ENSF480 - L02

Air Travel Reservation Application Design Documentation

Group 32

Haseeb Tahir	10190846
Shaheer Shakir	30171272
Abdul Shakoor Raed	30173834
Muhammed Umar Khan	30142631

System Description

The Flight Reservation System is a GUI-based application that allows a Customer and a Flight Agent to search for flights, view flight details, make reservations, and manage existing bookings. Users enter the system through a Login UI, and after successful authentication the system displays a role-specific menu in the Main UI. All persistent data (users, flights, reservations, payments, and notifications) is stored in the Database and accessed through the system's backend components.

The system also supports administrative operations. A System Administrator can manage flight records by adding, updating, deleting, and viewing flight schedules. In addition, the system interacts with external actors: a Payment System processes credit/debit payments, and a Clock/Timer System triggers monthly promotion notifications to users.

Major / More Complex Functionalities

1) Browsing and Selecting a Flight (Search + View Details)

> A User (Customer or Flight Agent) opens the flight search screen in the GUI and enters search criteria such as origin, destination, date, and airline. The System retrieves matching flights from the Database and displays a list of results. The user can then select a flight to view more detailed information. The system loads the selected flight's details from the database and displays them in the GUI. The user may proceed to reservation/booking or return to search.

2) Booking a Flight Ticket (Reservation Creation + Confirmation)

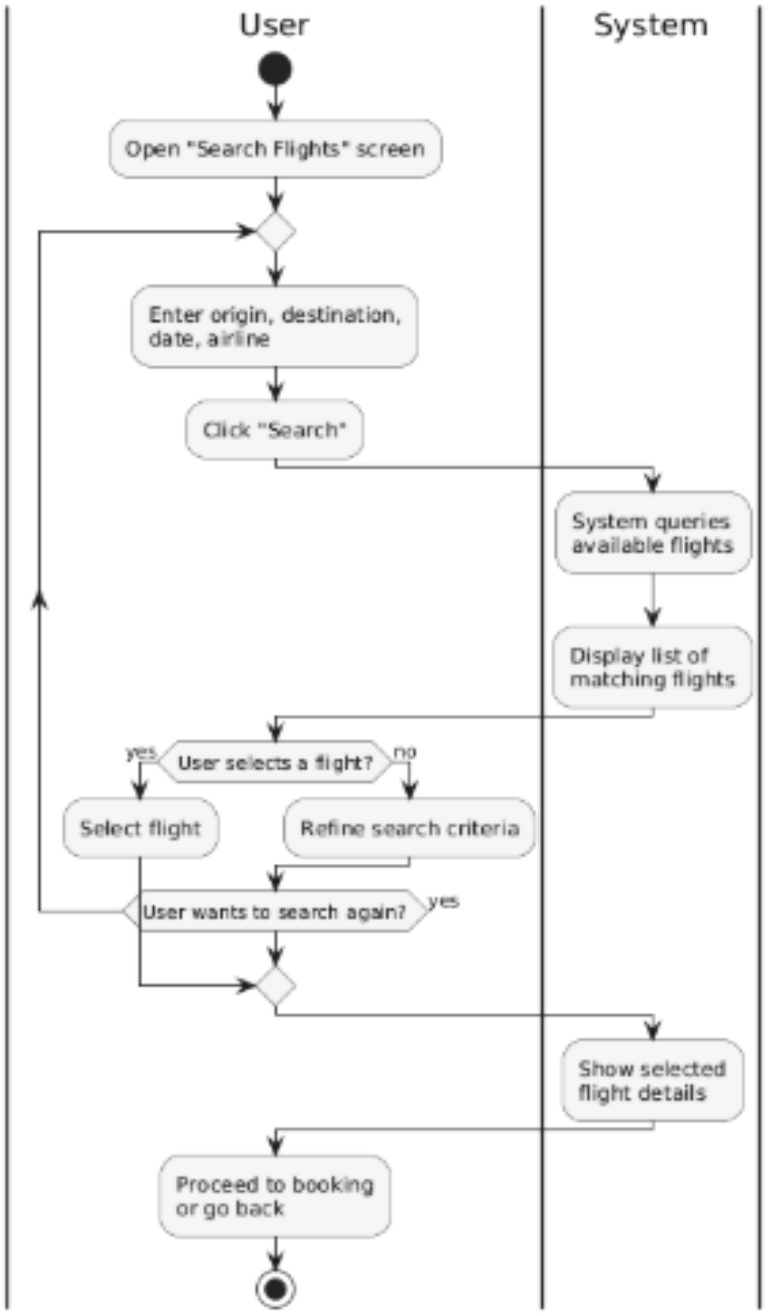
> After selecting a flight, the user enters passenger information and seat count through the Booking UI. The System validates the input and, if valid, creates a Reservation in a "Pending" state and presents a reservation summary to the user. The booking process then proceeds to payment. When payment is confirmed, the system updates the reservation status and generates a booking confirmation for the user.

3) Making Payment (Authorization + Recording + Receipt/Confirmation)

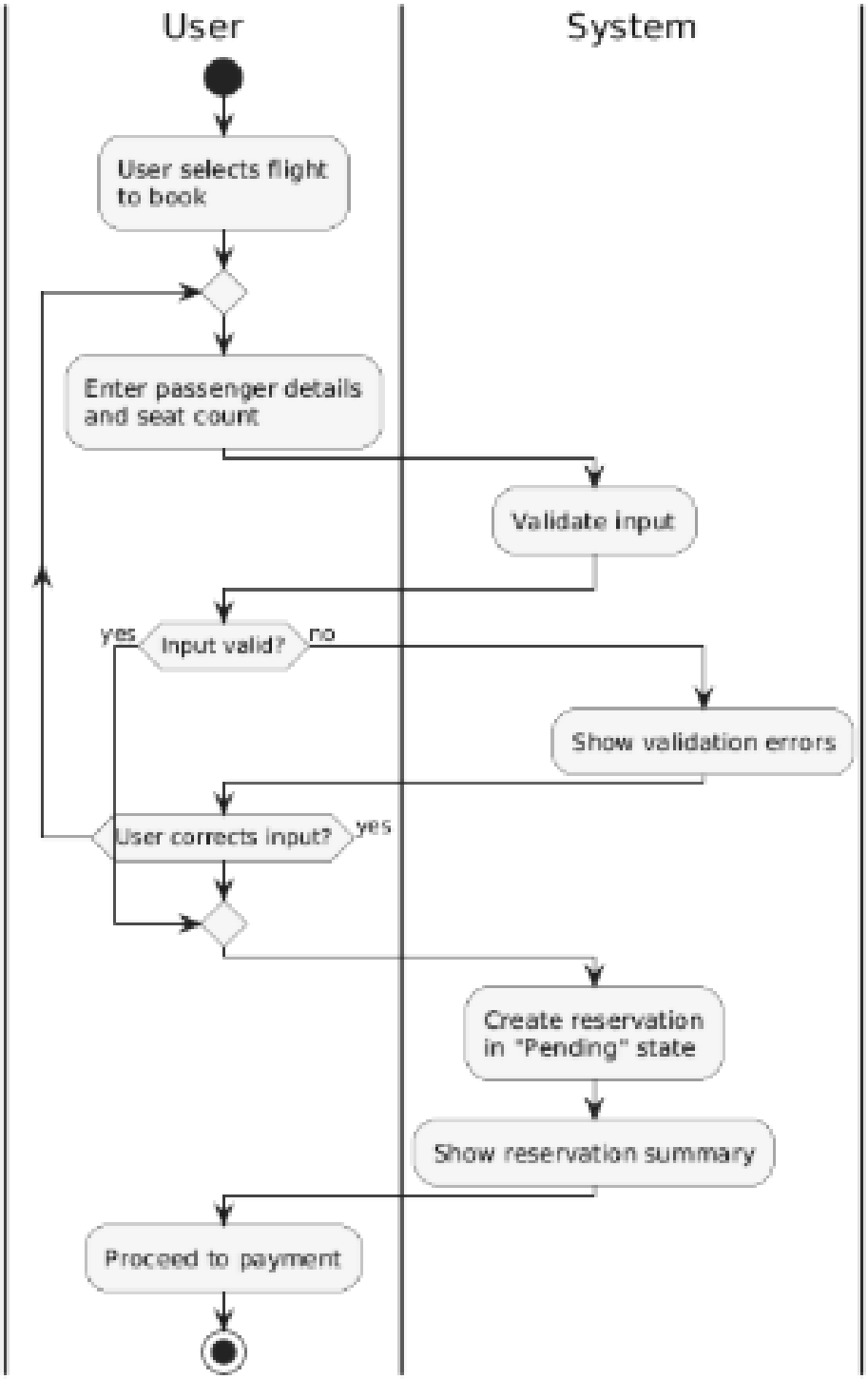
> The user enters payment information into the Payment UI. The System uses a Payment Strategy design pattern, to submit payments as either through Credit Card or PayPal, and simply logs the payment to the terminal; showing a successful status on the UI. If payment is declined or invalid, the system displays an error message and allows the user to retry. If payment is approved, the system records the payment in the Database, updates the reservation to "Confirmed," and generates a confirmation/receipt that is displayed to the user in the GUI.

