ENSF 480 Term Project Design Phase

Part A - System Analysis

1. System's Description

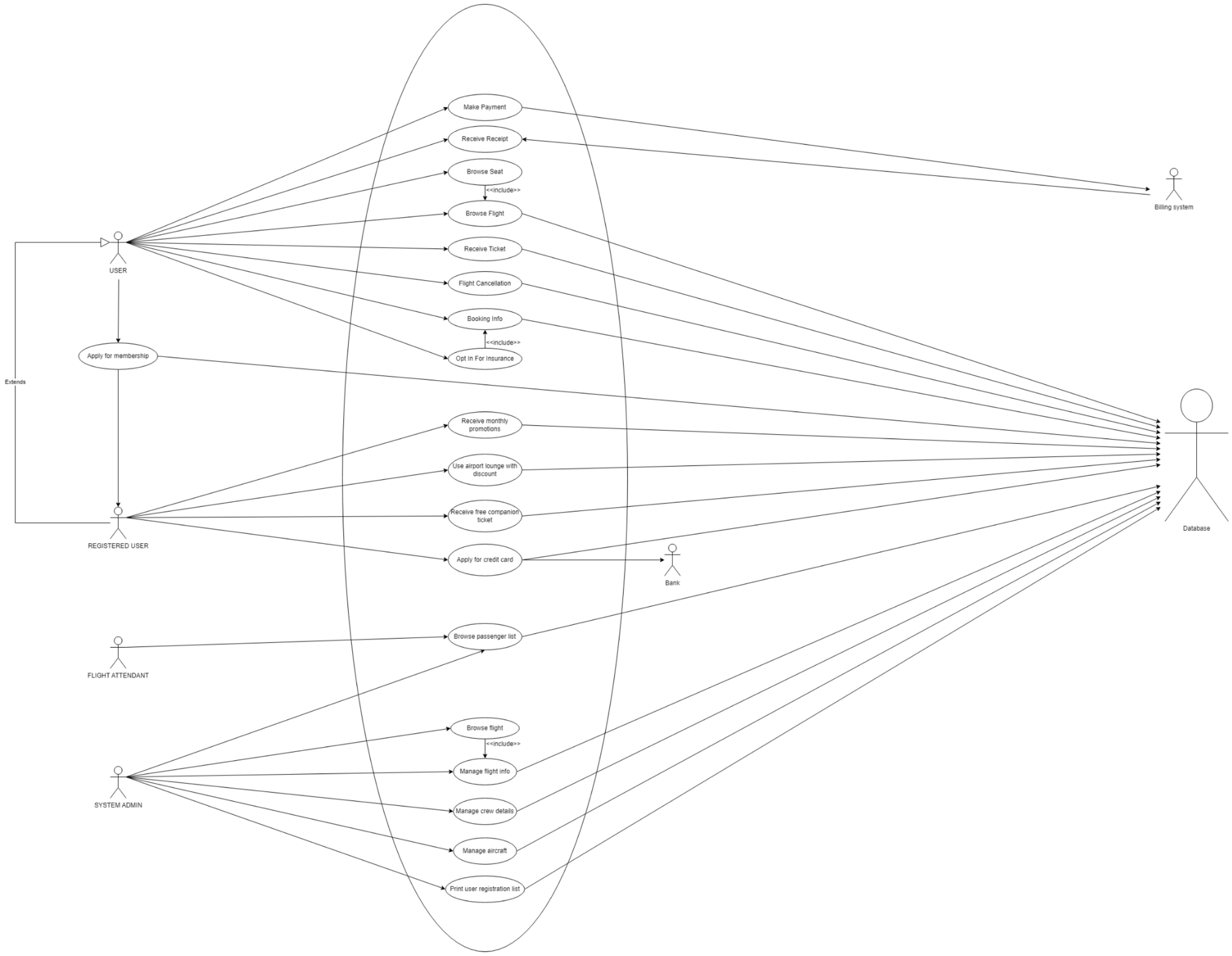
Product Perspective:

The Flight Reservation Web Application is a comprehensive online platform designed to streamline and enhance the air travel experience for various user roles within a single airline company. The goal of the system is to give travelers, flight crew and system administrators an easy to use and seamless platform for managing and booking flights. The system aims to optimize the reservation process and enhance the overall travel experience for all users involved.

Product Features:

- User Authentication and Authorization
- Flight Search and Browse
- Seat Map Visualization
- Seat Selection
- Ticket Cancellation Insurance Option
- Secure Payment Processing
- E-ticket and Receipt Generation
- Flight Cancellation
- Passenger List Viewing for Airline Agents
- Aircraft List Management for Admins
- Crew Management for Admins
- Airport Lounge Access Management for Registered Users
- Monthly Promotion News for Registered Users
- Free Companion Ticket Management for Registered Users
- User Registration and Profile Management
- Credit Card Application Process
- Dynamic Pricing Management
- Promotional Offer Management
- User History and Preferences Tracking

2. System’s Use-Case diagram



3. System’s scenarios for each use case

Browse Flight (User, Database):

- User access the system
- User searches for desired destination
- System displays available flights from the database
- User browses the list and selects desired flight

Browse Seat (User):

- User selects desired seat type from ordinary, comfort and business class seats
- System searches the database for the seat info for the selected flight
- System displays the seat info graphically as a seat map
- User selects their desired seat

Opt In For Insurance (User):

- System asks user to opt in for ticket cancellation insurance
- User decide whether he wants insurance or not
- System stores User option in Booking Info

Make Payment (User, Billing System):

- System displays cost to the User
- User selects payment option
- User enters payment information
- Payment information is sent to the billing system

Receive Receipt (User, Billing System):

- Billing system generates receipt for ticket purchase
- Billing system sends receipt to user via email

Receive Ticket (User, Database):

- System retrieves information from database
- System creates ticket
- User receives ticket via email

Flight Cancellation (User, Database, Billing System):

- User cancels selected flight
- System checks whether user opted in for cancellation insurance
- If User has cancellation Insurance billing system refunds ticket price
- System removes booking info from database
- User receives cancellation confirmation via email

Booking Info (User, Database):

- System asks user for information
- User inputs name, email address, and phone number
- System stores cancellation insurance information
- System sends the booking information to the database

Use Airport Lounge (User, Billing System):

- System displays cost to the User
- User selects payment option
- User enters payment information
- Payment information is sent to the billing system

Apply For Membership (User, Registered User, Database):

- User applies for membership
- Provides name, address, phone number, and email address
- The system stores user information in the database
- User becomes registered user

Receive Monthly Promotions (Registered User, Database):

- System retrieves registered emails from the database
- System retrieves the monthly newsletter
- Email the news letter to all the registered user at the start of the month

Airport Lounge Discount (Registered User, Database, Billing System):

- Registered user enters information to receive discount
- System confirms registered user is on the database
- System applies discount to the cost
- Payment information is sent to the billing system

Apply For Credit Card (Registered User, Database, Bank):

- Registered user applies for credit card
- System retrieves registered user information from database
- System sends registered User information to the bank to make a company card
- System updates registered user information in the database

Login (Registered User, Flight Attendant, System Admin, Database):

- Registered users, flight attendants, and system admits navigate to login area on website
- User inputs their login information and the it is checked with the login information stored in the database
- If valid login information is inputted, user is granted login, along with special privileges for registered users, flight attendants and system admins

Logout (Registered User, Flight Attendant, System Admin):

- Logged in users (registered users, flight attendants, system admin) navigate to logout area of website
- Special privileges for registered users, flight attendants, and system admins is removed, website with no special features is displayed

Receive Free Companion Ticket (Registered User, Database):

- System checks database to see if a registered user hasn't redeemed a companion ticket in the past year
- System gives registered user free companion ticket
- Registered user applies free companion ticket while booking flight

Browse Passenger List (Flight Attendant, System Admin, Database):

- System admin/flight attendant searches for passenger list for a specific flight
- System retrieves passenger list for the flight from the database
- System displays passenger list to the system admin/ flight attendant

Browse For Flight (System Admin, Database):

- System displays list of existing flights from the database
- System admin can browse for a specific flight using its name, destination, origin, or date of departure
- System admin can choose to manage the flight information for that specific flight

Manage Flight Information (System Admin):

- System admin can view the list of crew members on the flight and remove them or add new crew members
- System admin can change aircraft used for the flight
- System admin can change flight destination
- System admin can change flight origin
- System admin can change the date of departure for the flight

