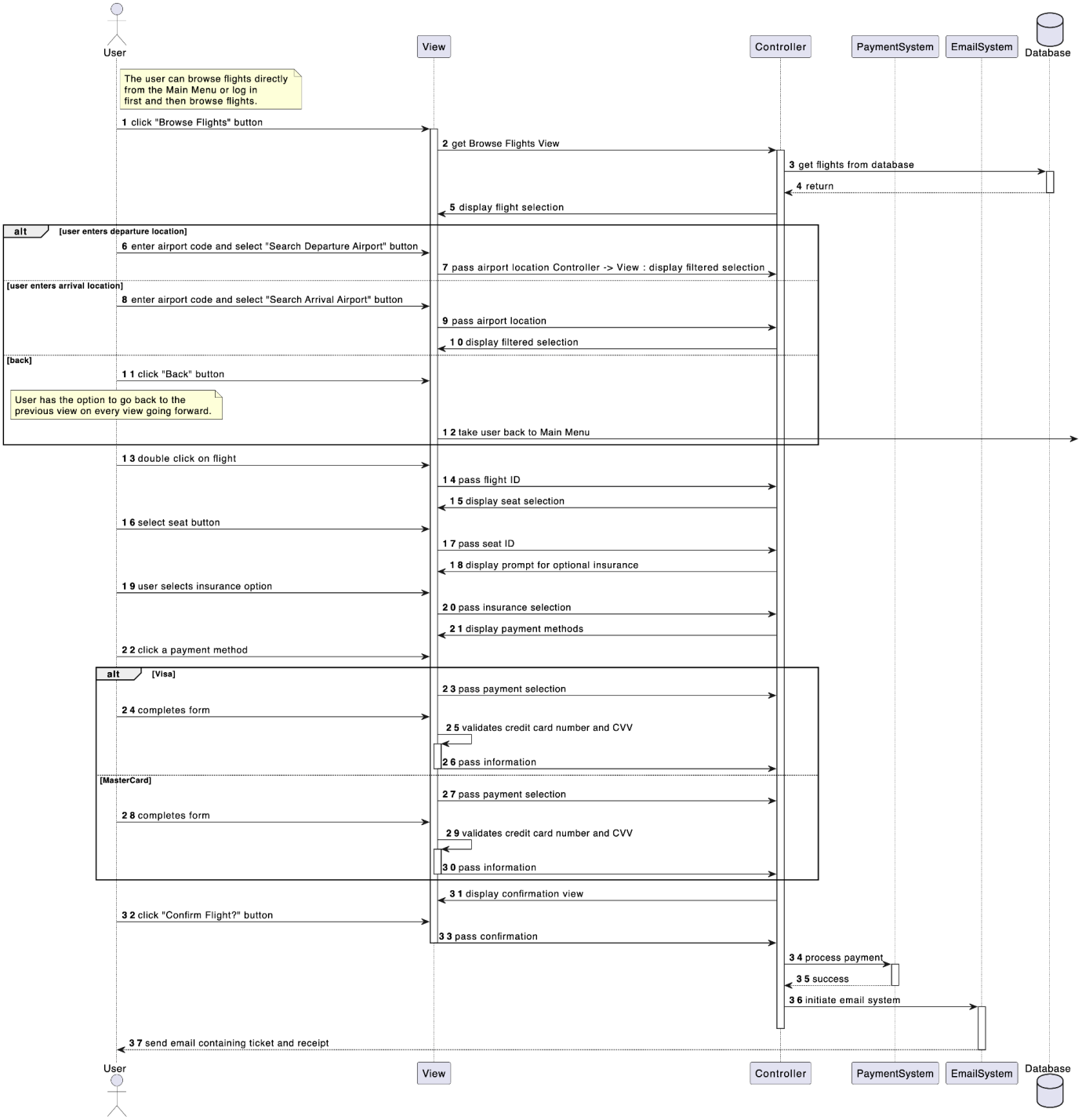
3. Systems Activity Diagram

3.1. Booking a Flight

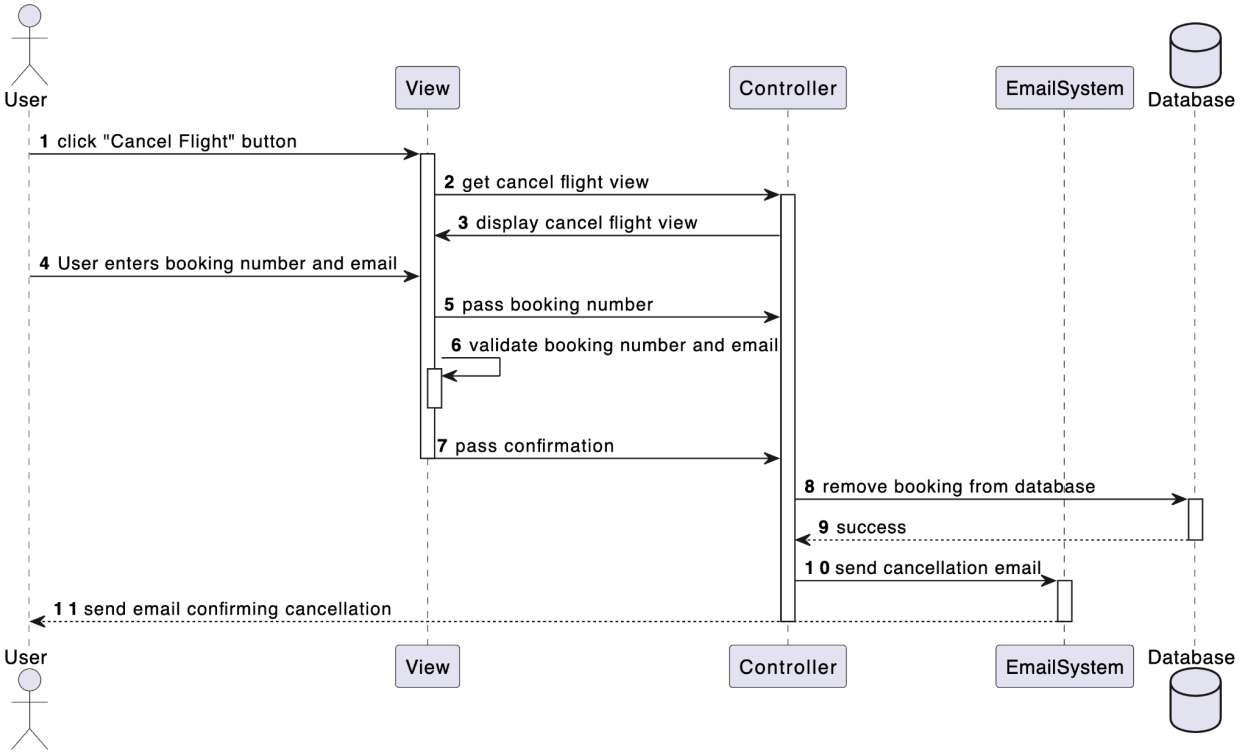