Activity Diagrams

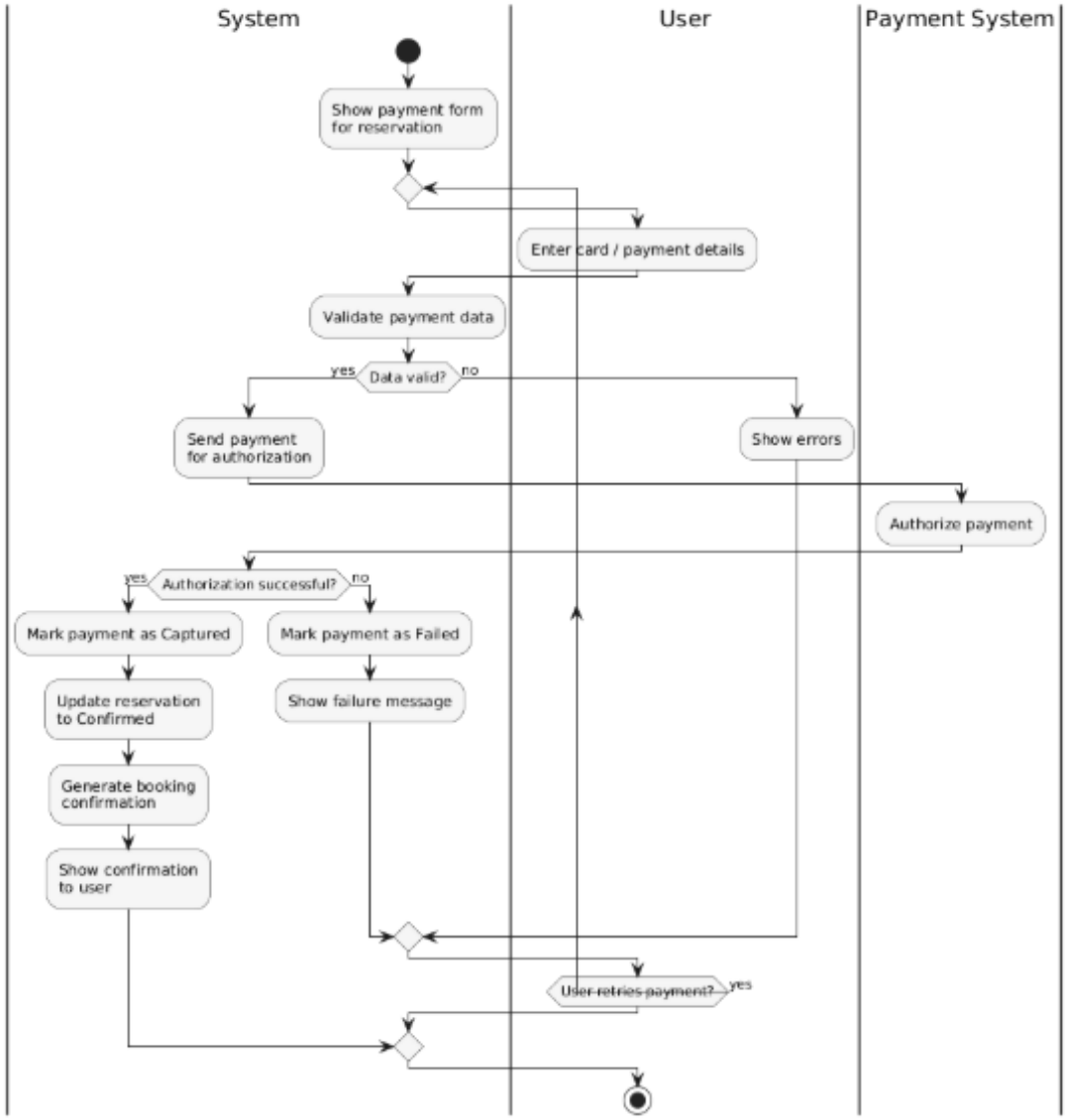
Login Activity Diagram



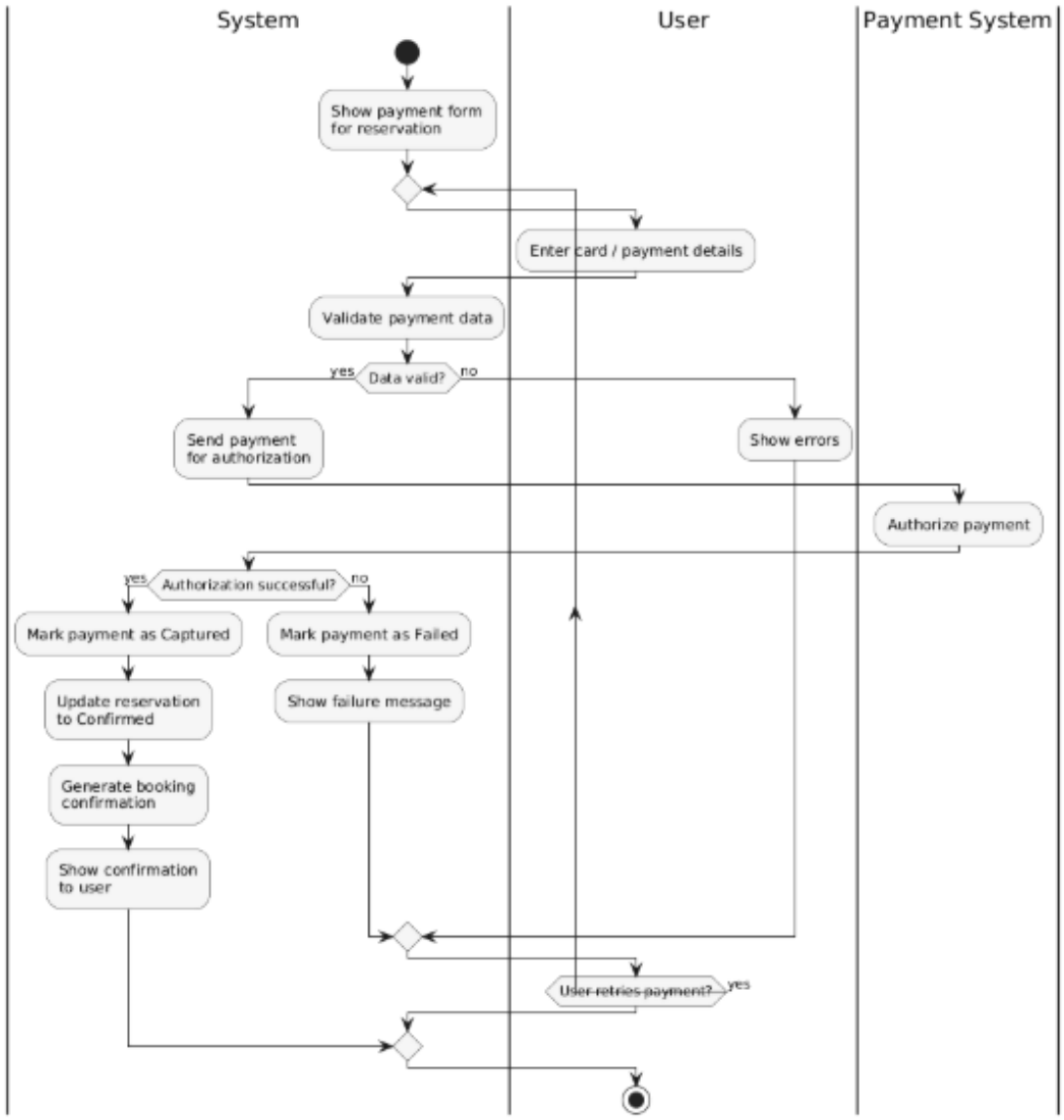
Browsing Flights Activity Diagram



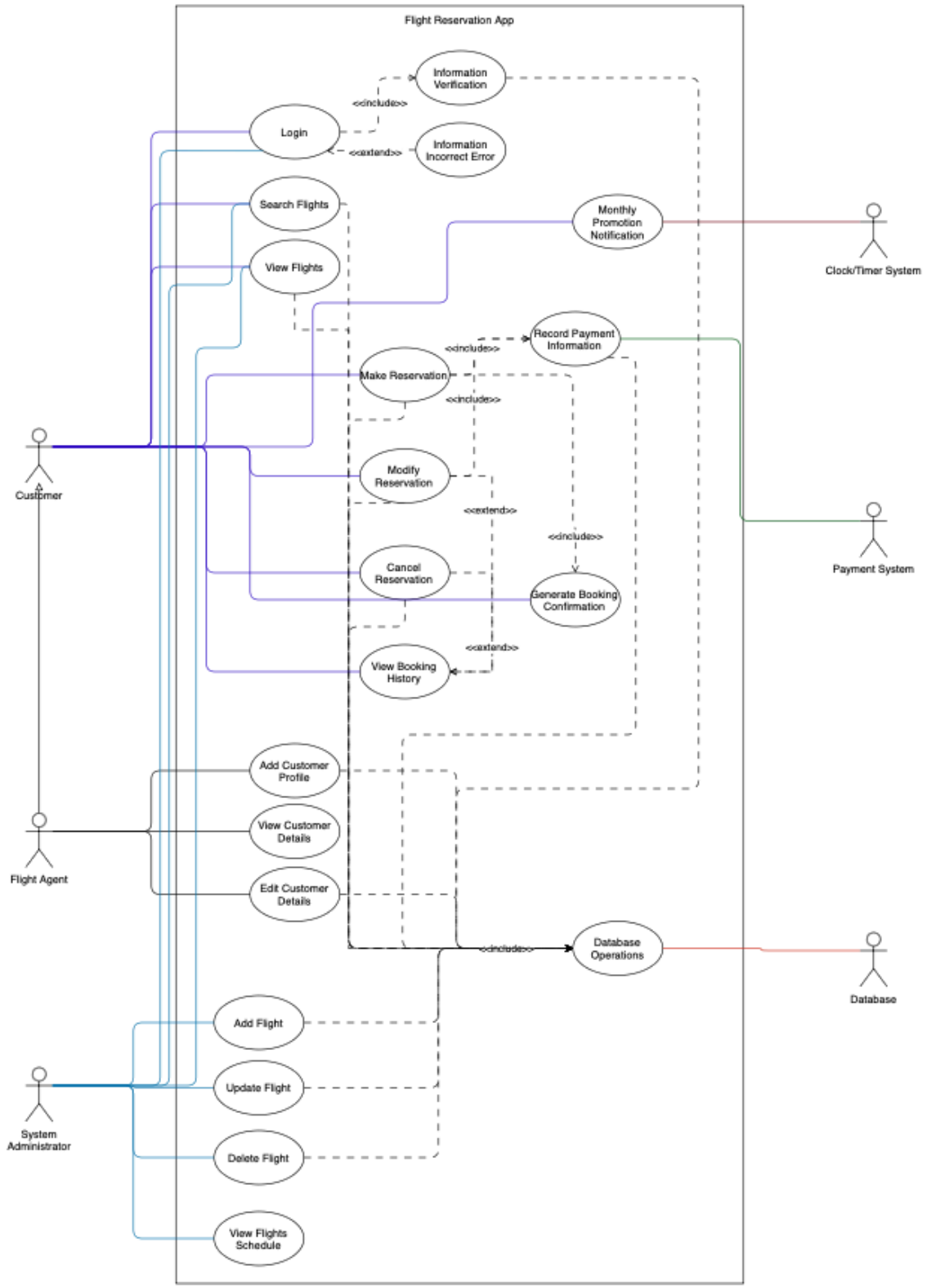
Booking Flight Activity Diagram:



Making Payment Activity Diagram:



UML Use-Case Diagram



Use Case System's Scenarios

Use-Case Scenario: Login

The User opens the LoginUI and enters login credentials including username, password, and role. The LoginUI calls login() through CustomerService. The CustomerService calls CustomerDAO to fetch the customer from the Database. If authentication fails, the LoginUI displays an error message and the User re-enters credentials. If authentication succeeds, the System loads the role-specific main menu in the MainWindowUI.

Use-Case Scenario: Search Flights

The User opens the SearchFlightsUI and enters search criteria such as origin, destination, and dates. The SearchFlightsUI calls searchFlights() on the FlightService. The FlightService calls FlightDAO, which queries the Database for matching Flight records. The SearchFlightsUI displays the returned flight list. If the result is empty, the system displays "No flights found," and the user can refine the criteria and search again.

Use-Case Scenario: View Flight Details

After a successful search, the User selects a specific flight and clicks "View Details" in the SearchFlightsUI. The SearchFlightsUI calls searchFlight() on the FlightService. The FlightService calls FlightDAO, which retrieves the selected Flight from the Database. The SearchFlightsUI displays the returned flight details to the user. The user may proceed to booking or return to the search screen.

Use-Case Scenario: Make Reservation (Book Flight)

The User selects a flight and chooses to reserve/book using the BookingUI. The BookingUI collects passenger information and seat count, then calls makeReservation() on the BookingService. The BookingService sets the reservation as pending, and initiates the payment procedure. If the payment fails, an error is brought up and brings the user back to the BookingUI. If payment is successfully made, the Reservation is updated to Confirmed. Finally, the BookingUI shows the reservation summary and prompts the user to proceed to payment.

Use-Case Scenario: Make Payment

The User opens the PaymentUI for a selected reservation and enters payment details. The PaymentUI calls pay() on the PaymentSystem. The PaymentSystem uses PaymentStrategy to accept multiple types of payment. If the payment data is invalid, the PaymentUI displays errors and the user re-enters payment information. If payment data is valid, the PaymentSystem returns the validity of the payment, such that the reservation is confirmed.

Use-Case Scenario: View Booking History

The User selects “View Booking History” from the MainWindowUI. The System calls getBookingHistory() (through the relevant controller/service). The system queries the Database for all Reservation records associated with the user and displays them in the GUI. The user may select any reservation to view details, modify it, or cancel it where permitted.

Use-Case Scenario: Modify Reservation

The User selects a reservation from booking history and chooses “Modify Reservation.” The GUI collects the requested changes and calls modifyReservation() on the relevant controller/service. The system validates the change request using validateModification() and updates the reservation using updateReservation() in the Database. The GUI then displays the updated reservation details.

Use-Case Scenario: Cancel Reservation

The User selects a reservation and chooses “Cancel Reservation.” The GUI prompts the user to confirm cancellation. The system calls cancelReservation() and updates the reservation status in the Database. The GUI displays a cancellation confirmation message.

Use-Case Scenario: Receive Monthly Promotion News

The external ClockTimerSystem triggers the monthly promotion event. The TimerSystem calls notifyObservers() (or equivalent) to notify the NotificationsService. The NotificationsService performs createNotification() to store a promotion notification in the Database, and then signals the GUI to display the new promotion notification to users (for example, through a notification indicator or popup).