Manage Crew Details (System Admin, Database):

- System displays list of existing crew members hired by the company from the database
- System admin can filter the crew list based on the flight they are on or their name
- System admin can add or remove a crew member from the crew list stored on the database

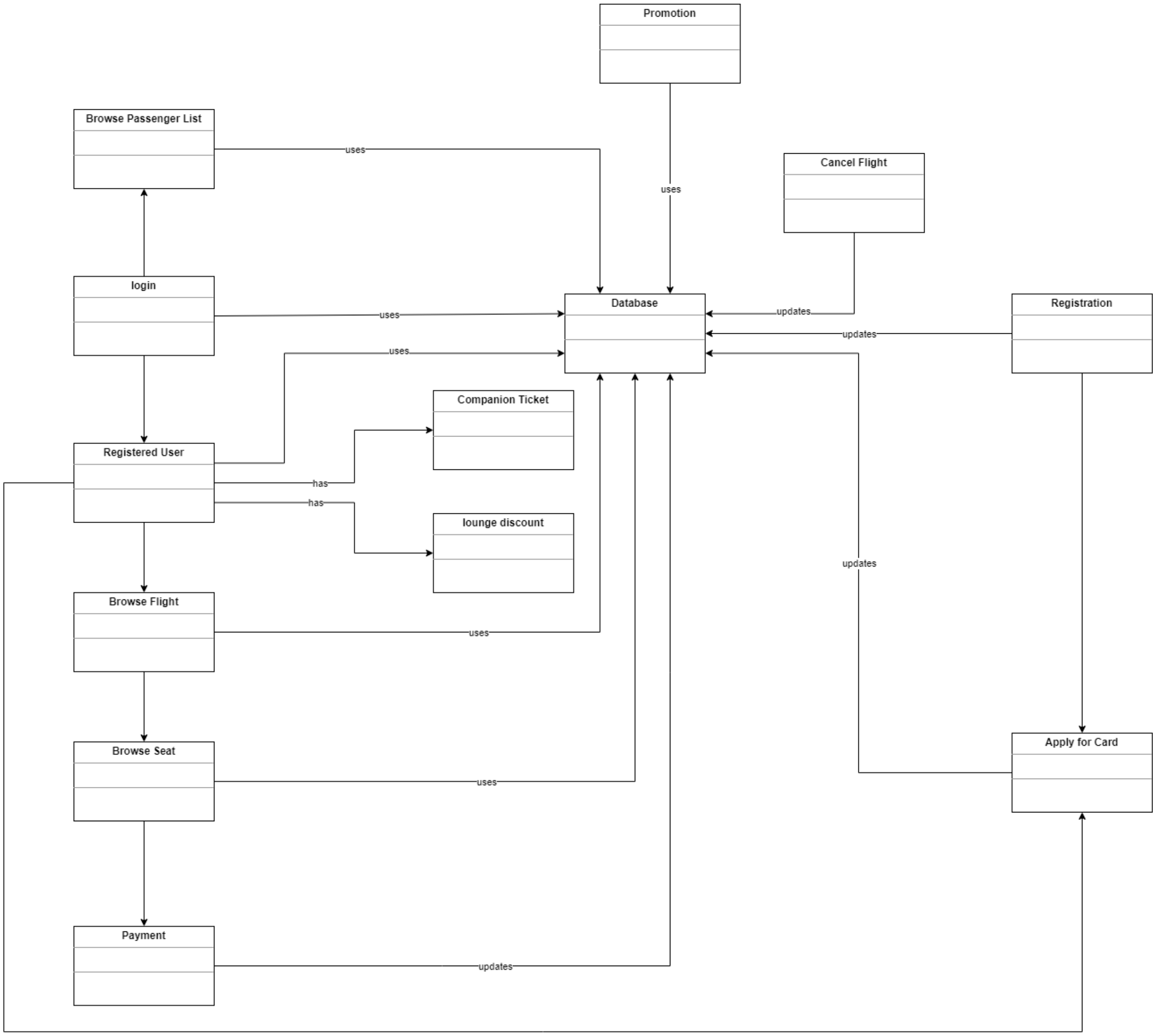
Manage Aircraft (System Admin, Database):

- System displays list of existing aircrafts owned by the company from the database
- System admin can filter the aircraft list based on the flight or aircraft type
- System admin can add or remove aircrafts from the aircraft list stored on the database

Print Registered User List (System Admin, Database):

System admin asks the system for the registered user list
System retrieves the list from the database and admin prints that list