4. Sequence Diagram

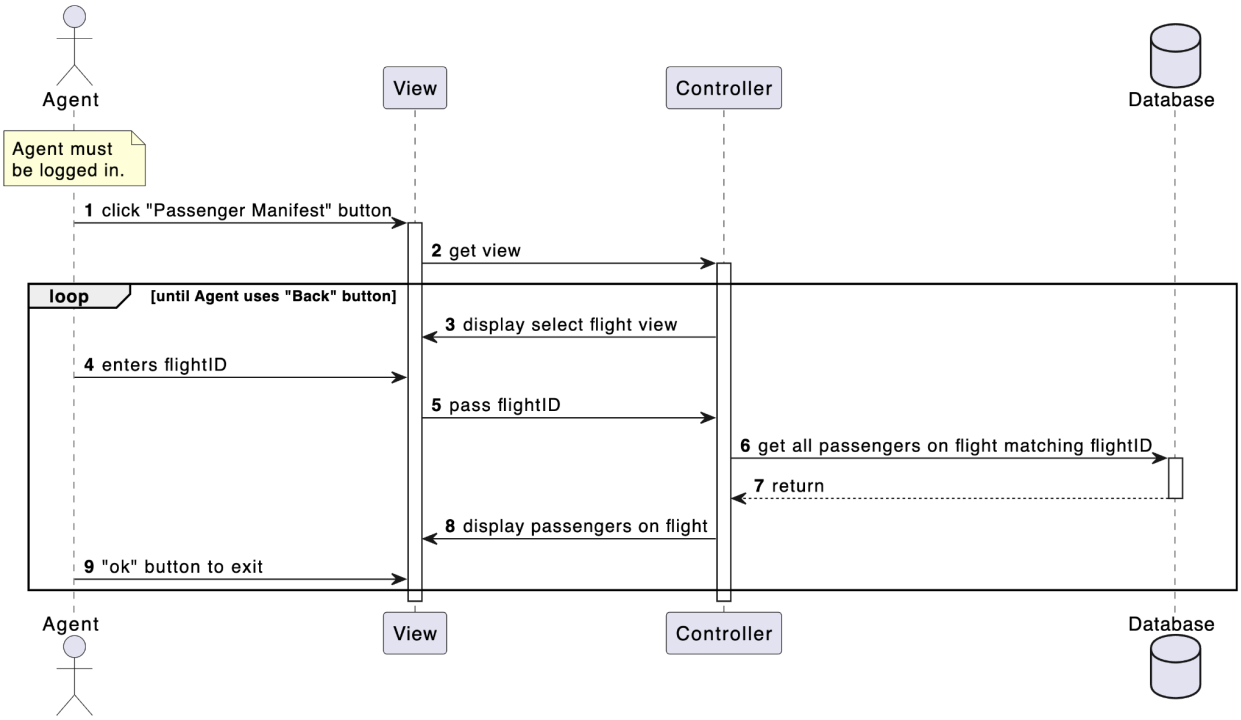
4.1. Booking a Flight



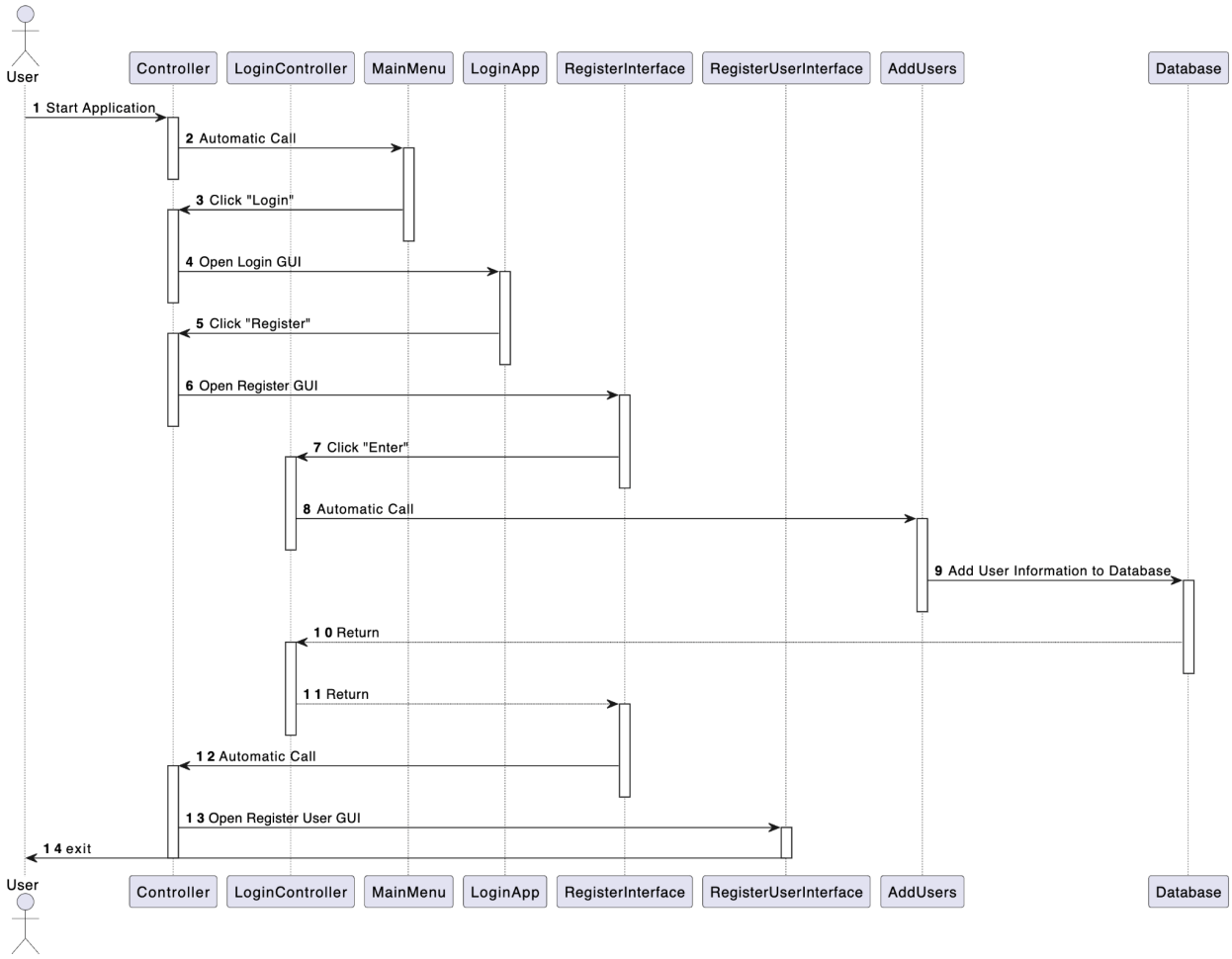
4.2. Cancelling a Flight