Use-Case Scenario: Manage Flights (System Administrator)

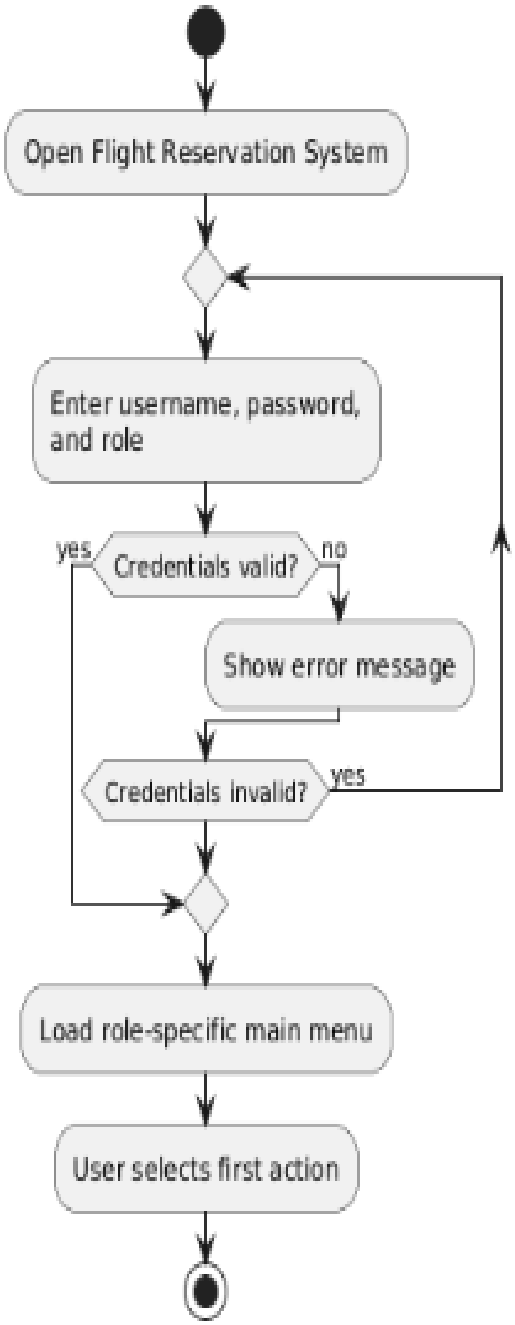
The SystemAdministrator opens the flight management view in the GUI. To add a flight, the GUI calls addFlight() on the FlightService, which persists the new Flight in the Database. To update a flight, the GUI calls updateFlight() and the system writes changes to the database. To delete a flight, the GUI calls deleteFlight() and the system removes or disables the flight record in the database. The administrator may also view a schedule by calling getFlightSchedule(), which queries the database and displays a list of flights in the GUI.

Use-Case Scenario: Manage Customer Profiles (Flight Agent / System Admin)

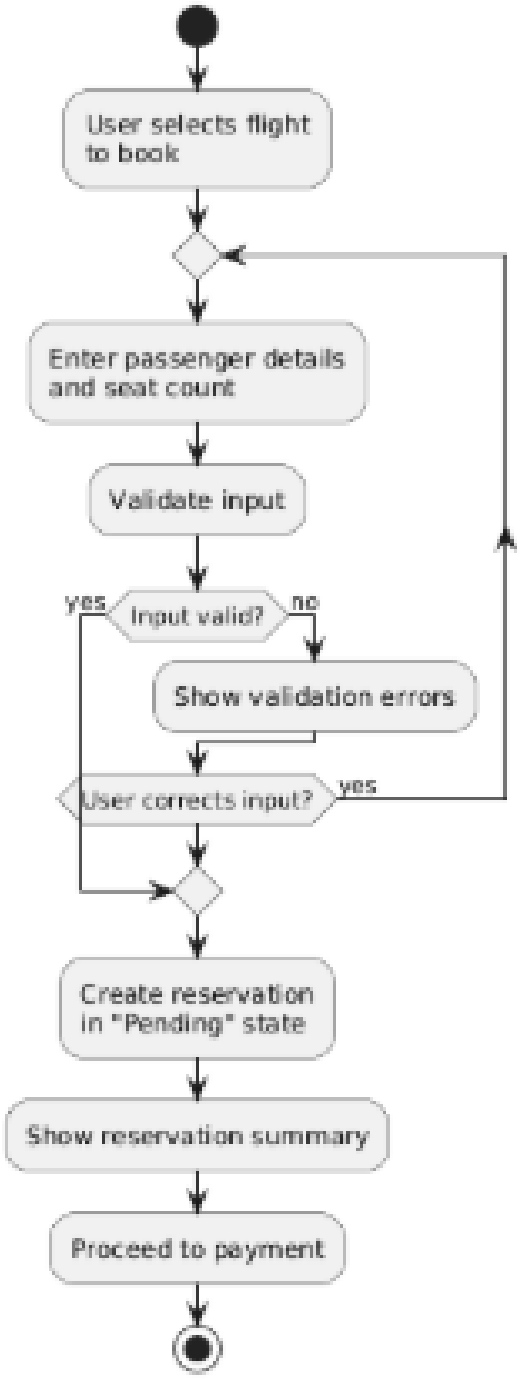
A FlightAgent (and possibly SystemAdministrator) selects a customer management option in the GUI. To create a customer profile, the GUI calls addCustomerProfile() and stores the new Customer in the Database. To view customer details, the GUI calls getCustomerDetails() and displays the returned information. To edit customer details, the GUI calls updateCustomerDetails() and persists changes in the database.