4. **System’s conceptual model**



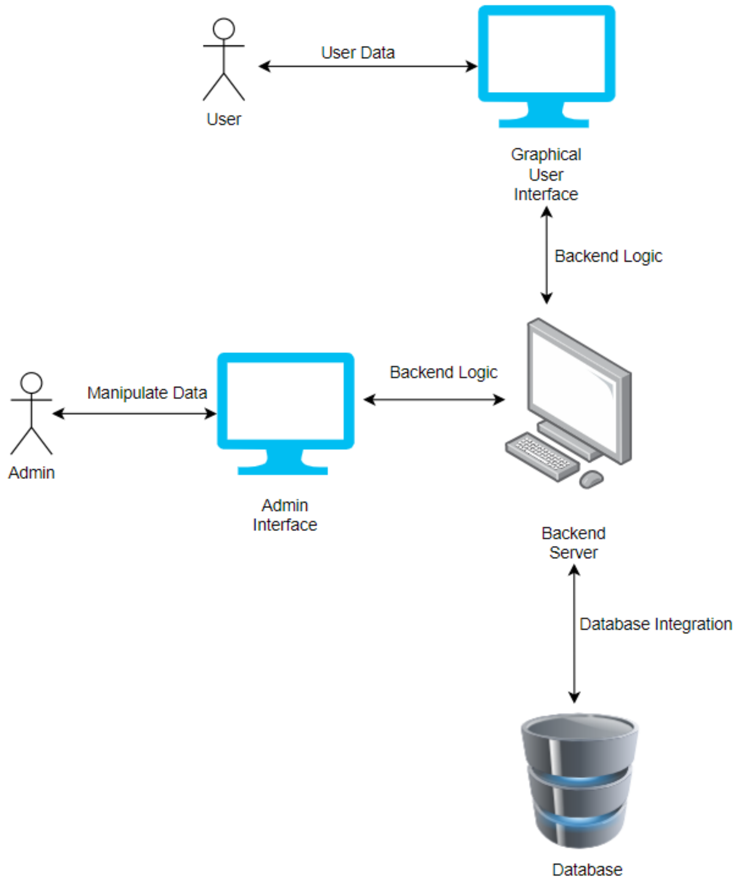
Part B - Domain Diagrams

1. **Highlights of System’s Architecture**

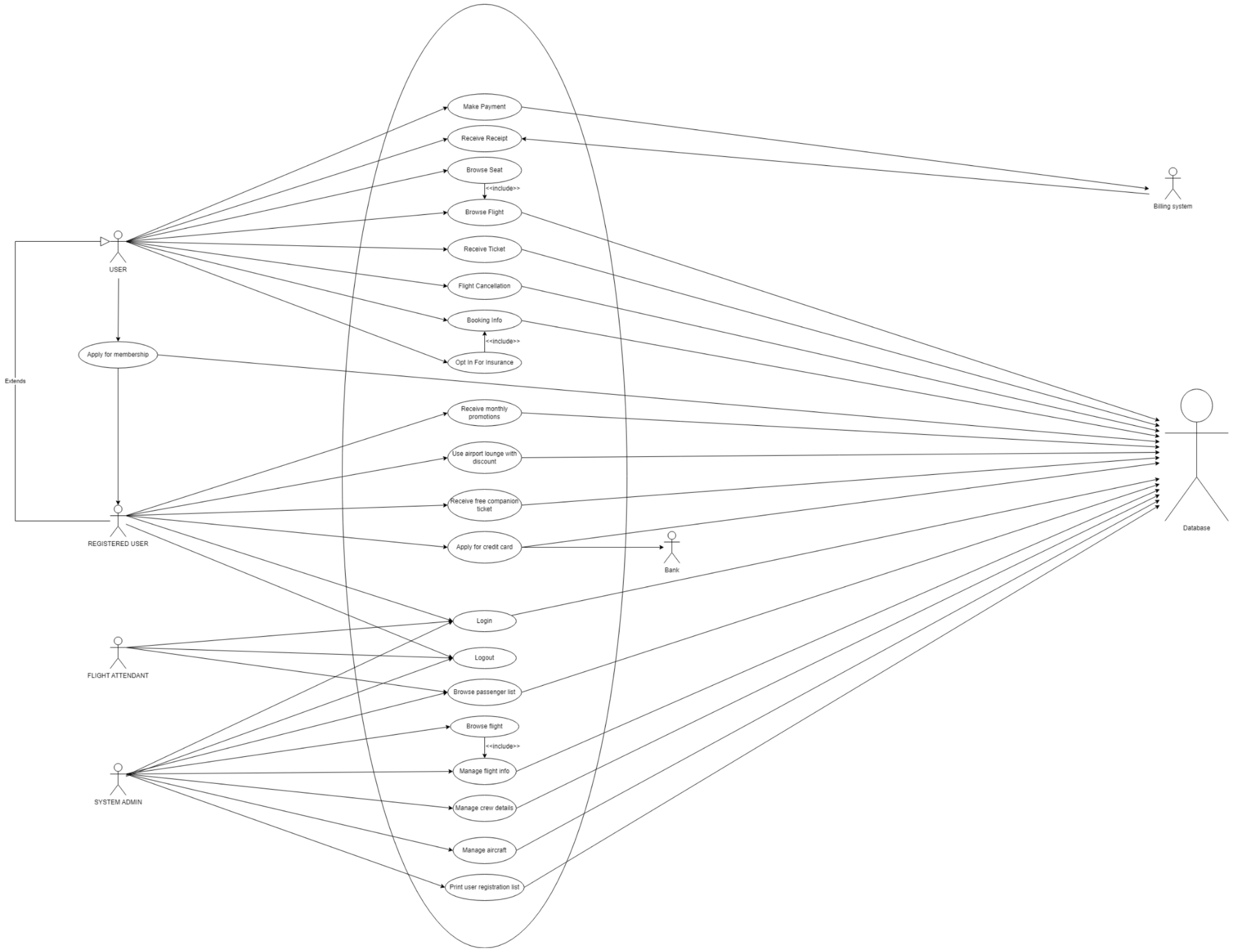
Highlights of System’s Architecture

The architecture employs a client-server architecture with a client-side graphical interface, and a server-side backend implemented in java handling logic, data management, communication with database, etc. The system has an integrated relational database implemented in MySql which stores and retrieves basic information. Both users and admins can change the database; users when they register with the airline or book a flight, and admins who can add, delete, or edit the varying fields of information stored on the database, such as airplanes, flights, usernames and passwords, etc.

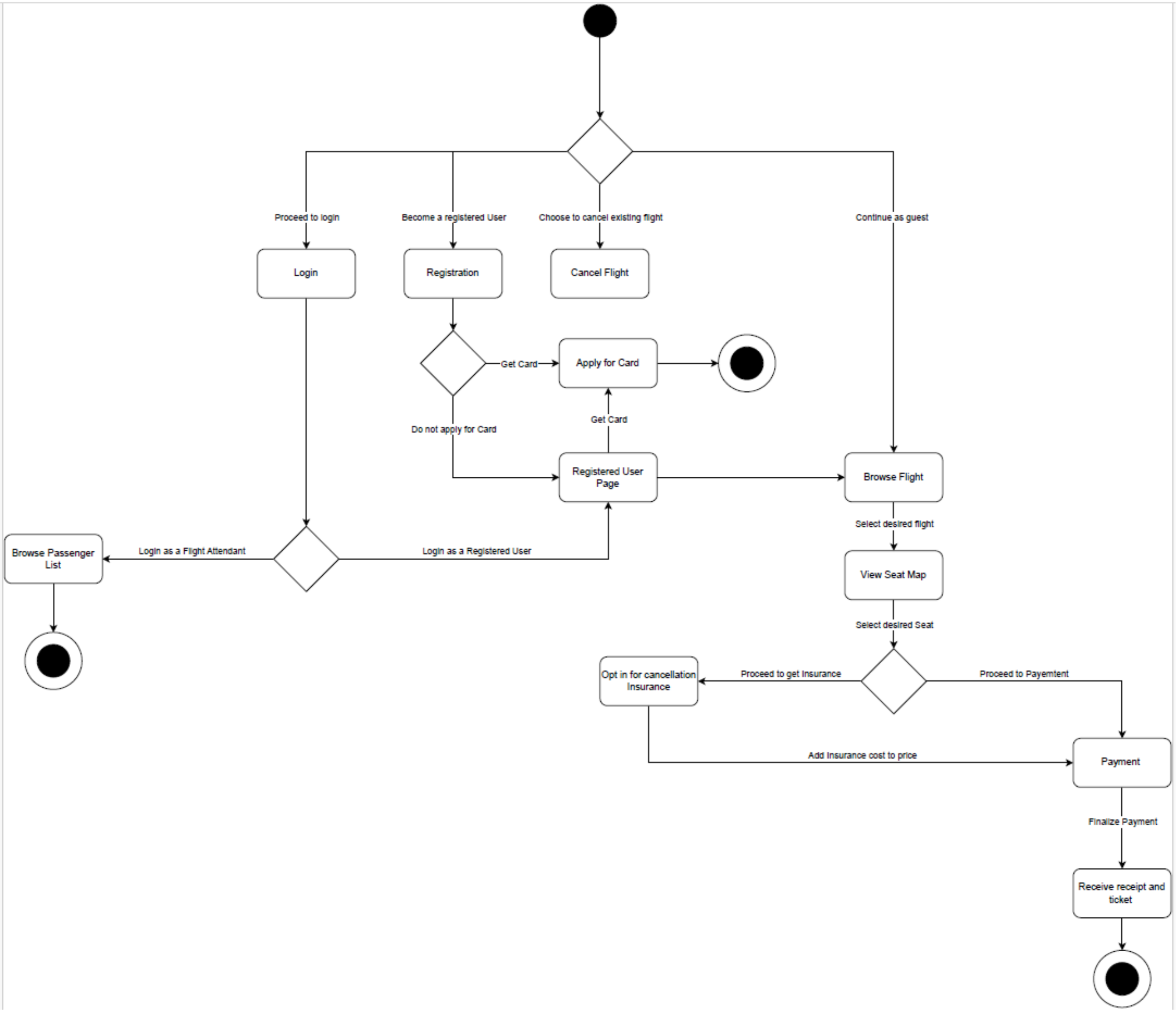
The program is implemented with 2 design patterns in mind; a singleton pattern for the database and decorator for seats. The singleton pattern for the database is to ensure we only create one instance of the database, as it is unnecessary to create more than one instance of the same data, and the decorator is to allow us to create different types of seats (ordinary, business, comfort) efficiently.