4.3. Airline Agents Print List of Registered Users

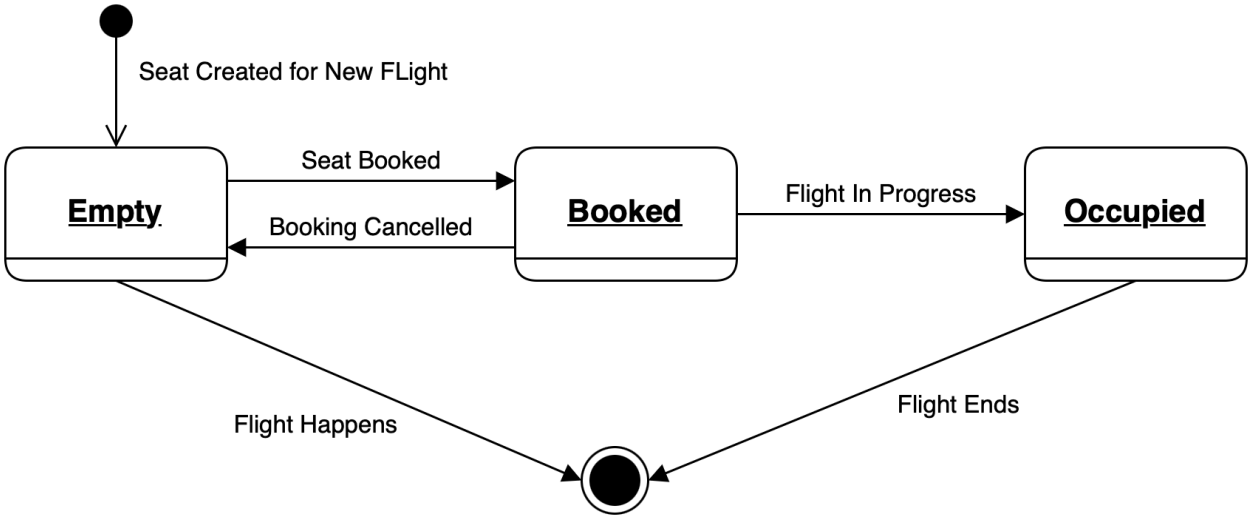


4.4. Register a User

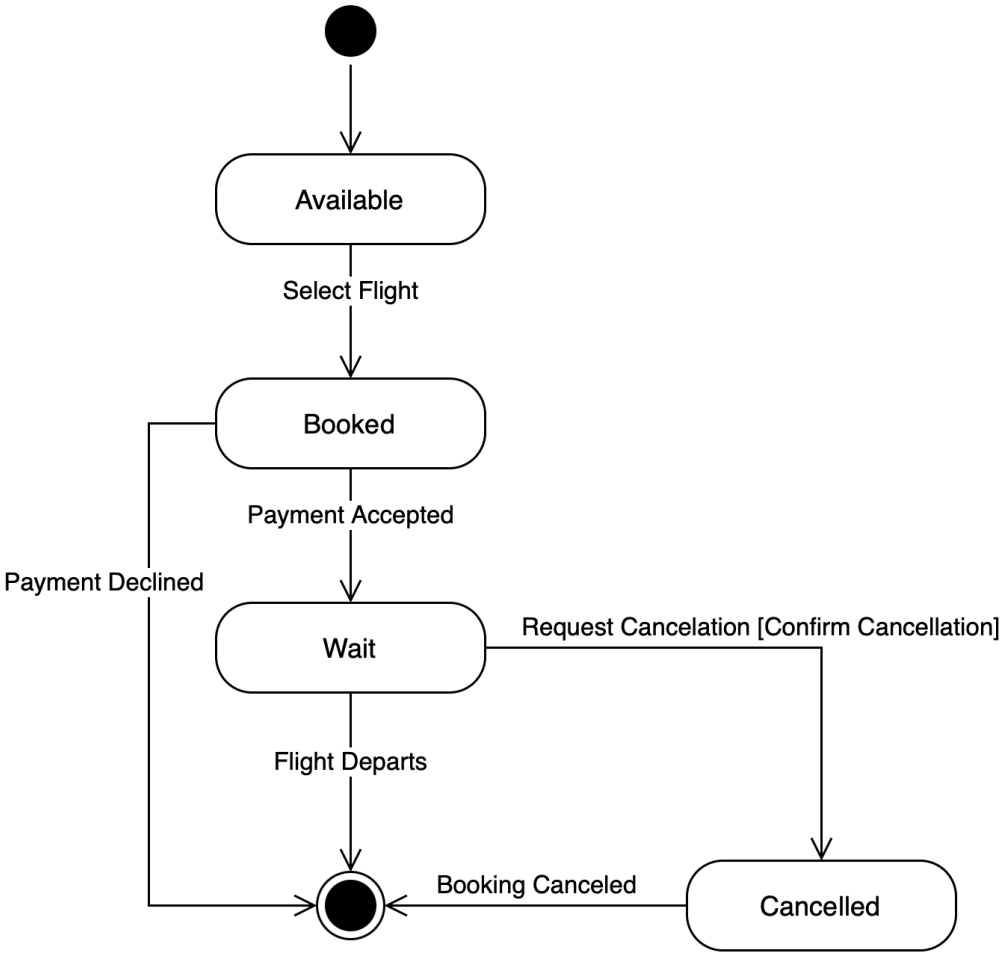