State Transition Diagrams

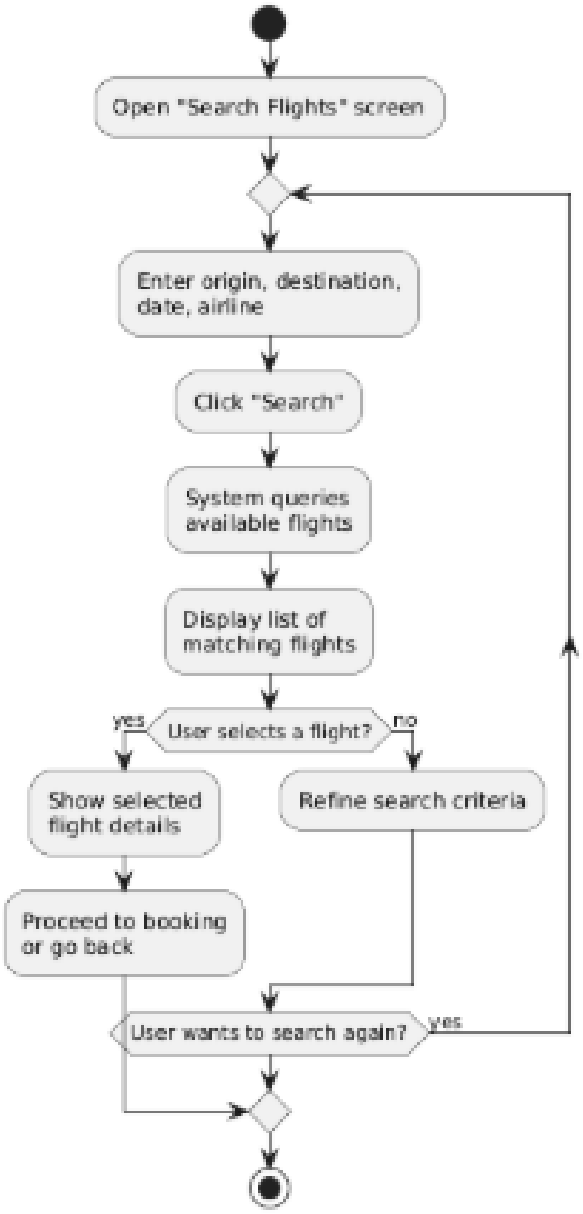
Entire System



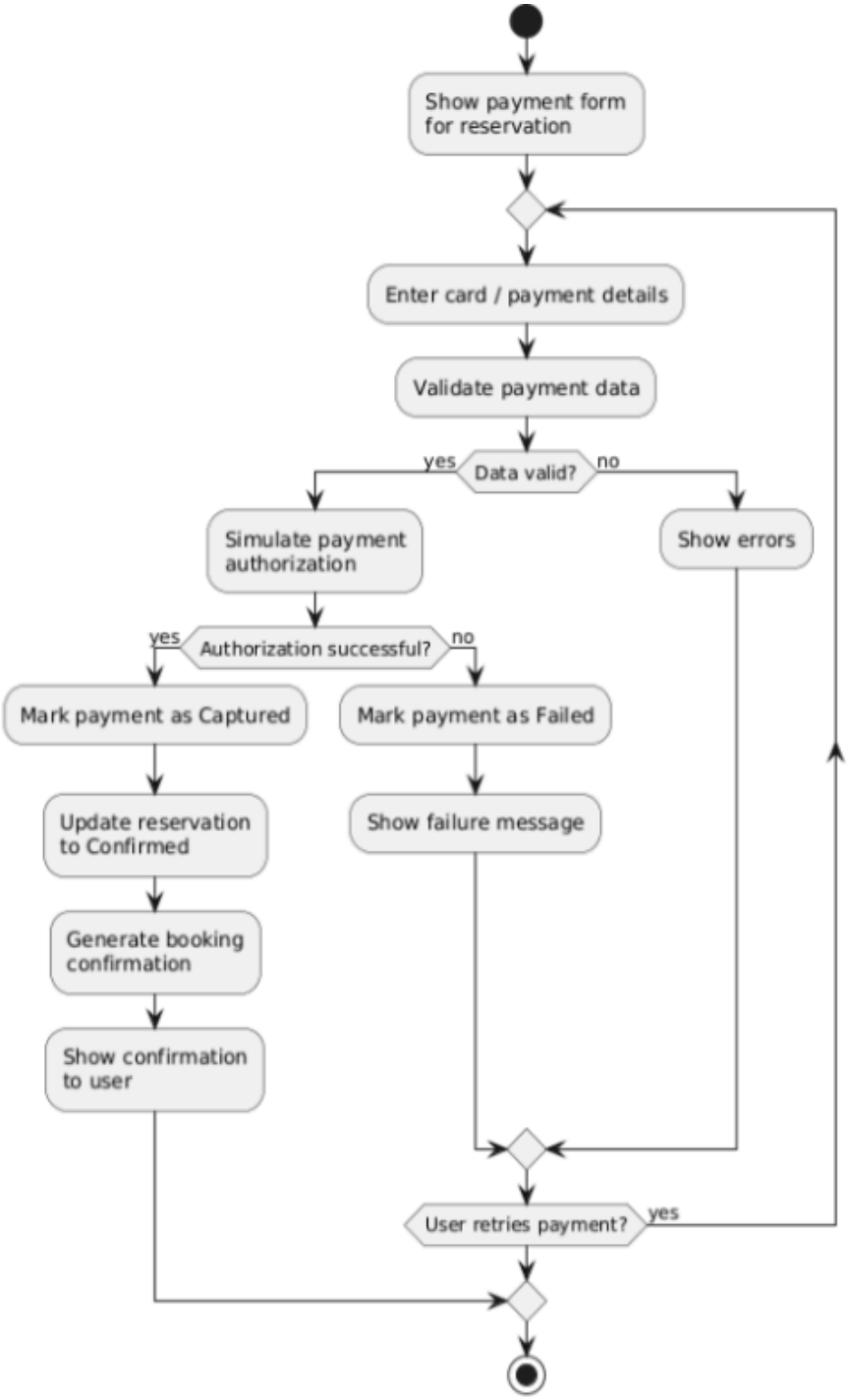
Reservation System



Flight System

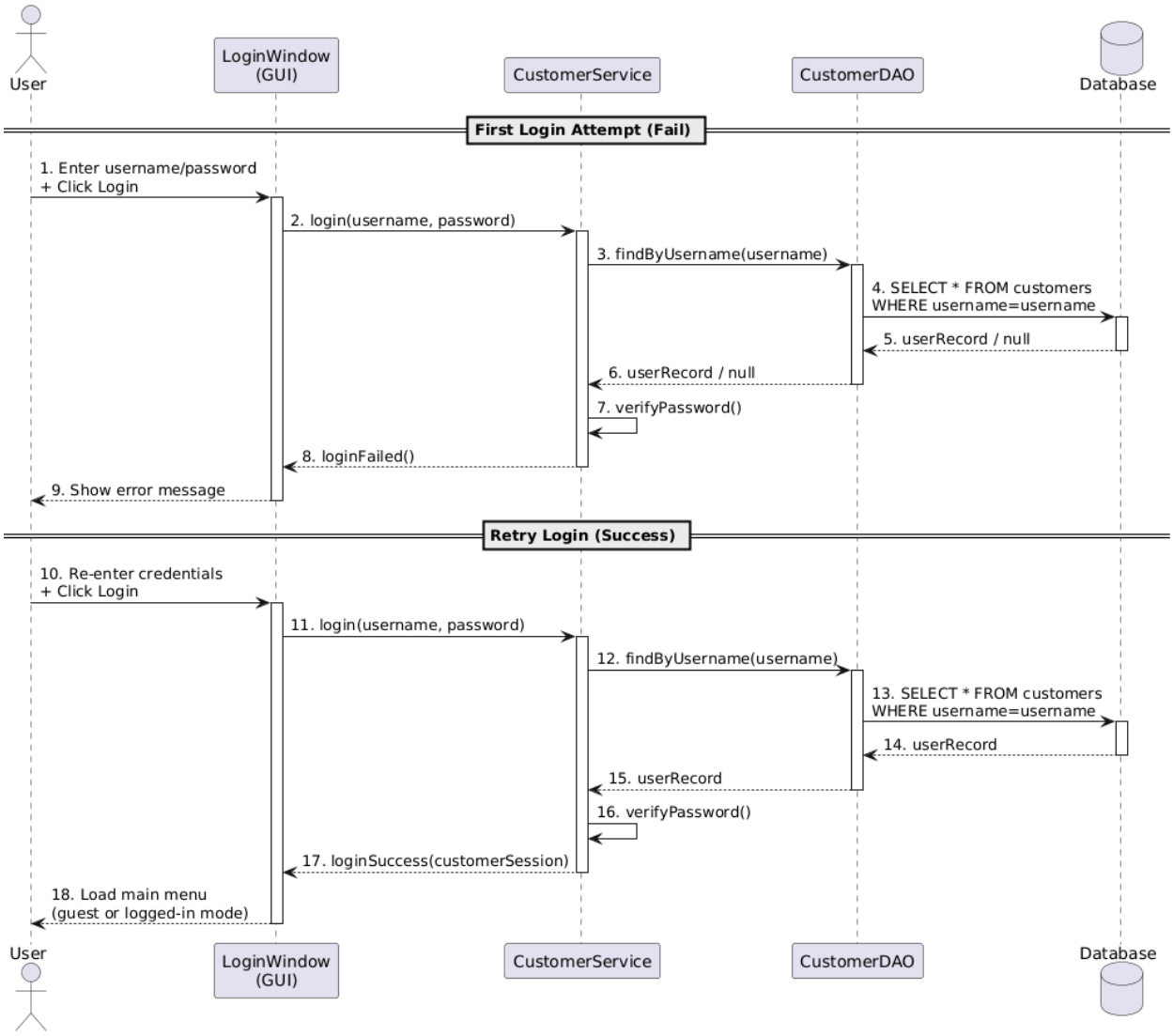


Payment System

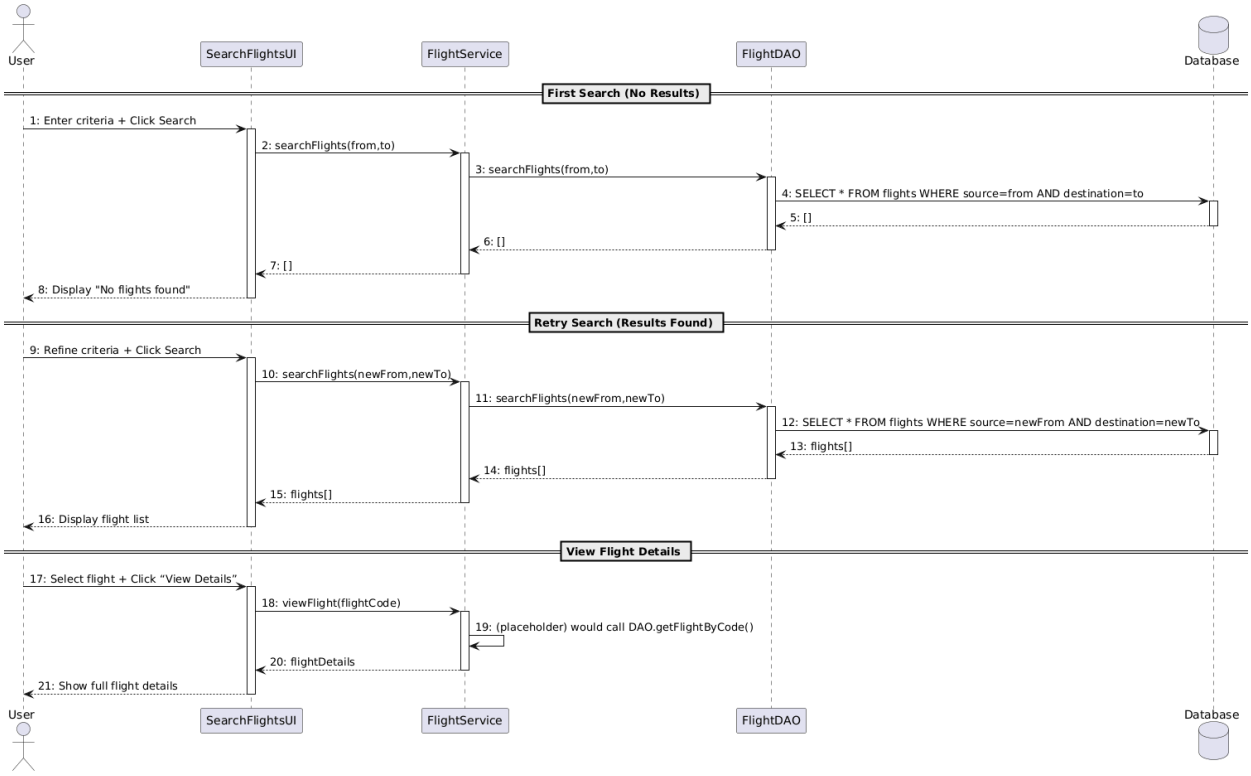


Interaction Diagrams

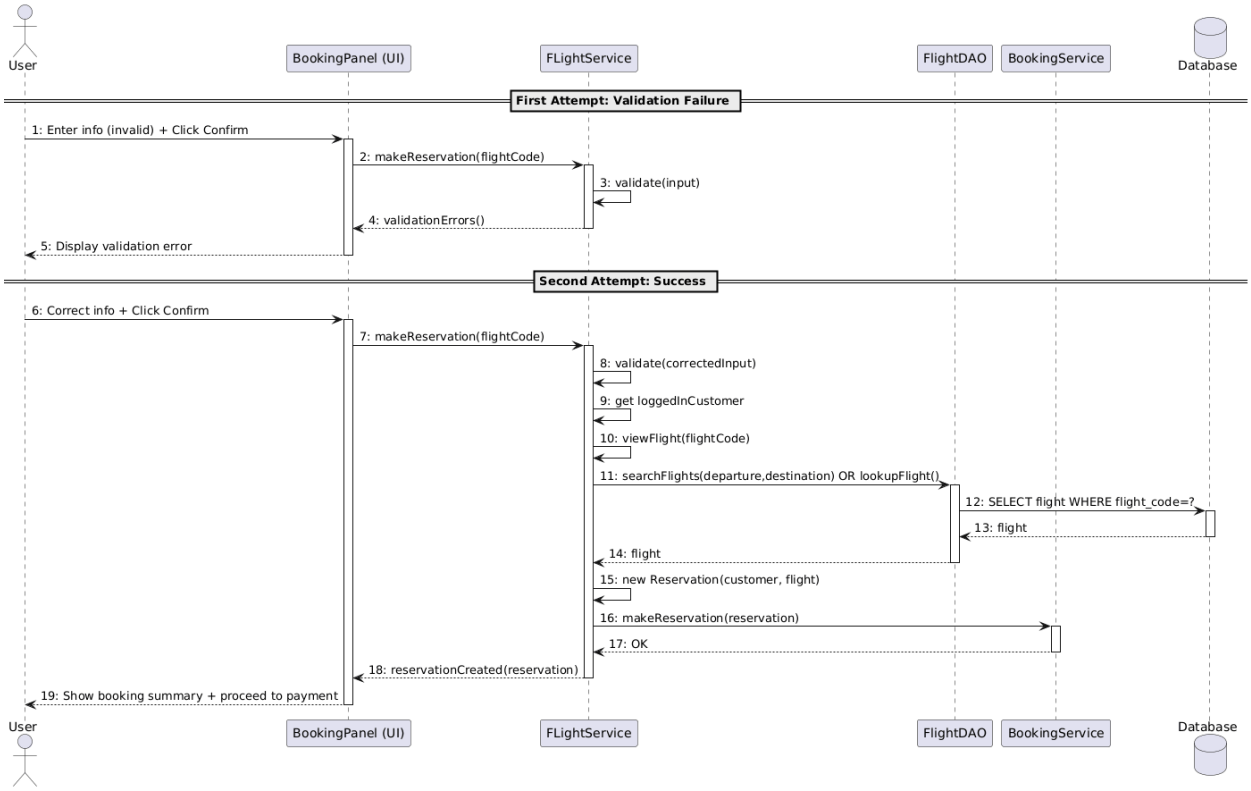
Login (with failure + retry) interaction diagram:



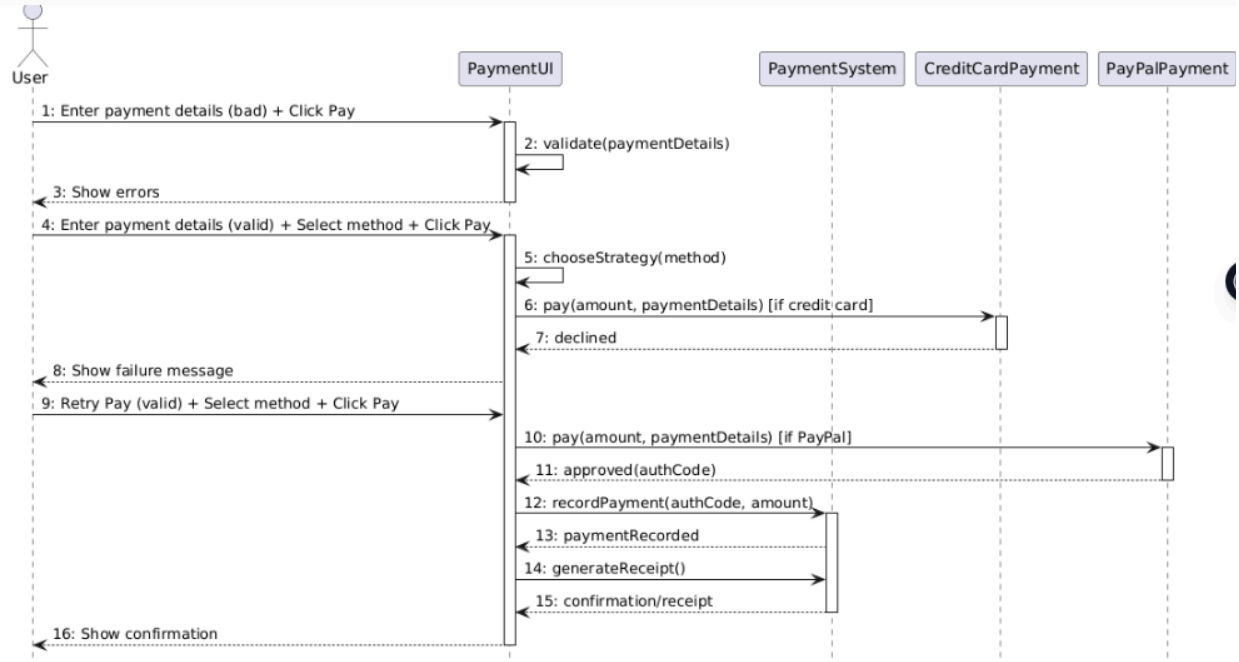
Search Flights + View Details (with “no results” + retry) interaction diagram:



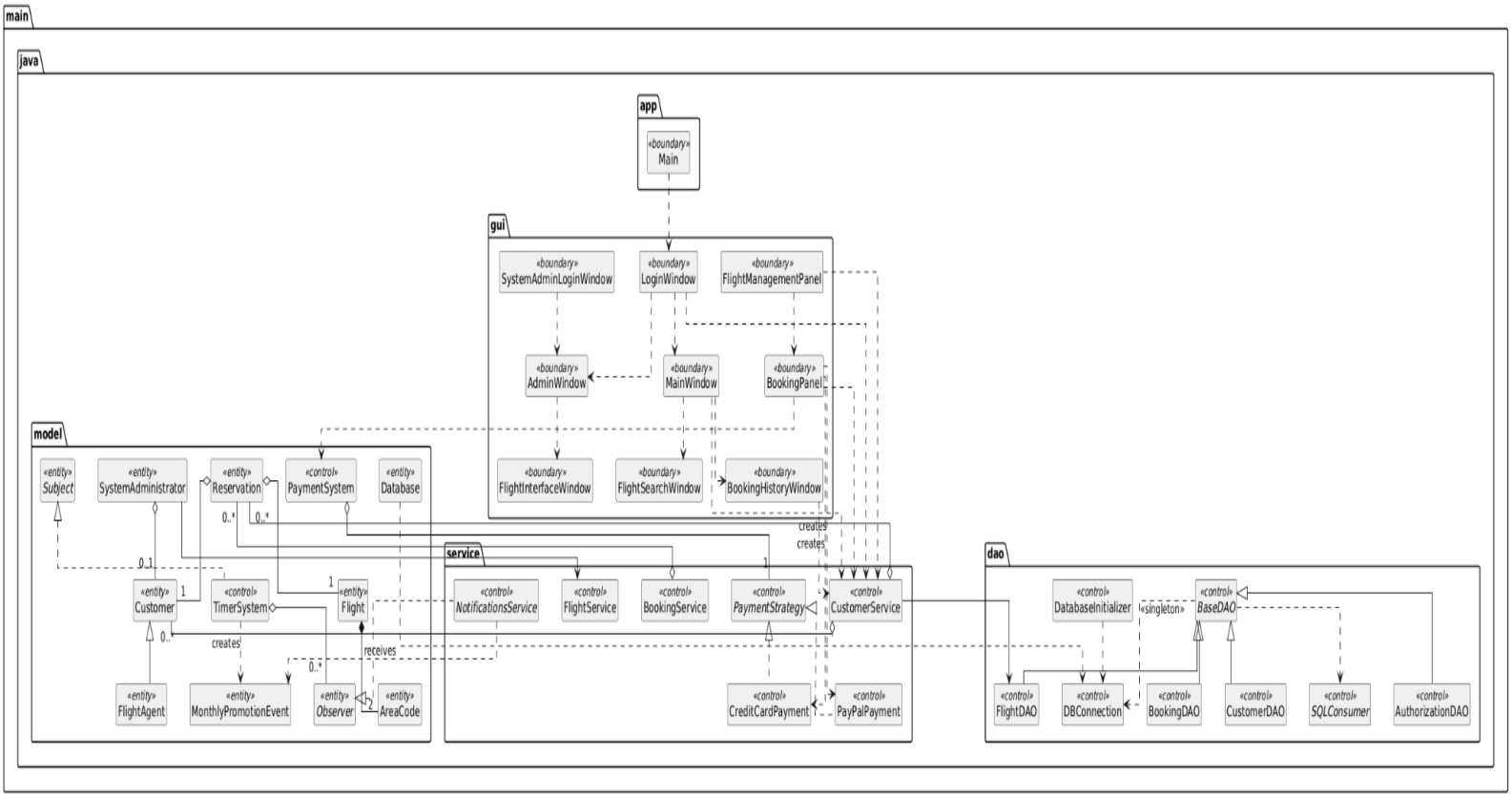
Make Reservation (with validation failure + retry) interaction diagram:



Payment interaction diagram:

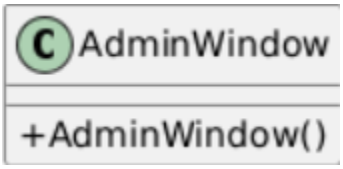


Class Diagram



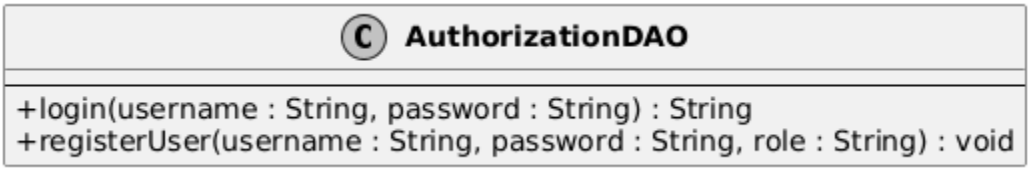
Detailed Class Diagrams

AdminWindow.java

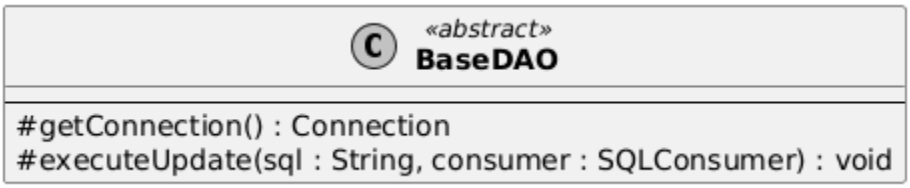


AreaCode.java:

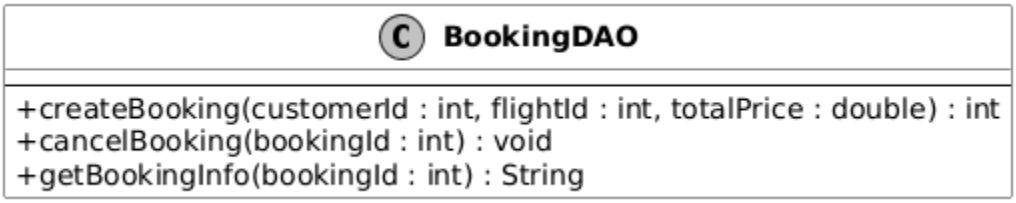
AuthorizationDAO.java:



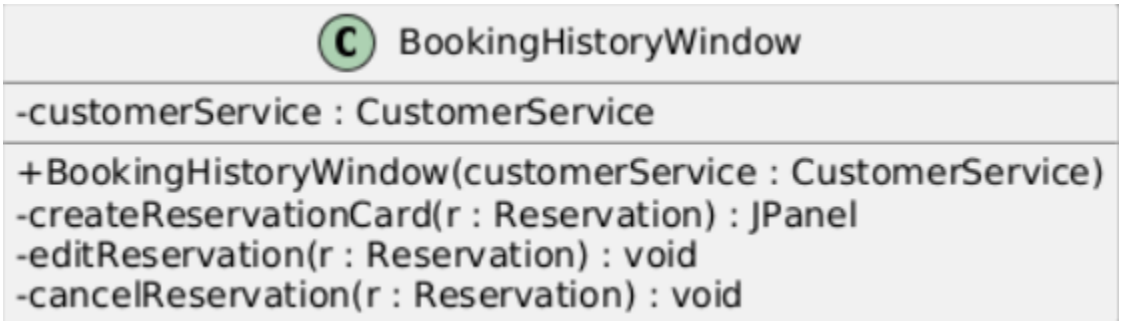
BaseDAO.java:



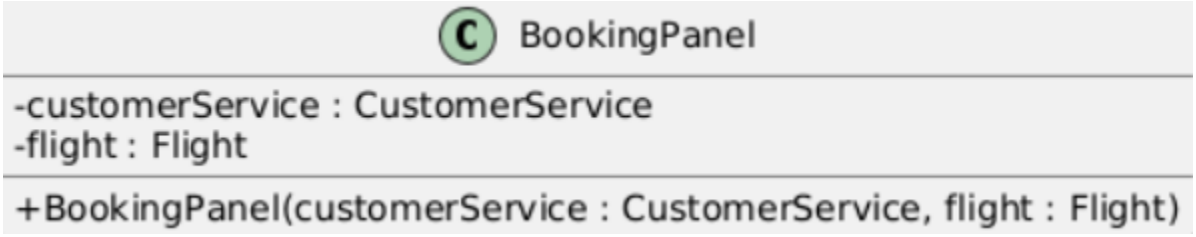
BookingDAO.java:



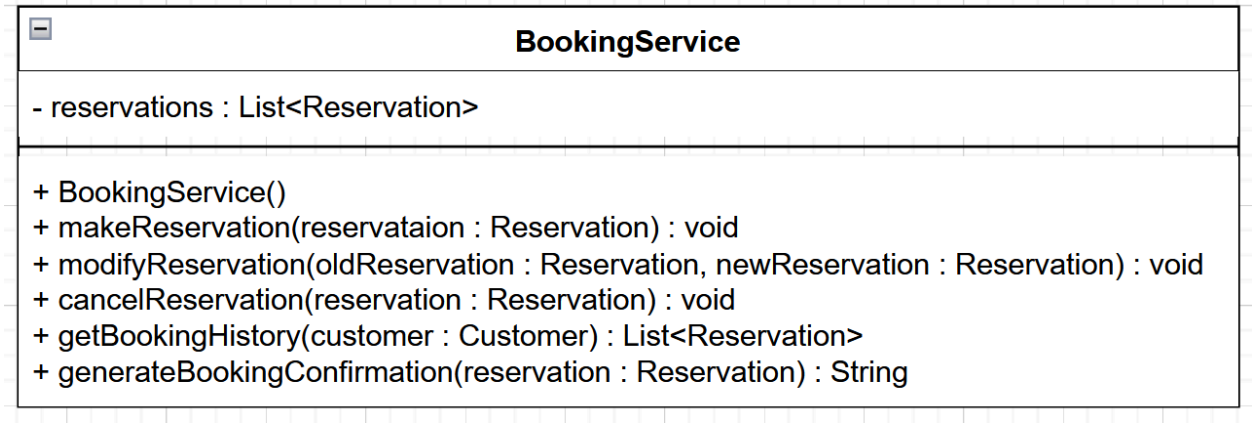
BookingHistoryWindow.java



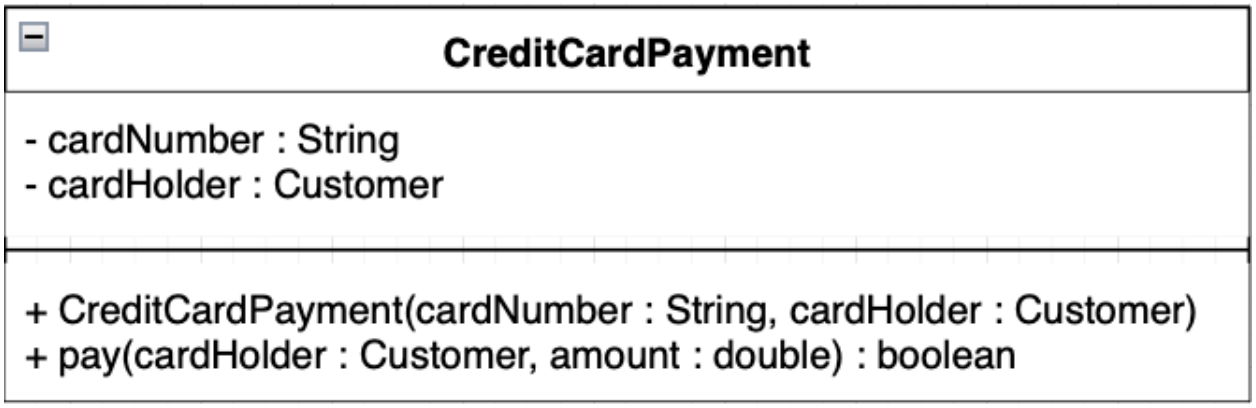
BookingPanel.java



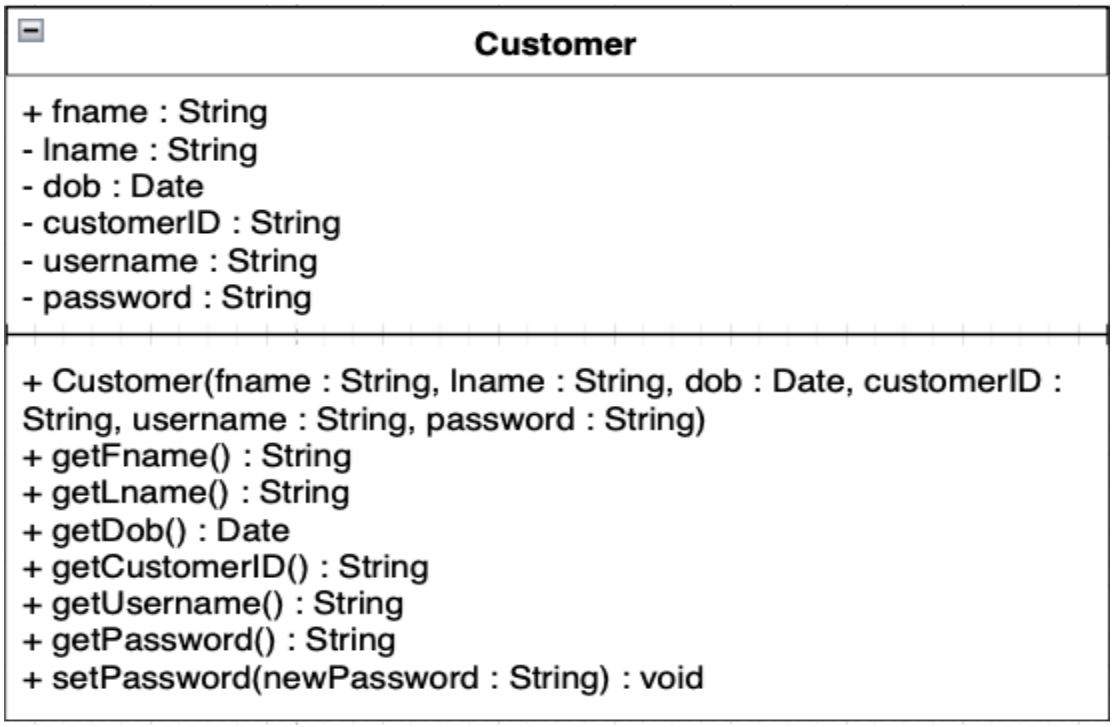
BookingService.java



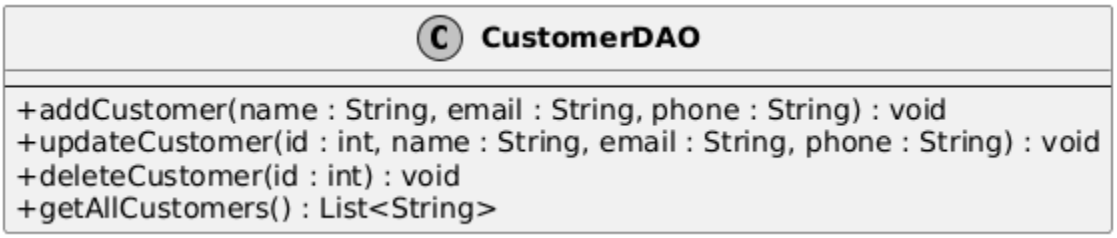
CreditCardPayment.java:



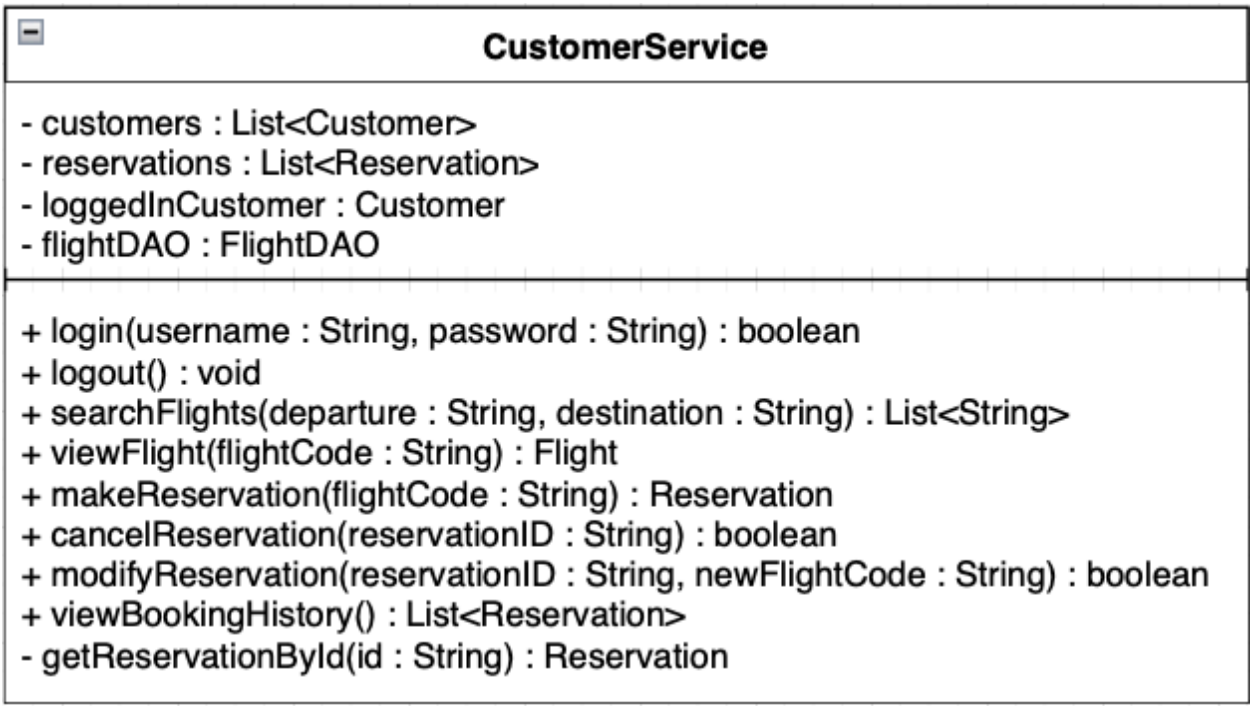
Customer.java:



CustomerDAO.java:

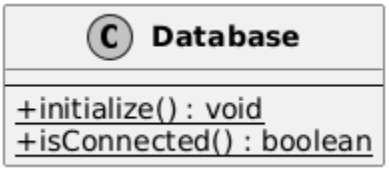


CustomerService.java:

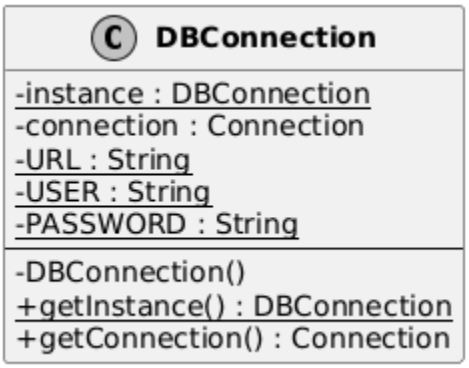


DatabaseInitializer.java:

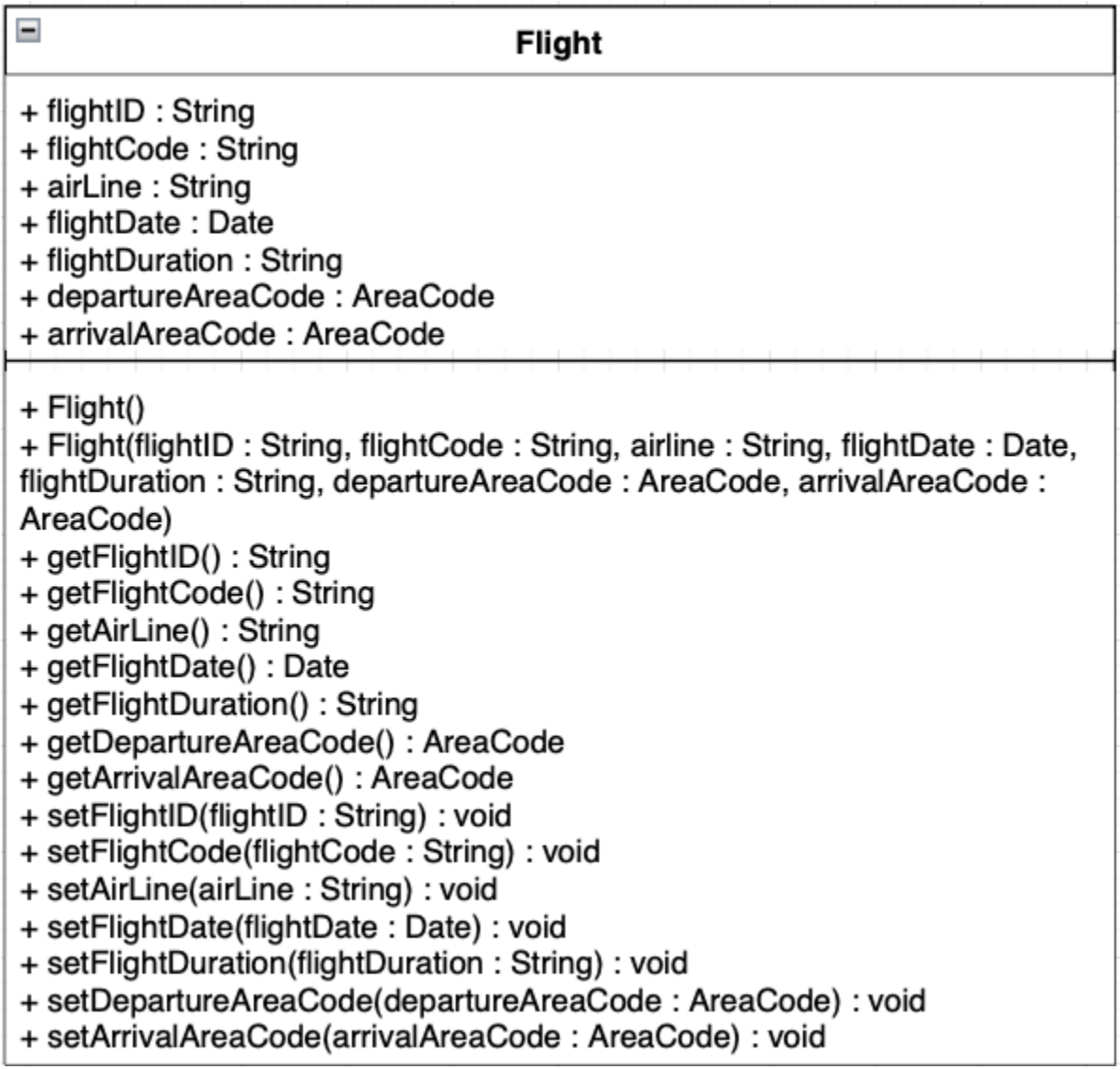
Database.java:



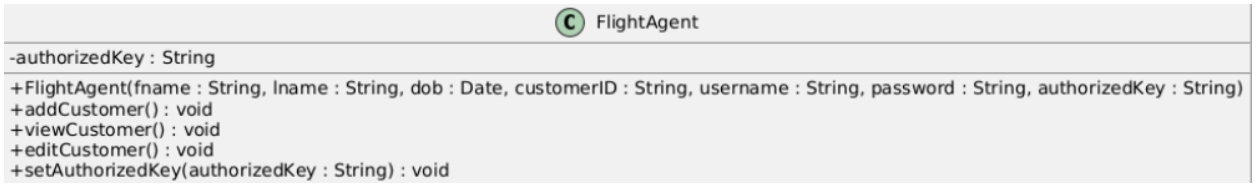
DBConnection.java:



Flight.java:



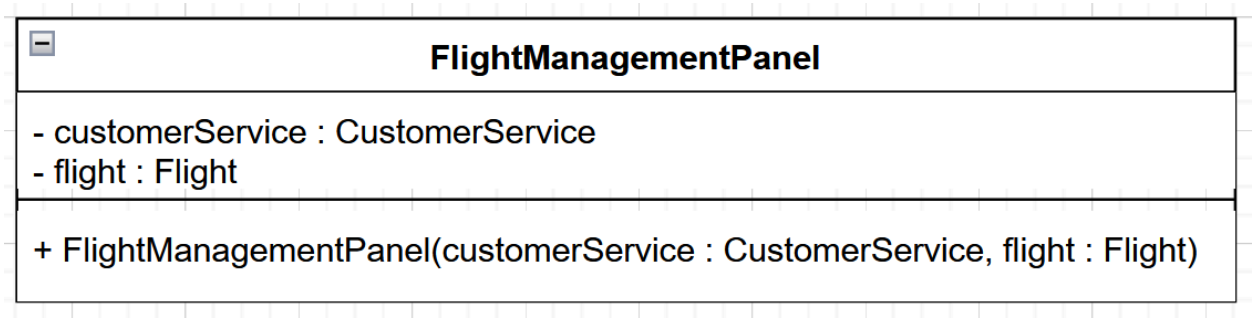
FlightAgent.java:



FlightDAO.java:

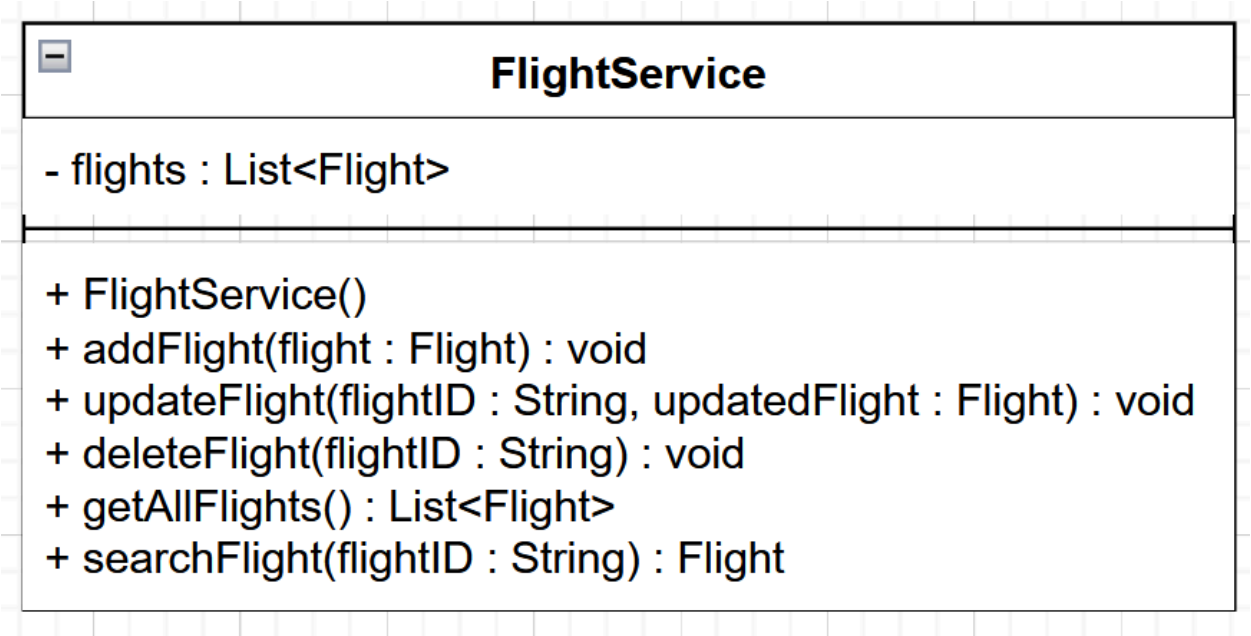
FlightinterfaceWindow.java

FlightManagementPanel.java

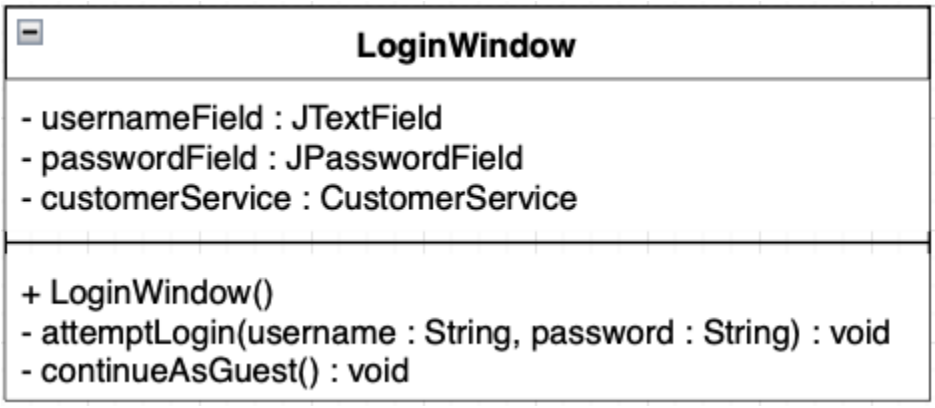


FlightSearchWindow.java

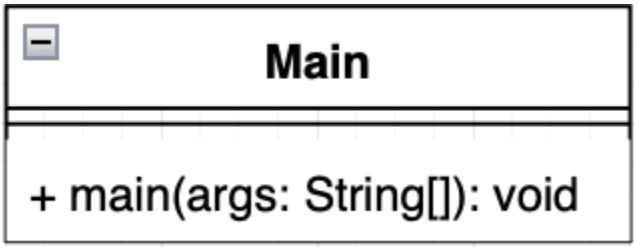
FlightService.java



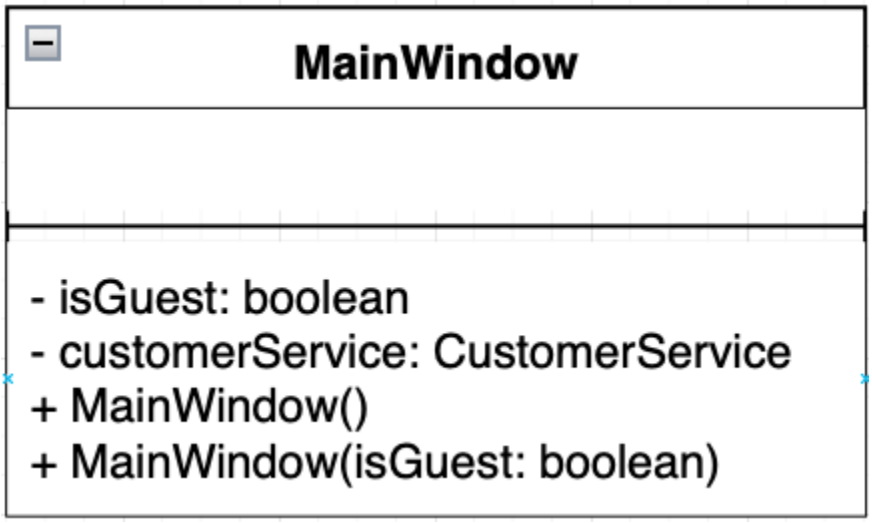
LoginWindow.java:



Main.java:



MainWindow.java:

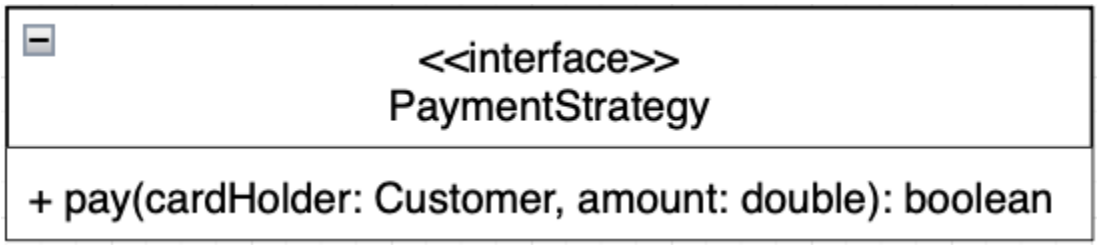


MonthlyPromotionEvent.java

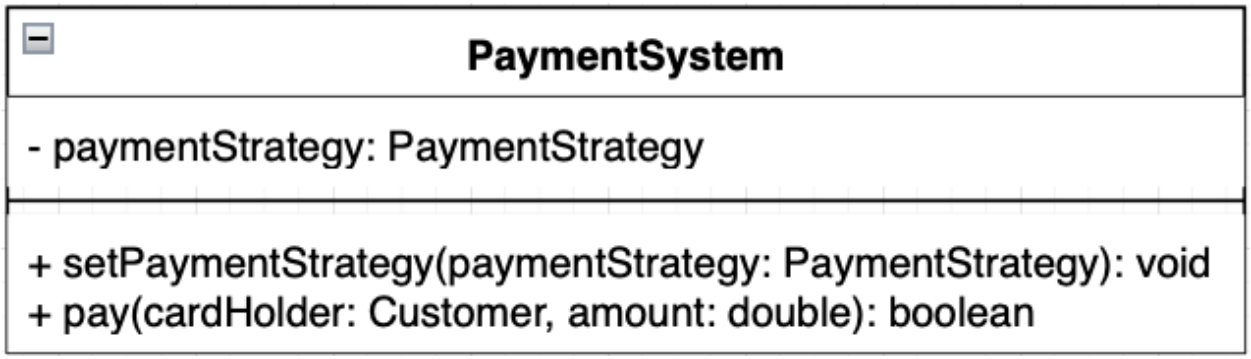
NotificationsService.java

Observer.java

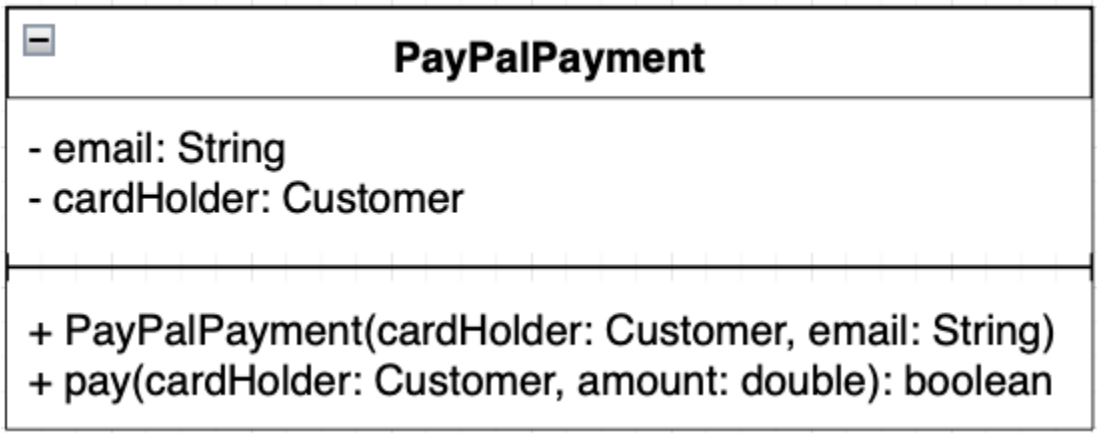
PaymentStrategy.java:



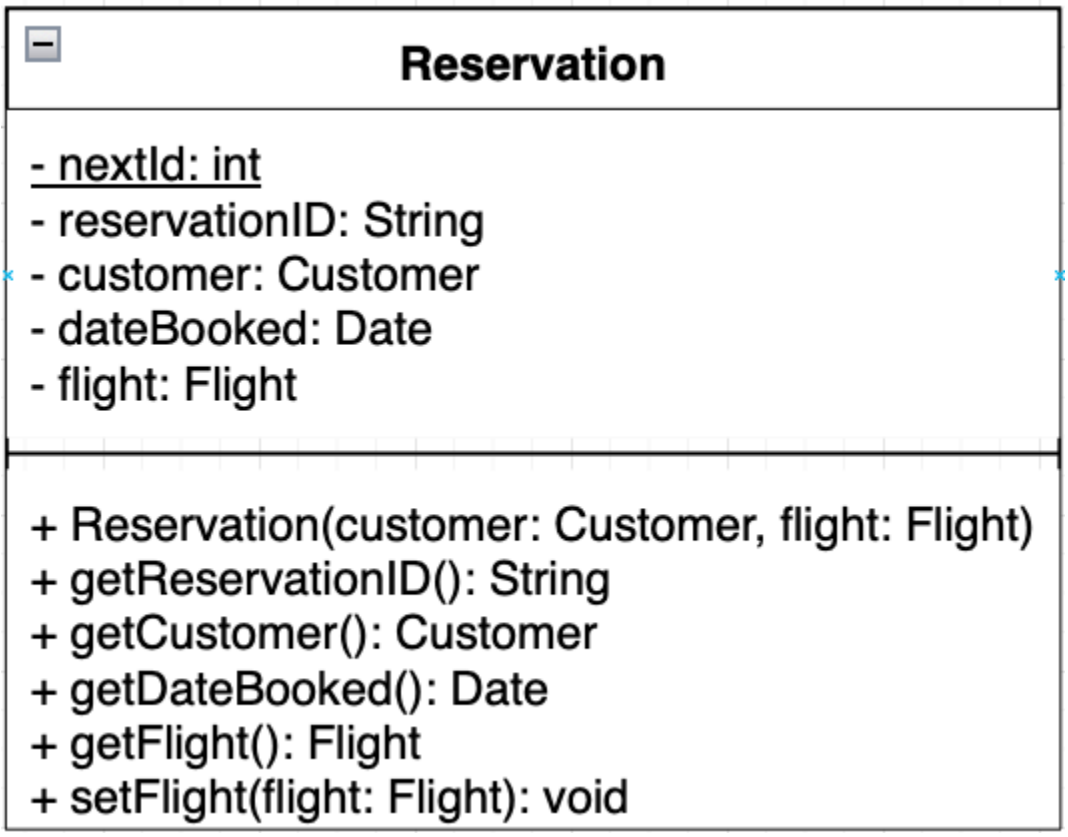
PaymentSystem.java:



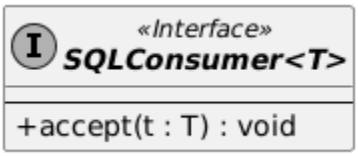
PayPalPayment.java:



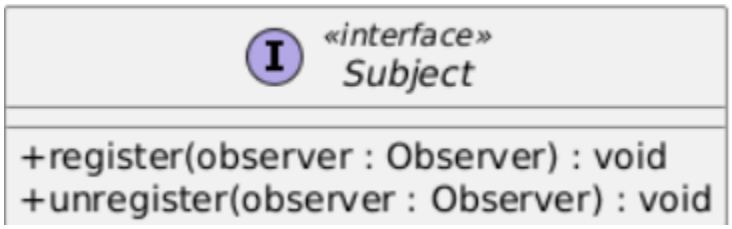
Reservation.java:



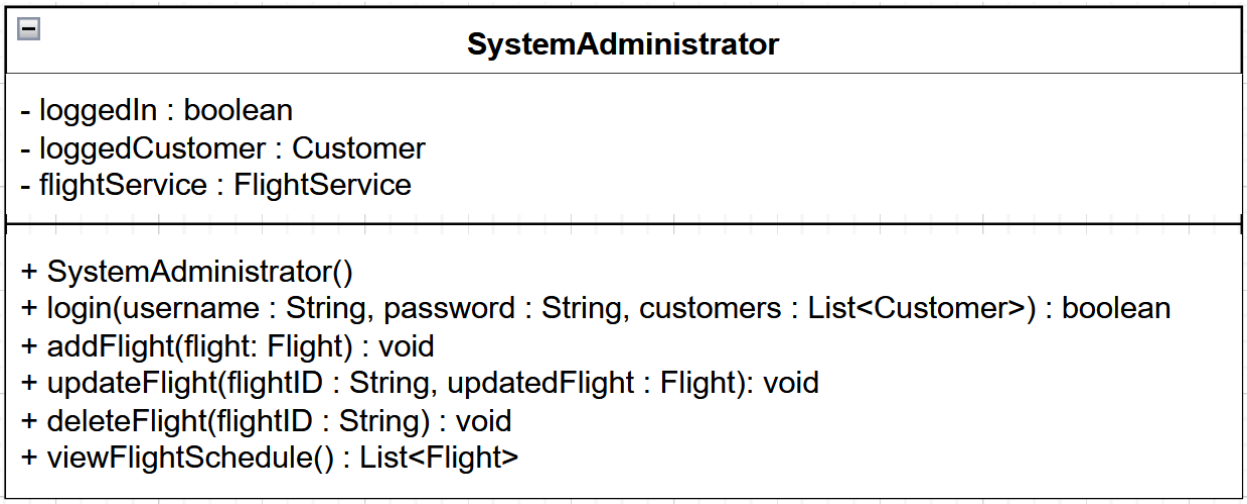
SQLConsumer.java:



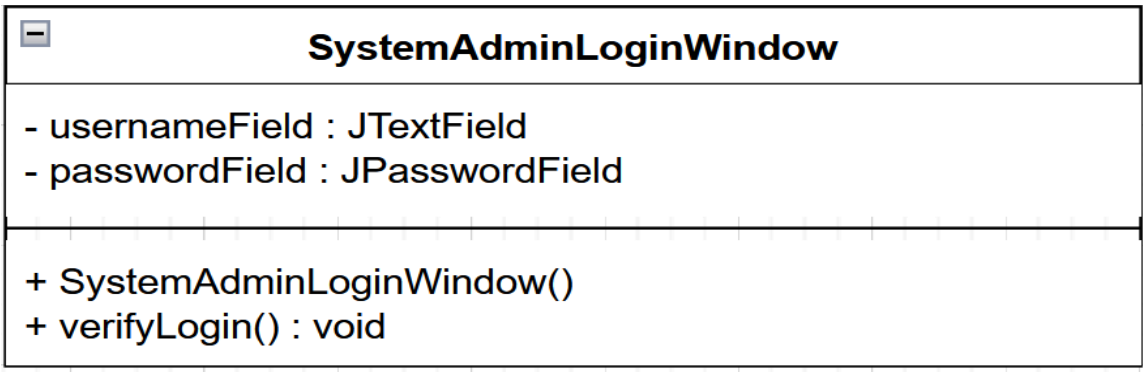
Subject.java



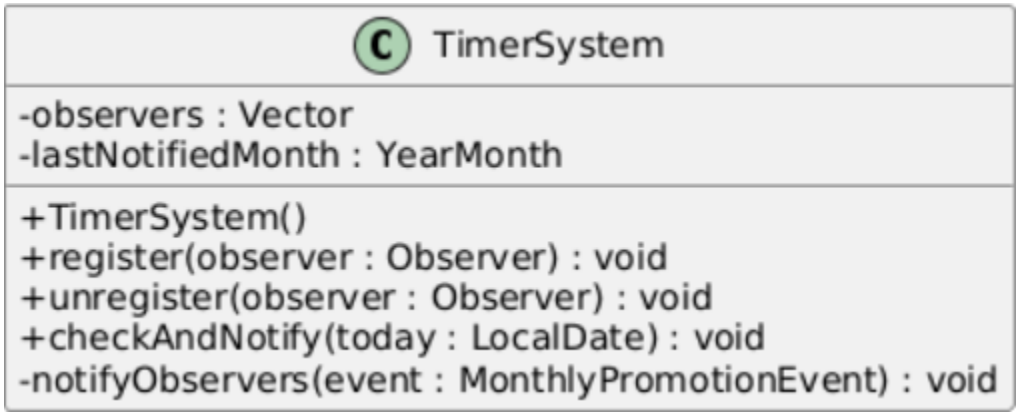
SystemAdministrator.java:



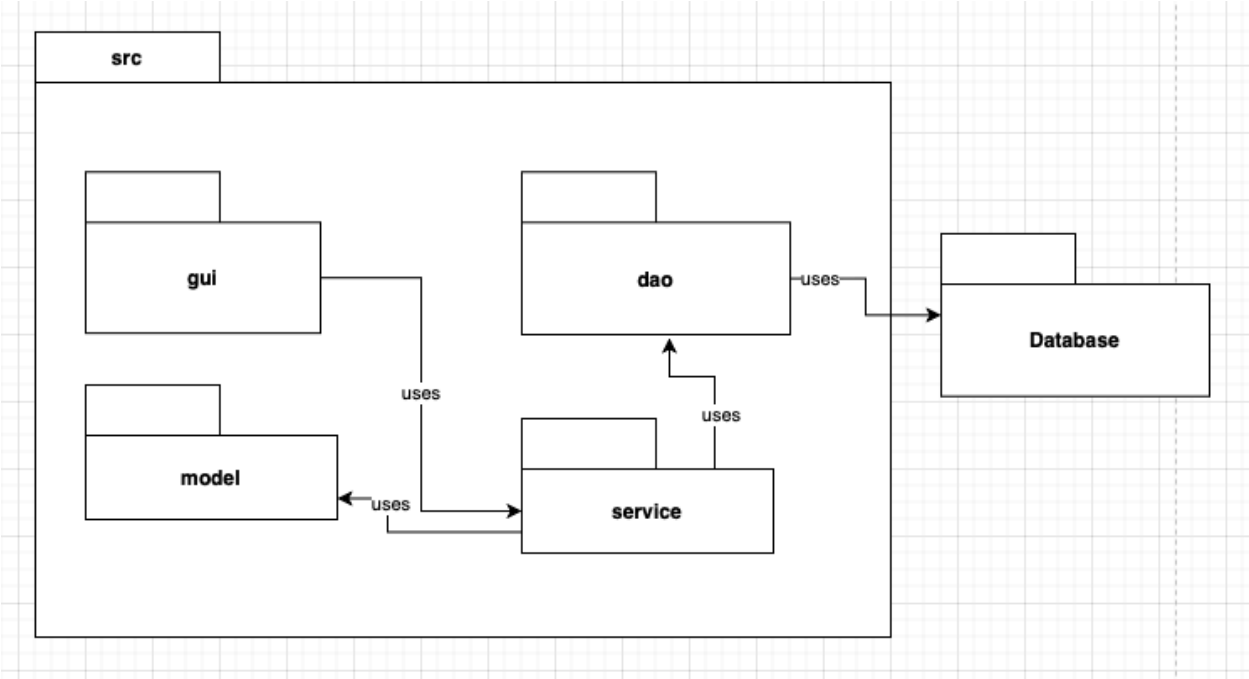
SystemAdminLoginWindow.java:



TimerSystem.java



Package Diagram



Package	Classes
app	• AppConfig.Java • Main.java
dao	• AuthorizationDAO.java • BaseDAO.java • BookingDAO.java • CustomerDAO.java • DatabaseInitializer.java • DBConnection.java • FlightDAO.java • SQLConsumer.java
gui	• AdminWindow.java • BookingHistoryWindow.java • BookingPanel.java • FlightInterfaceWindow.java • FlightManagementPanel.java • FlightSearchWindow.java • LoginWindow.java • MainWindow.java

	• SystemAdminLoginWindow.java
model	• AreaCode.java • Customer.java • Database.java • Flight.java • FlightAgent.java • MonthlyPromotionEvent.java • Observer.java • PaymentSystem.java • Reservation.java • Subject.java • SystemAdministrator.java • TimerSystem.java
service	• BookingService.java • CreditCardPayment.java • CustomerService.java • FlightService.java • NotificationsService.java • PaymentStrategy.java • PayPalPayment.java