2. Updated Use-case

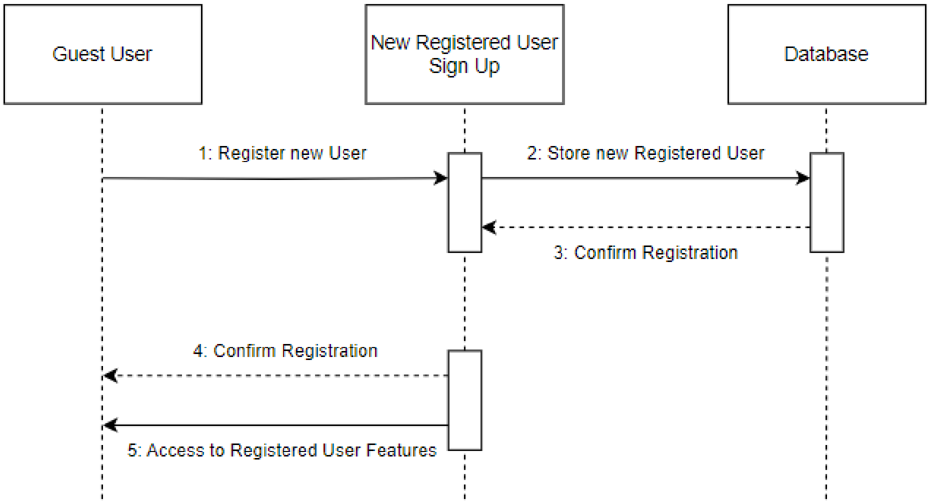


3. Systems Activity Diagram

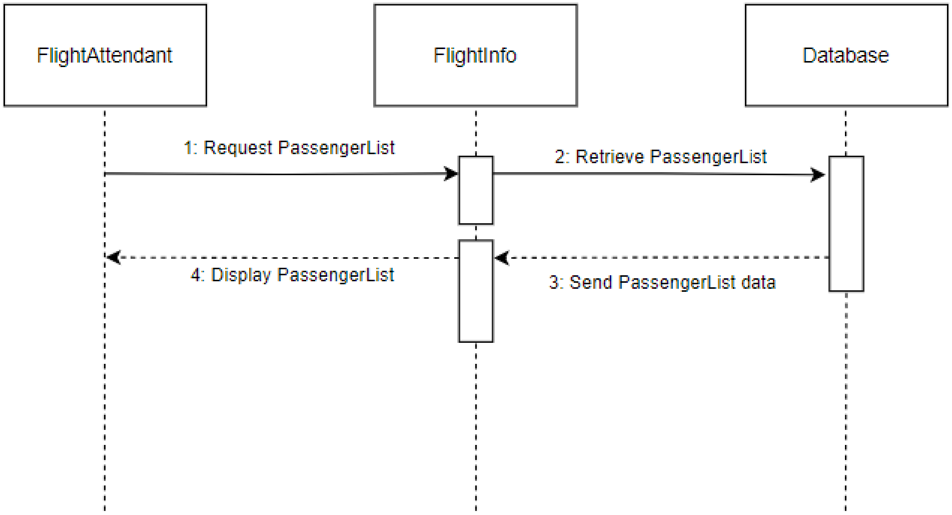


4. Sequence Diagram for 4 Use-Cases

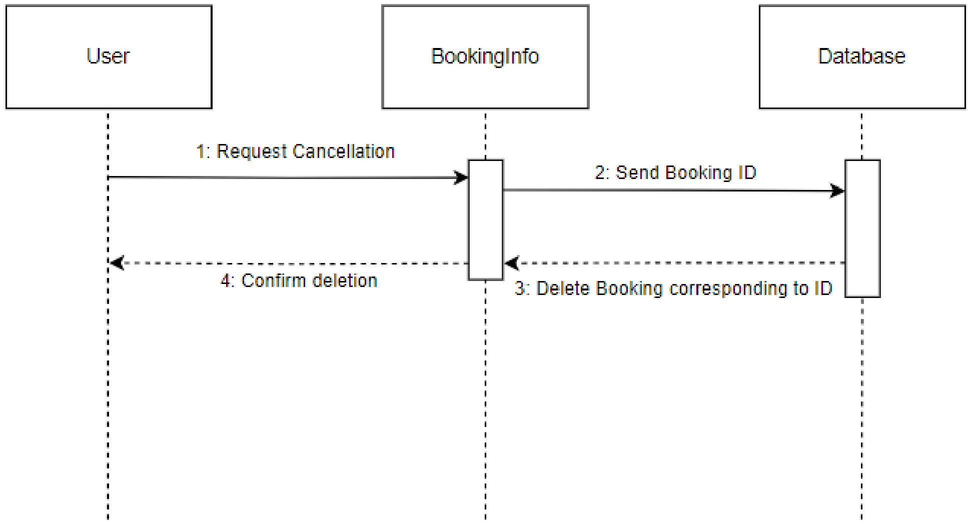
Register New User


Guest User
New Registered User Sign Up
Database
1: Register new User
2: Store new Registered User
3: Confirm Registration
4: Confirm Registration
5: Access to Registered User Features