5. State Transition Diagram

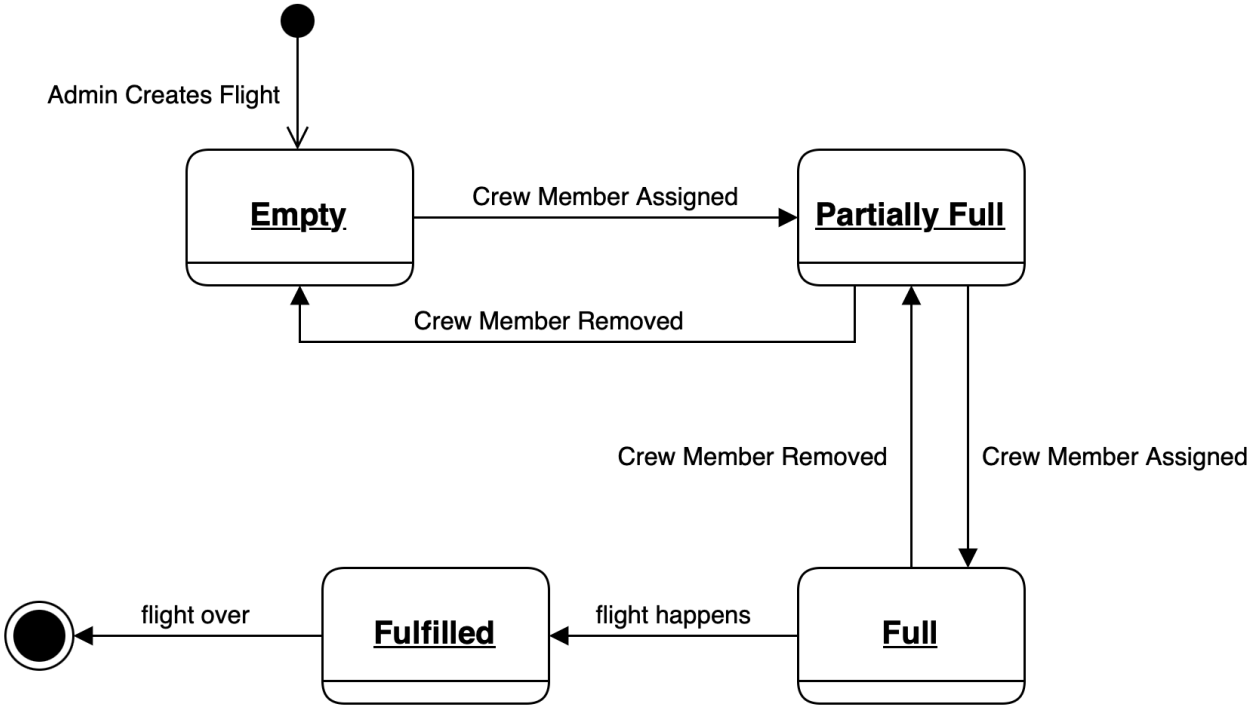
5.1. Seat Object



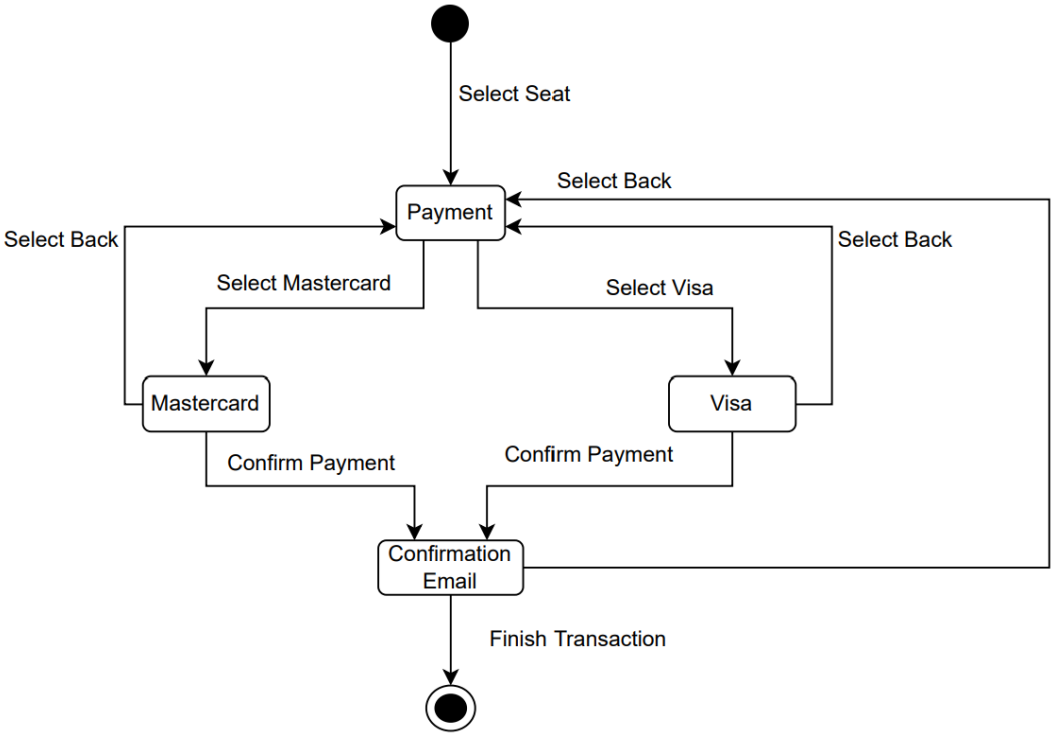