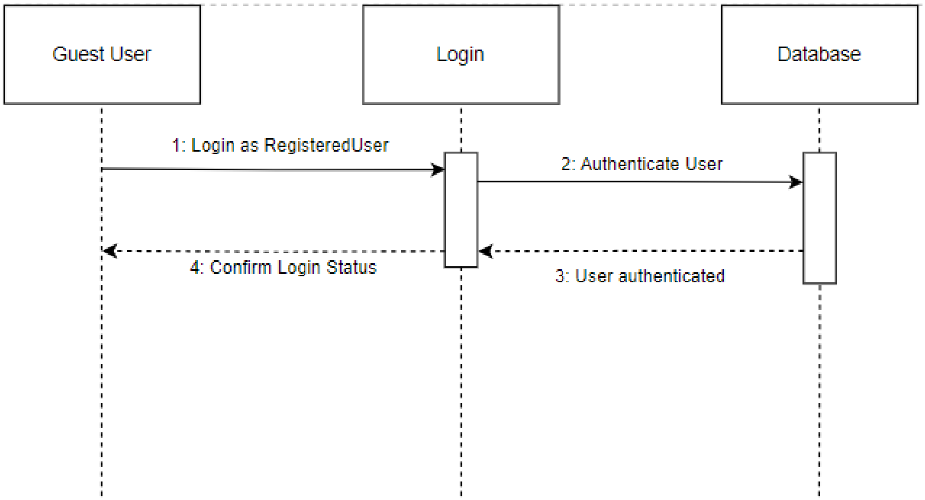
Browse PassengerList



Flight Cancellation

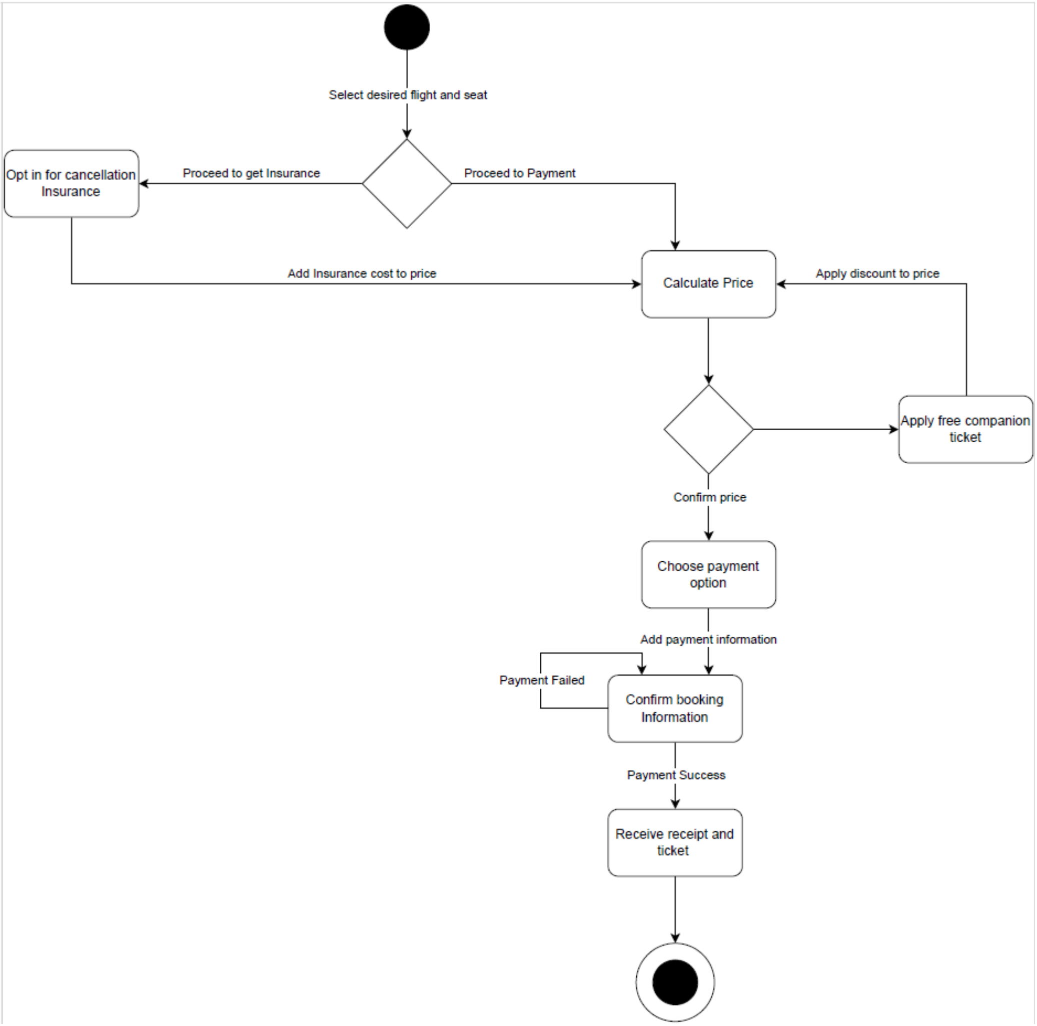


Login System

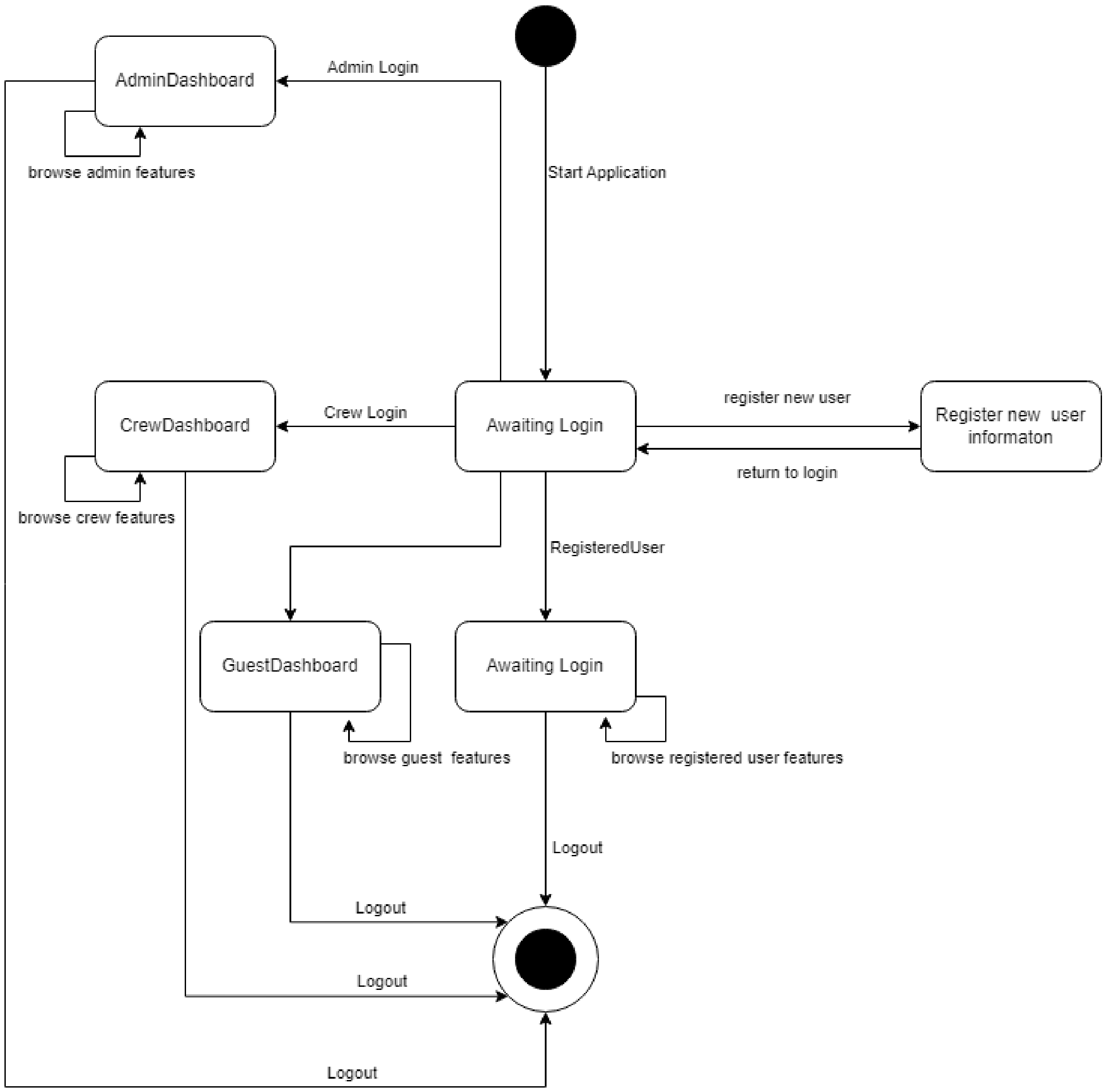


5. State Transition Diagram for 4 GUI Objects (1 per page)

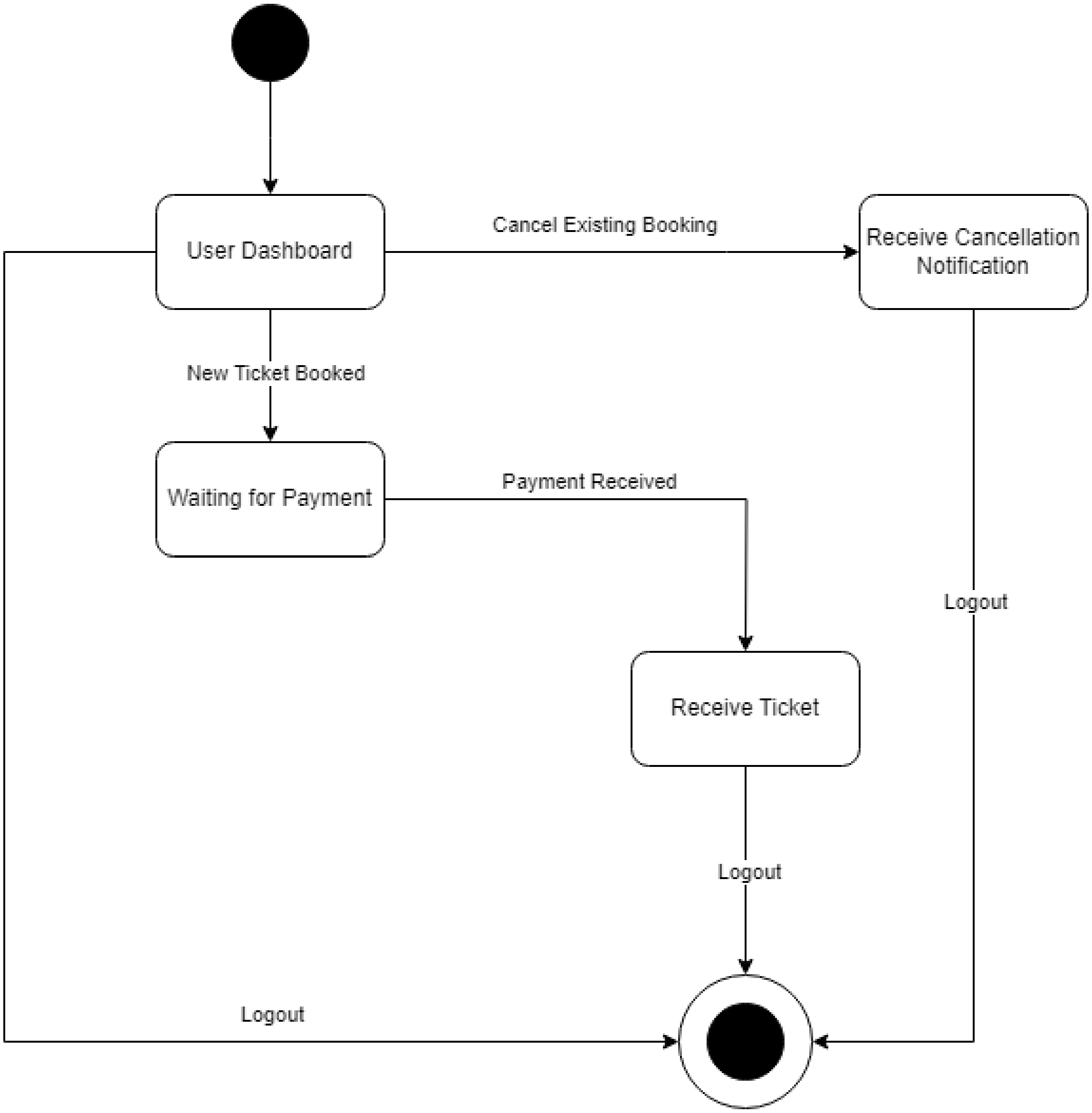
Payment:


Select desired flight and seat
Proceed to get Insurance
Proceed to Payment
Opt in for cancellation Insurance
Add Insurance cost to price
Calculate Price
Apply discount to price
Apply free companion ticket
Confirm price
Choose payment option
Add payment information
Payment Failed
Confirm booking Information
Payment Success
Receive receipt and ticket

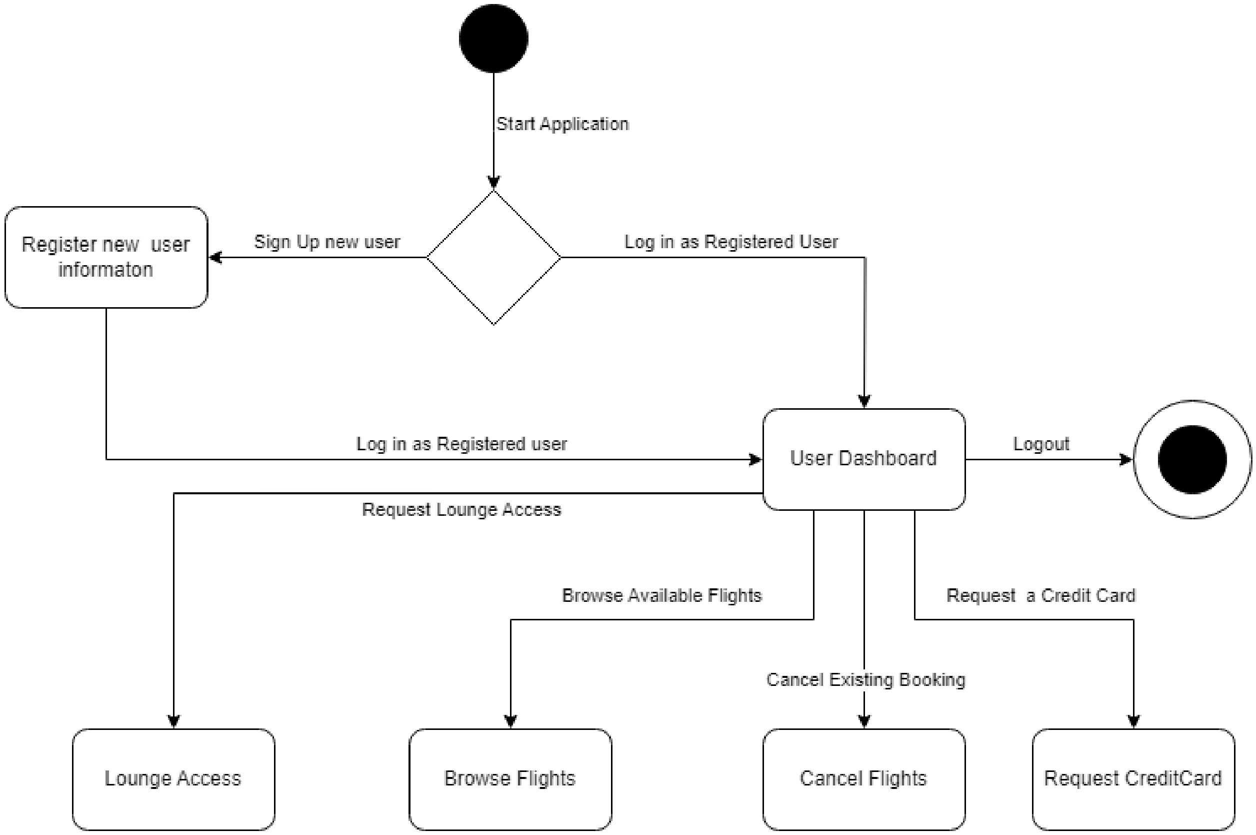
Login:


AdminDashboard
Admin Login
browse admin features
Start Application
CrewDashboard
Crew Login
Awaiting Login
register new user
Register new user informaton
return to login
browse crew features
RegisteredUser
GuestDashboard
Awaiting Login
browse guest features
browse registered user features
Logout
Logout
Logout
Logout

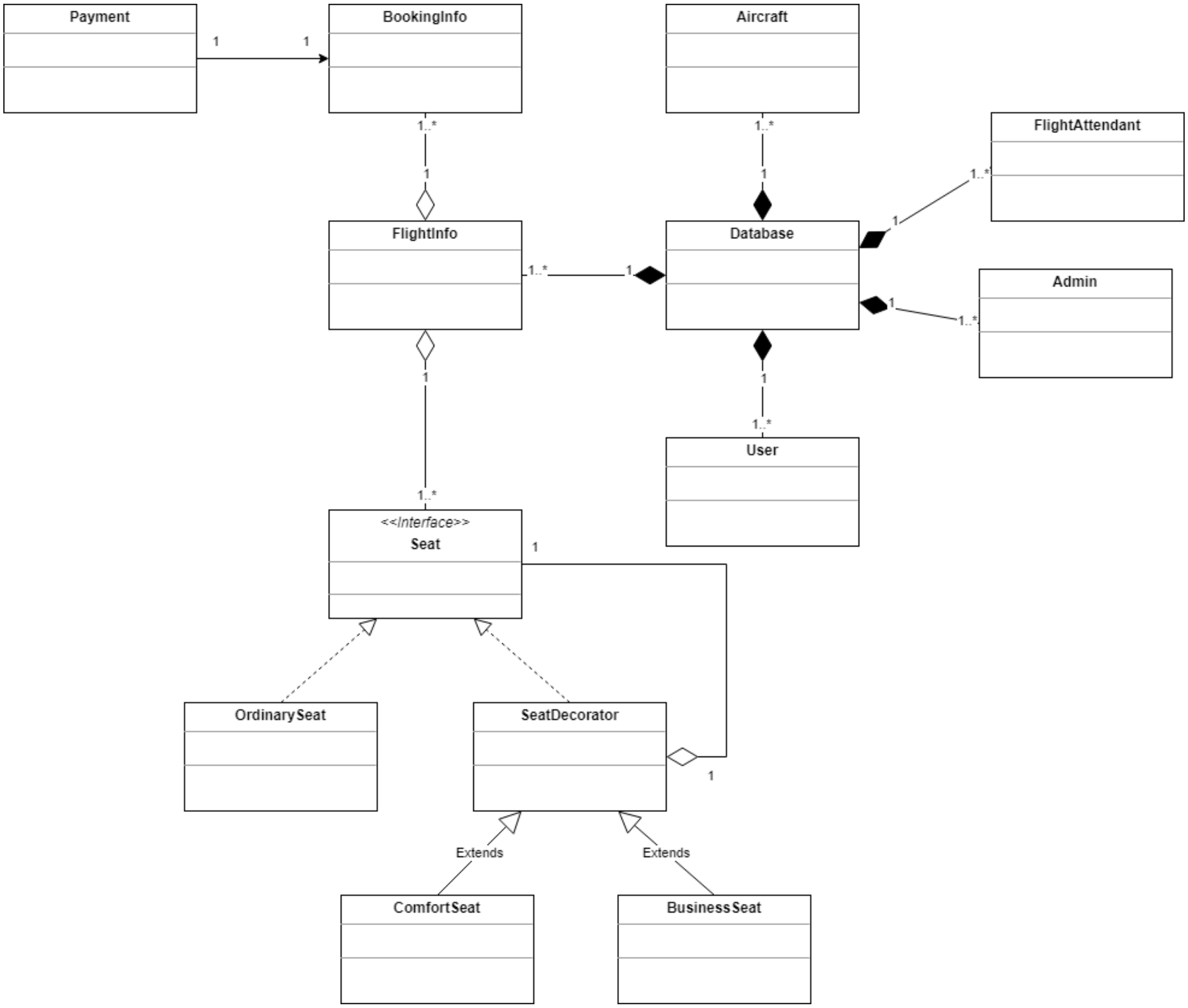
Cancel/Payment Received:



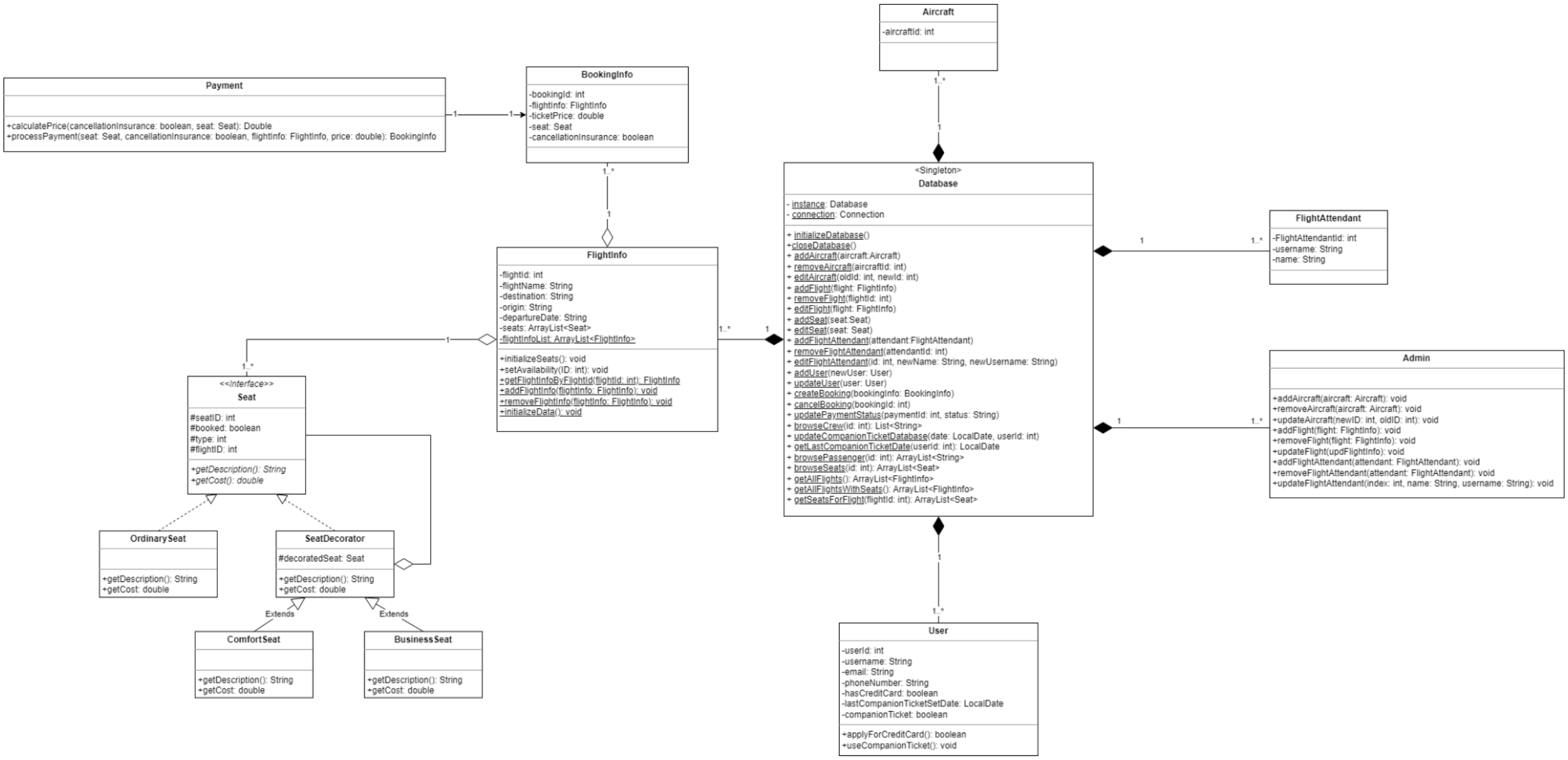
User Features:



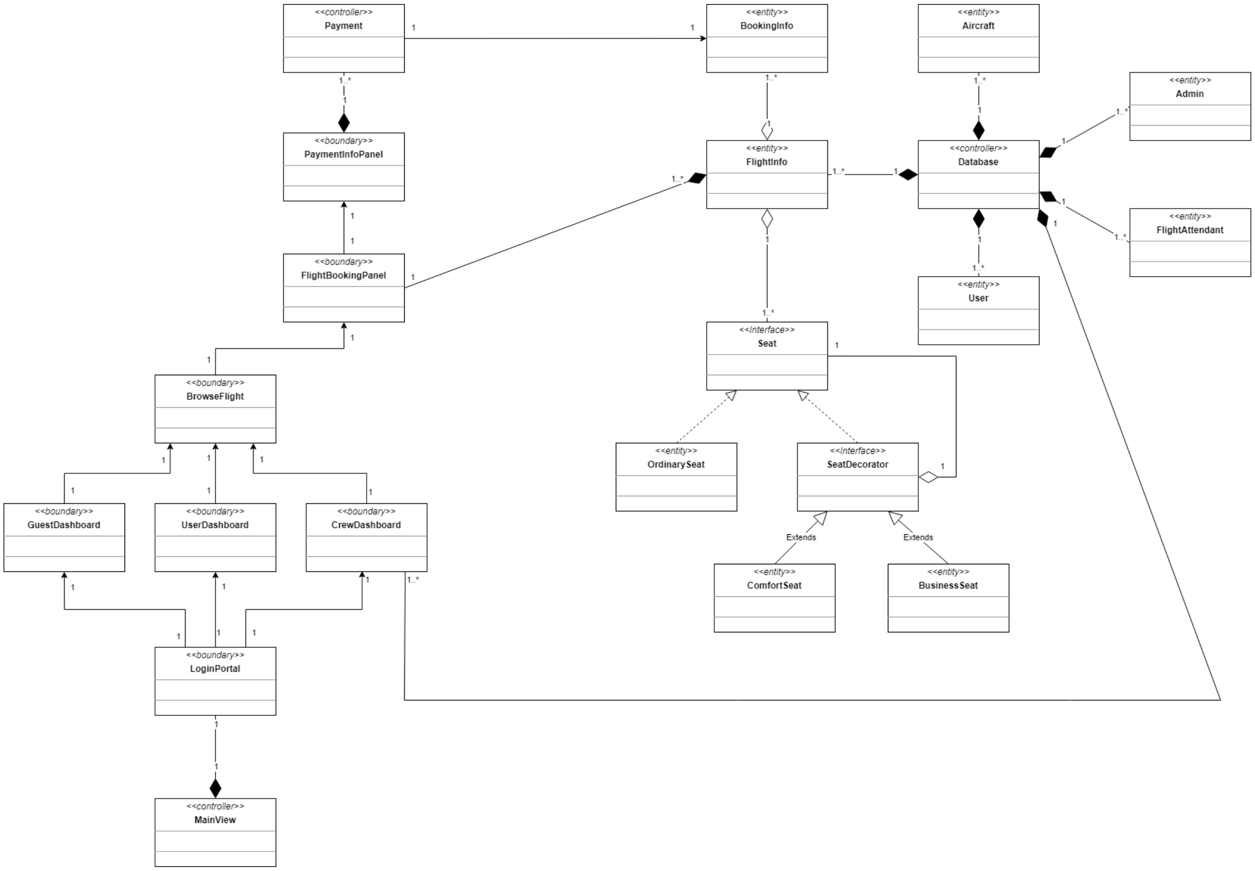
6. System's Domain Class Diagram (no attributes, only relationships and multiplicities)



7. System’s Domain Class Diagram (with attributes, functionalities, and relationships/multiplicities)

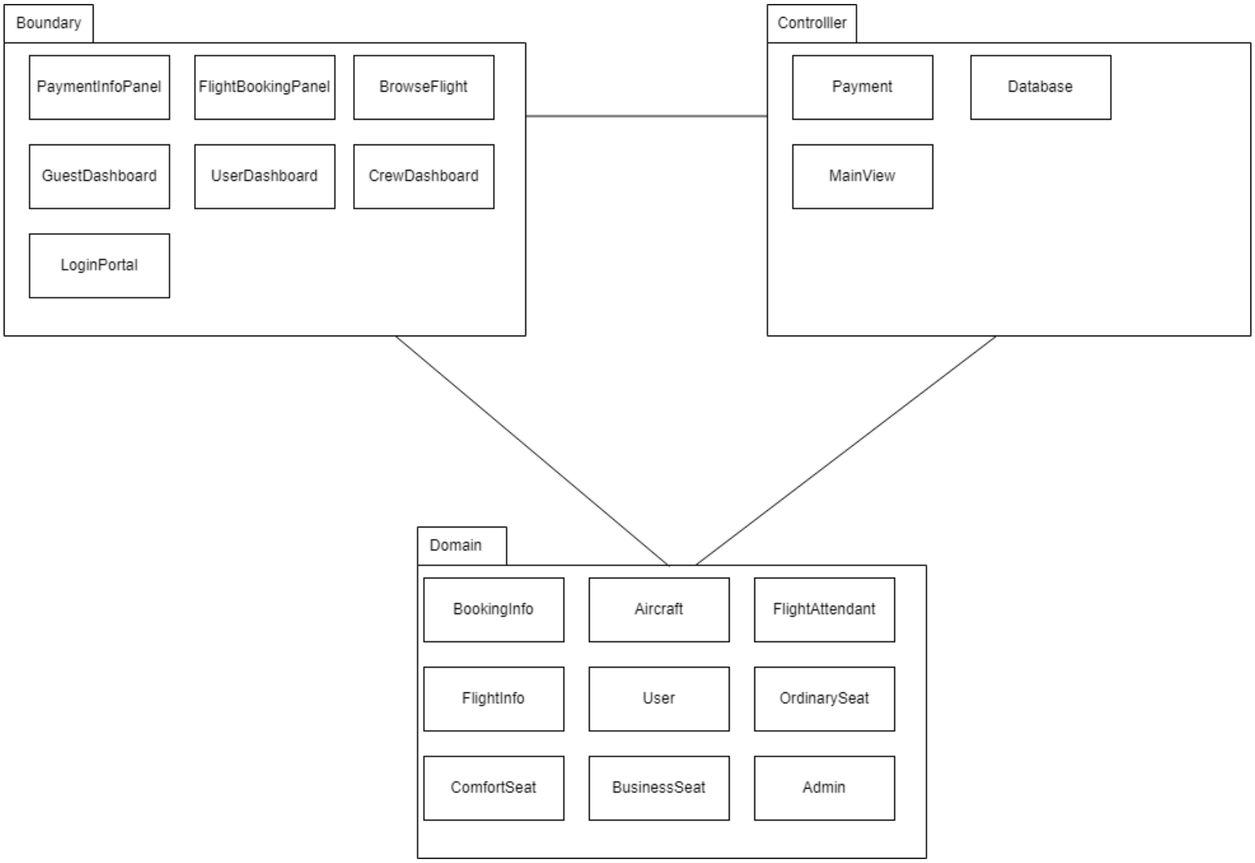

Aircraft
-aircraftId: int
1..*
1
<Singleton>
Database
- instance: Database
- connection: Connection
+ initializeDatabase()
+closeDatabase()
+ addAircraft(aircraft:Aircraft)
+ removeAircraft(aircraftId: int)
+ editAircraft(oldId: int, newId: int)
+ addFlight(flight: FlightInfo)
+ removeFlight(flightId: int)
+ editFlight(flight: FlightInfo)
+ addSeat(seat:Seat)
+ editSeat(seat: Seat)
+ addFlightAttendant(attendant:FlightAttendant)
+ removeFlightAttendant(attendantId: int)
+ editFlightAttendant(id: int, newName: String, newUsername: String)
+ addUser(newUser: User)
+ updateUser(user: User)
+ createBooking(bookingInfo: BookingInfo)
+ cancelBooking(bookingId: int)
+ updatePaymentStatus(paymentId: int, status: String)
+ browseCrew(id: int): List<String>
+ updateCompanionTicketDatabase(date: LocalDate, userId: int)
+ getLastCompanionTicketDate(userId: int): LocalDate
+ browsePassenger(id: int): ArrayList<String>
+ browseSeats(id: int): ArrayList<Seat>
+ getAllFlights(): ArrayList<FlightInfo>
+ getAllFlightsWithSeats(): ArrayList<FlightInfo>
+ getSeatsForFlight(flightId: int): ArrayList<Seat>
1
1..*
FlightAttendant
-FlightAttendantId: int
-username: String
-name: String
1
1..*
Admin
+addAircraft(aircraft: Aircraft): void
+removeAircraft(aircraft: Aircraft): void
+updateAircraft(newID: int, oldID: int): void
+addFlight(flight: FlightInfo): void
+removeFlight(flight: FlightInfo): void
+updateFlight(updFlightInfo): void
+addFlightAttendant(attendant: FlightAttendant): void
+removeFlightAttendant(attendant: FlightAttendant): void
+updateFlightAttendant(index: int, name: String, username: String): void
1
1..*
User
-userId: int
-username: String
-email: String
-phoneNumber: String
-hasCreditCard: boolean
-lastCompanionTicketSetDate: LocalDate
-companionTicket: boolean
+applyForCreditCard(): boolean
+useCompanionTicket(): void
Payment
+calculatePrice(cancellationInsurance: boolean, seat: Seat): Double
+processPayment(seat: Seat, cancellationInsurance: boolean, flightInfo: FlightInfo, price: double): BookingInfo
1
1
BookingInfo
-bookingId: int
-flightInfo: FlightInfo
-ticketPrice: double
-seat: Seat
-cancellationInsurance: boolean
1..*
1
FlightInfo
-flightId: int
-flightName: String
-destination: String
-origin: String
-departureDate: String
-seats: ArrayList<Seat>
-flightInfoList: ArrayList<FlightInfo>
+initializeSeats(): void
+setAvailability(ID: int): void
+getFlightInfoByFlightId(flightId: int): FlightInfo
+addFlightInfo(flightInfo: FlightInfo): void
+removeFlightInfo(flightInfo: FlightInfo): void
+initializeData(): void
1..*
1
1
1..*
<<Interface>>
Seat
#seatID: int
#booked: boolean
#type: int
#flightID: int
+getDescription(): String
+getCost(): double
OrdinarySeat
+getDescription(): String
+getCost: double
SeatDecorator
#decoratedSeat: Seat
+getDescription(): String
+getCost: double
Extends
Extends
ComfortSeat
+getDescription(): String
+getCost: double
BusinessSeat
+getDescription(): String
+getCost: double

Part C - System's Detailed Design-Class Diagram

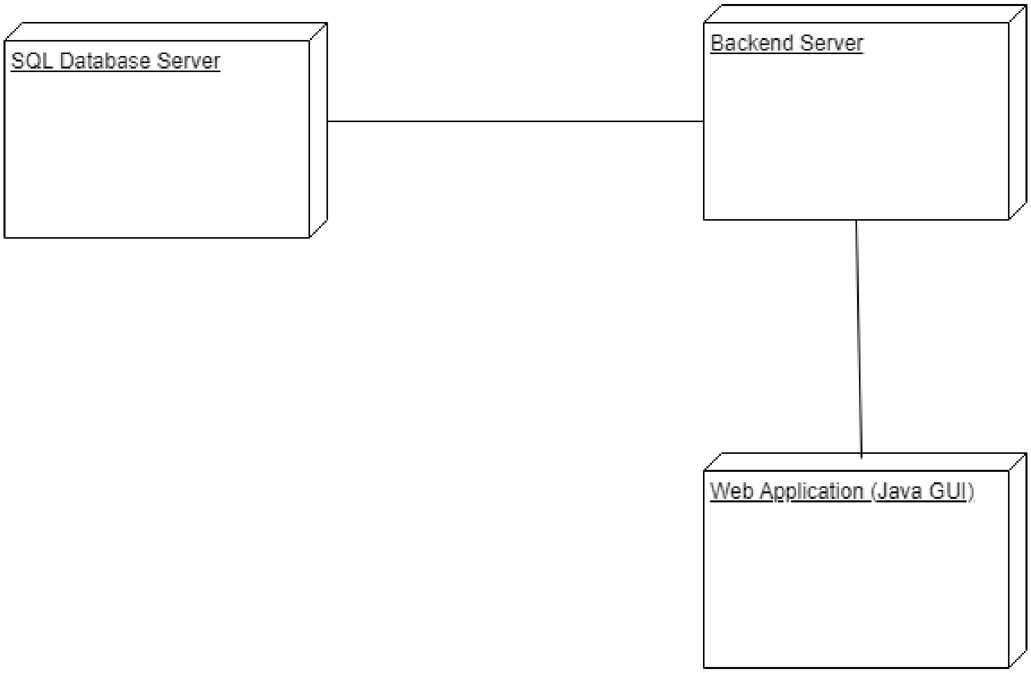


Part D - High-Level System’s Architecture

Package Diagram:



Deployment Diagram:


SQL Database Server
Backend Server
Web Application (Java GUI)