5.2. Flight Object



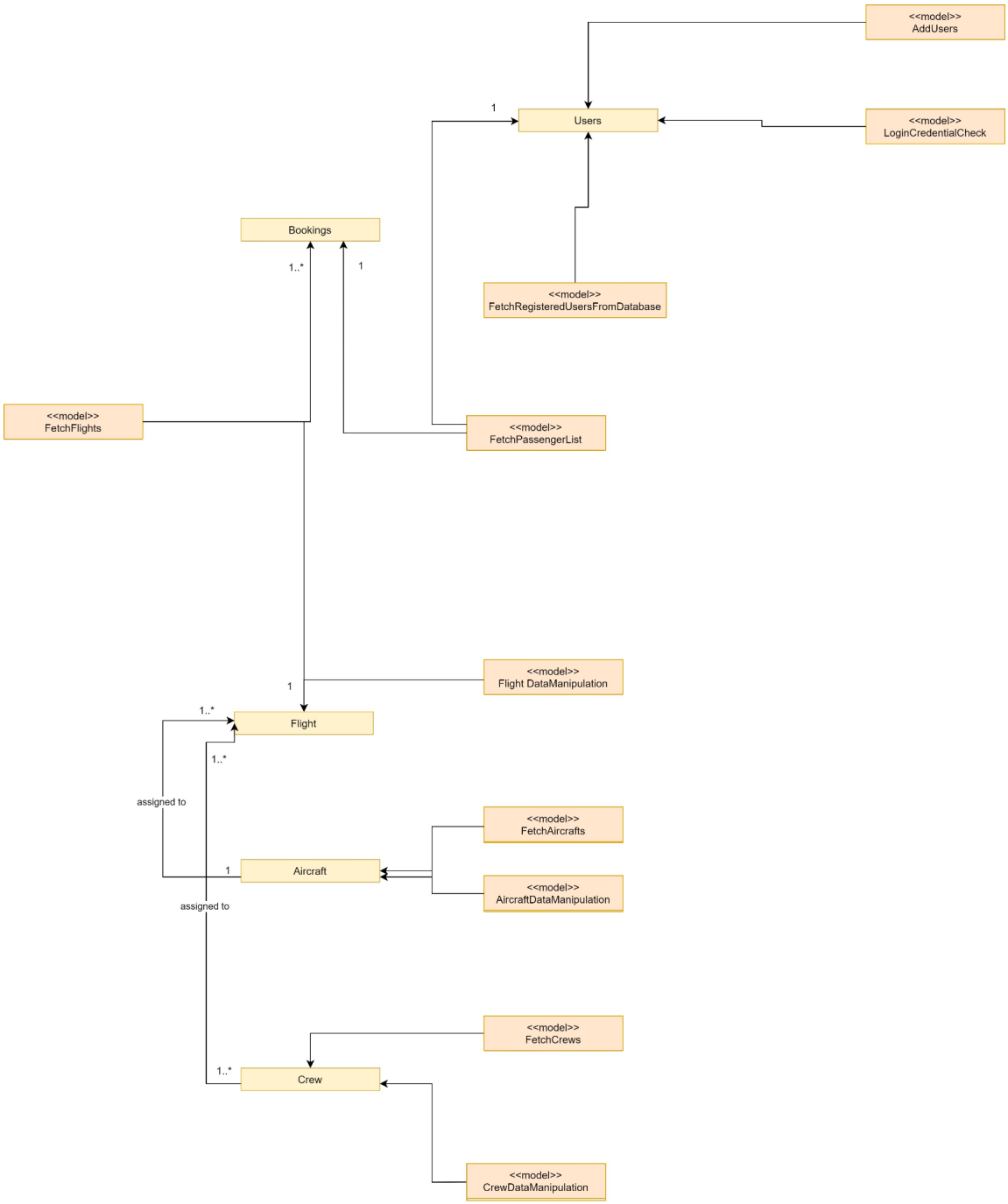
5.3. Crew Assigned to Flight Object



5.4. Payment



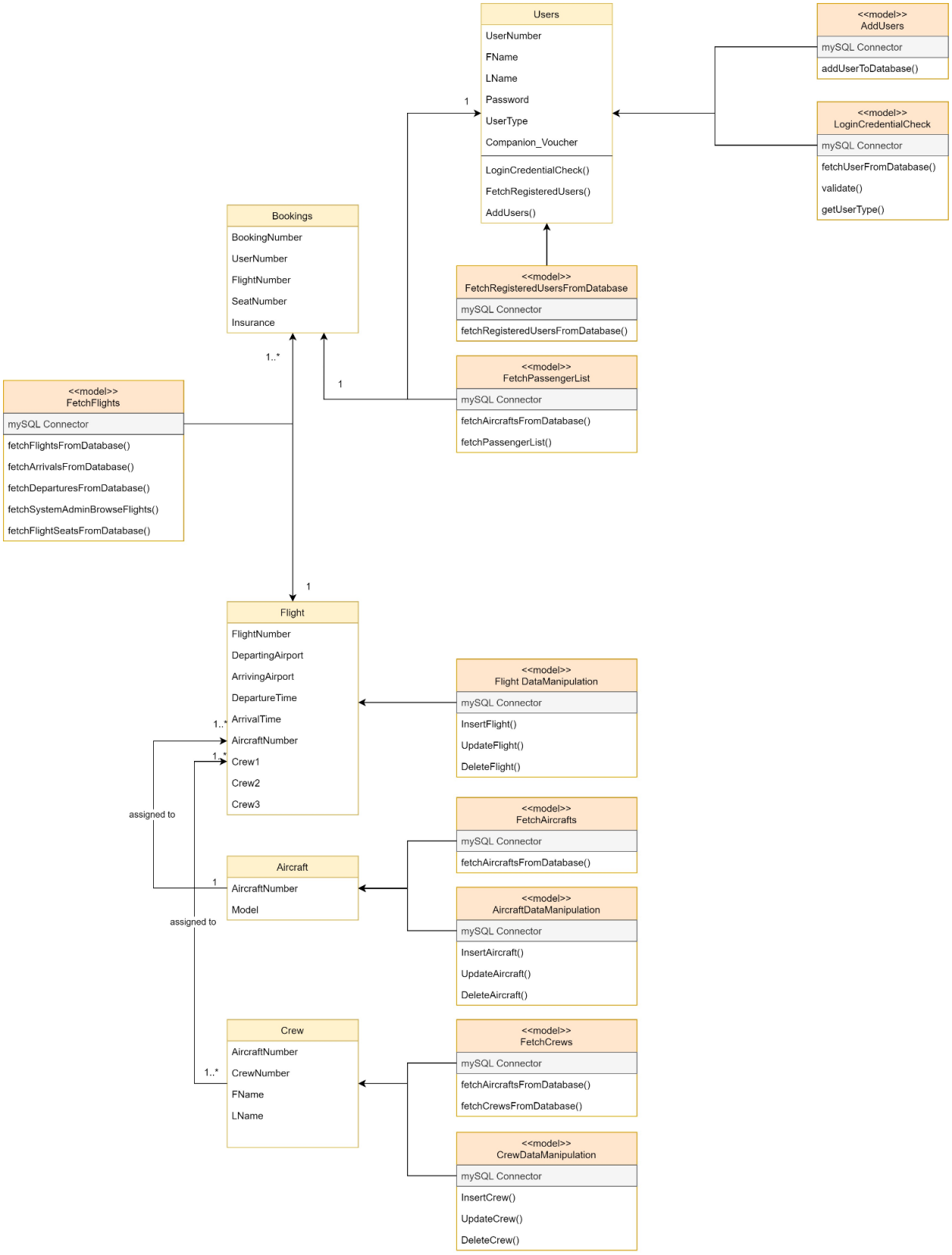
6. System's Domain Class Diagram



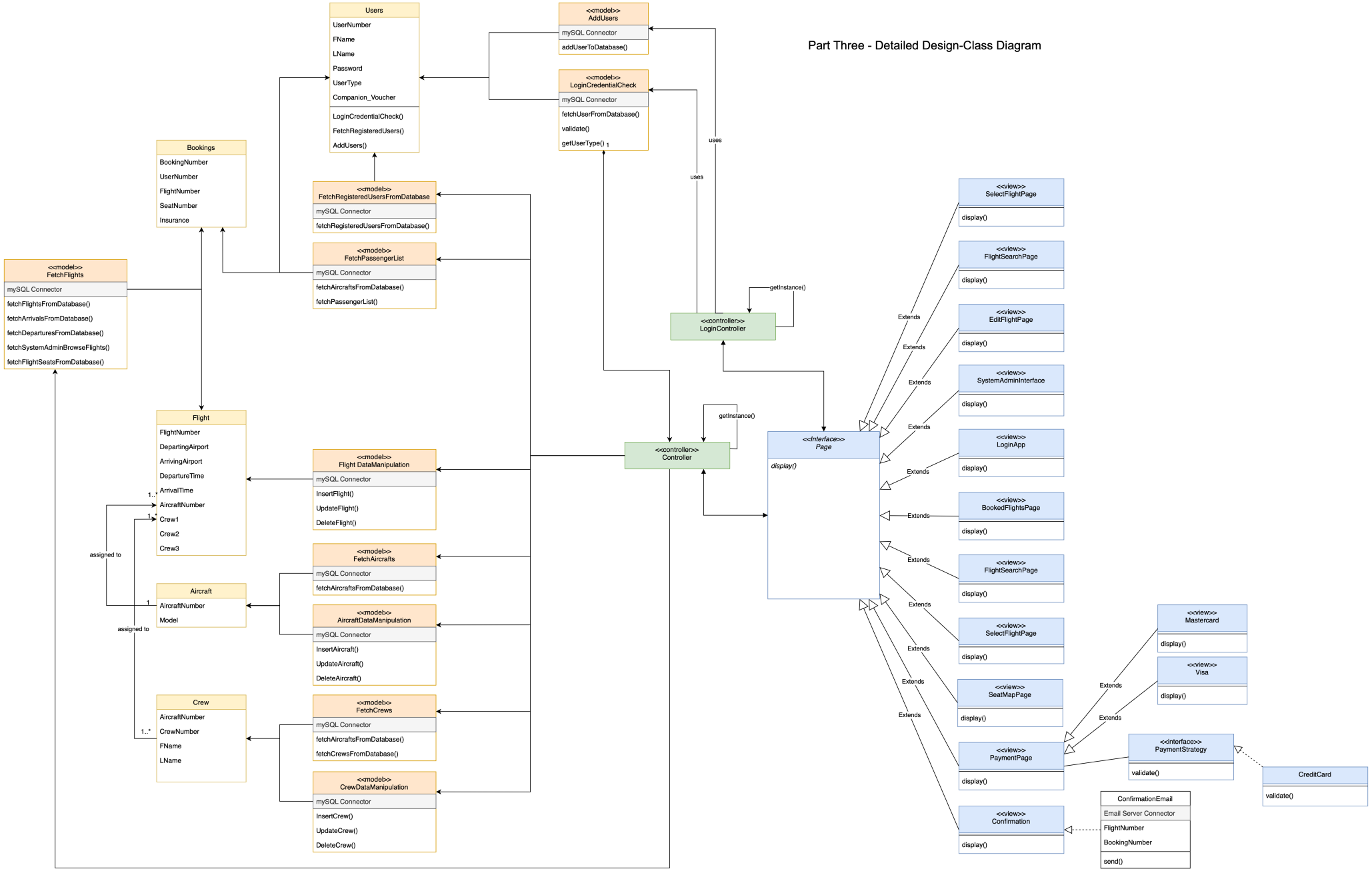
7. System's Domain Class Diagram

Design Patterns Used:

1. Singleton Pattern for sign-in controller object
2. Strategy Pattern for different forms of Payment (Visa and MasterCard)

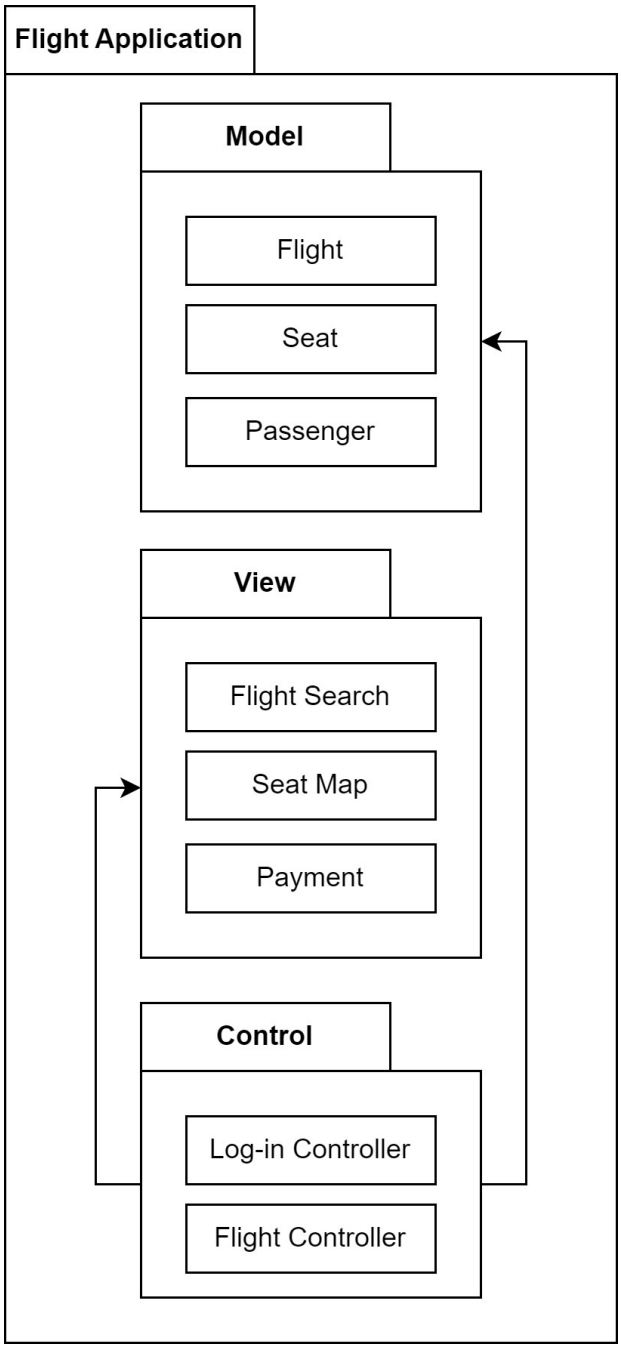

Users
UserNumber
FName
LName
Password
UserType
Companion_Voucher
LoginCredentialCheck()
FetchRegisteredUsers()
AddUsers()
<<model>>
AddUsers
mySQL Connector
addUserToDatabase()
<<model>>
LoginCredentialCheck
mySQL Connector
fetchUserFromDatabase()
validate()
getUserType()
Bookings
BookingNumber
UserNumber
FlightNumber
SeatNumber
Insurance
<<model>>
FetchRegisteredUsersFromDatabase
mySQL Connector
fetchRegisteredUsersFromDatabase()
<<model>>
FetchPassengerList
mySQL Connector
fetchAircraftsFromDatabase()
fetchPassengerList()
<<model>>
FetchFlights
mySQL Connector
fetchFlightsFromDatabase()
fetchArrivalsFromDatabase()
fetchDeparturesFromDatabase()
fetchSystemAdminBrowseFlights()
fetchFlightSeatsFromDatabase()
1
1..*
1
1
Flight
FlightNumber
DepartingAirport
ArrivingAirport
DepartureTime
ArrivalTime
AircraftNumber
Crew1
Crew2
Crew3
<<model>>
Flight DataManipulation
mySQL Connector
InsertFlight()
UpdateFlight()
DeleteFlight()
1..*
1..*
assigned to
<<model>>
FetchAircrafts
mySQL Connector
fetchAircraftsFromDatabase()
Aircraft
AircraftNumber
Model
1
<<model>>
AircraftDataManipulation
mySQL Connector
InsertAircraft()
UpdateAircraft()
DeleteAircraft()
assigned to
Crew
AircraftNumber
CrewNumber
FName
LName
1..*
<<model>>
FetchCrews
mySQL Connector
fetchAircraftsFromDatabase()
fetchCrewsFromDatabase()
<<model>>
CrewDataManipulation
mySQL Connector
InsertCrew()
UpdateCrew()
DeleteCrew()

Part Three - Detailed Design-Class Diagram

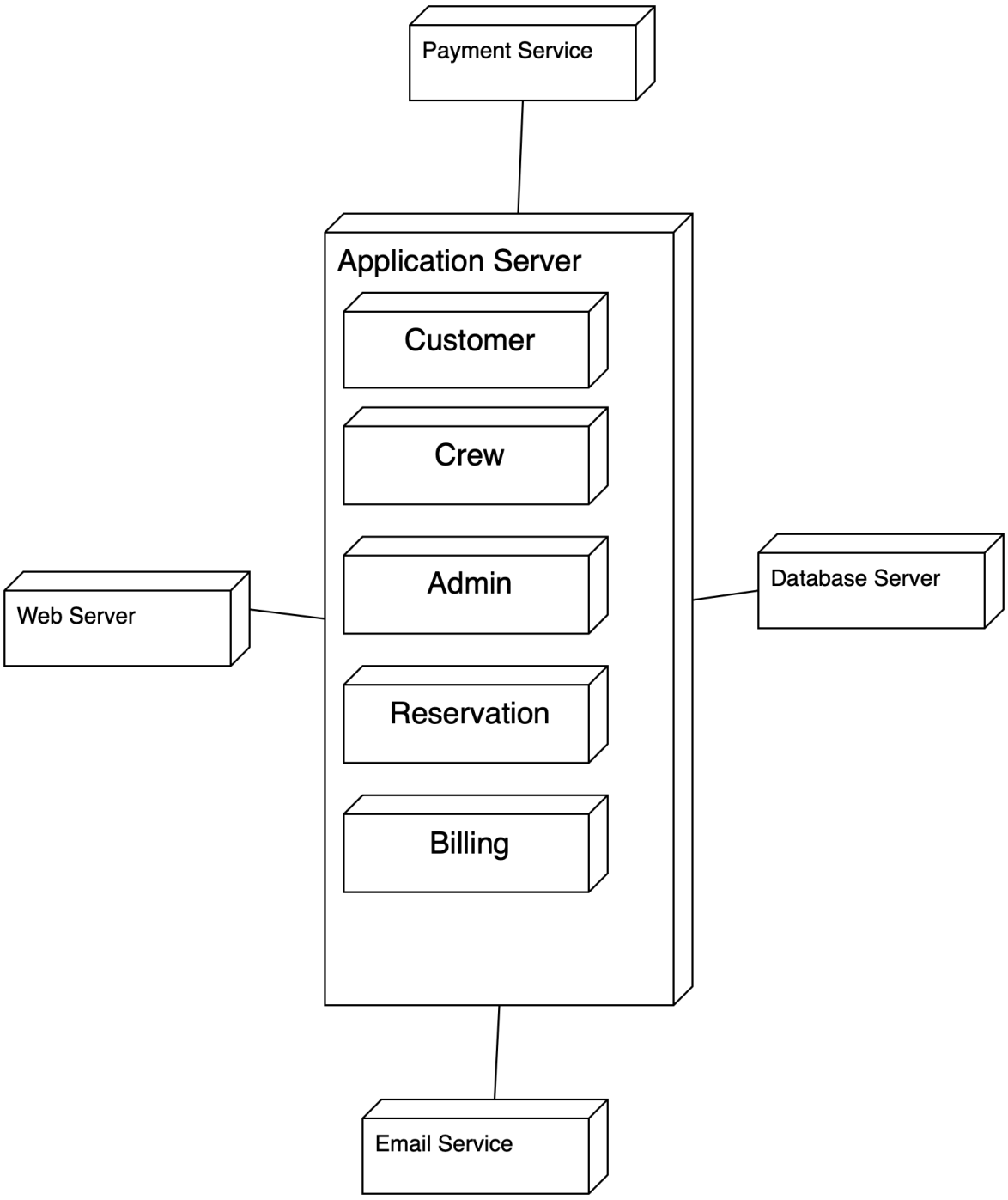


Part Four - High-Level System Architecture

1. Package Diagram



2. Deployment Diagram



Implementation Phase

To explain the implementation of our project, we have created a video that discusses the below information in detail.

Link to video on UCalgary OneDrive

First Part

1. Which part of the implementation was done by each team member.
   We all contributed to the implementation.

2. What is the distribution percentage of each member:
   Jenn Bushey 25%
   Robby Parmar 25%
   Harshil Patel 25%
   Benjamin Reid 25%

Second Part

In this part of the video, you should precisely show how well your design class diagram matches your java code.
We have implemented a controller that controls the flow of views that are enabled from user interactions. Depending on what the user requests, the controller will fetch information from the model class and display or insert it into the database.

Third Part

The project is implemented as a mobile app using Java Swing to implement the graphical user interface. Java connects to MySQL using the JDBC driver.

Fourth Part

In this part, you need to explain clarify which one of the following requirements is implemented, and which one is not:

- Main/initial GUI that allows selecting options:
  - Start as a regular user.
  - Start as a registered user.

    - Start as a system admin(bonus).
- Browse/Search for flights.
- Select flight.
- Browse graphics seat map.
- Make payment.
- Cancel flight by user and sending cancellation notification.
- Browse passenger list only by (staff and not regular or registered users).
- Payment and Ticket notification by email (bonus)
- Selecting seats from a seat map, by a mouse-click on an available seat.
- Make payment for ticket and receipt notification.
- Ticket cancellation and refund notification.
- Sending receipt and cancellation email (bonus).
- User registration for registered users.
- User login only of registered user and/or staff/admin.
- Manage promotion for registered users.
- Admin duties: maintain aircrafts, flights, crews, etc. (bonus).
- use of design patterns.

We implemented all of the above requirements into our Flight Application.

Fifth Part

In this part, you should run your application and show:
How each requirement listed above works.

Accessing Flight Application:

To run the jar file, direct your command prompt (Windows) or terminal (Mac) to the folder containing the jar file and run the following command:

```
java -jar FinalProjectGroup19Jar.jar
```

You will need your mySQL root password as the first thing the program asks for is your username (root) and password.

Login Information:

Registered User

Email: john.doe@email.com
Password: 1111

System Admin

Email: emma.wilson@email.com
Password: 1111

Agent

Email: liam.johnson@email.com
Password: 1111