ENSF 480 Fall 2023 - Term Project Design Document

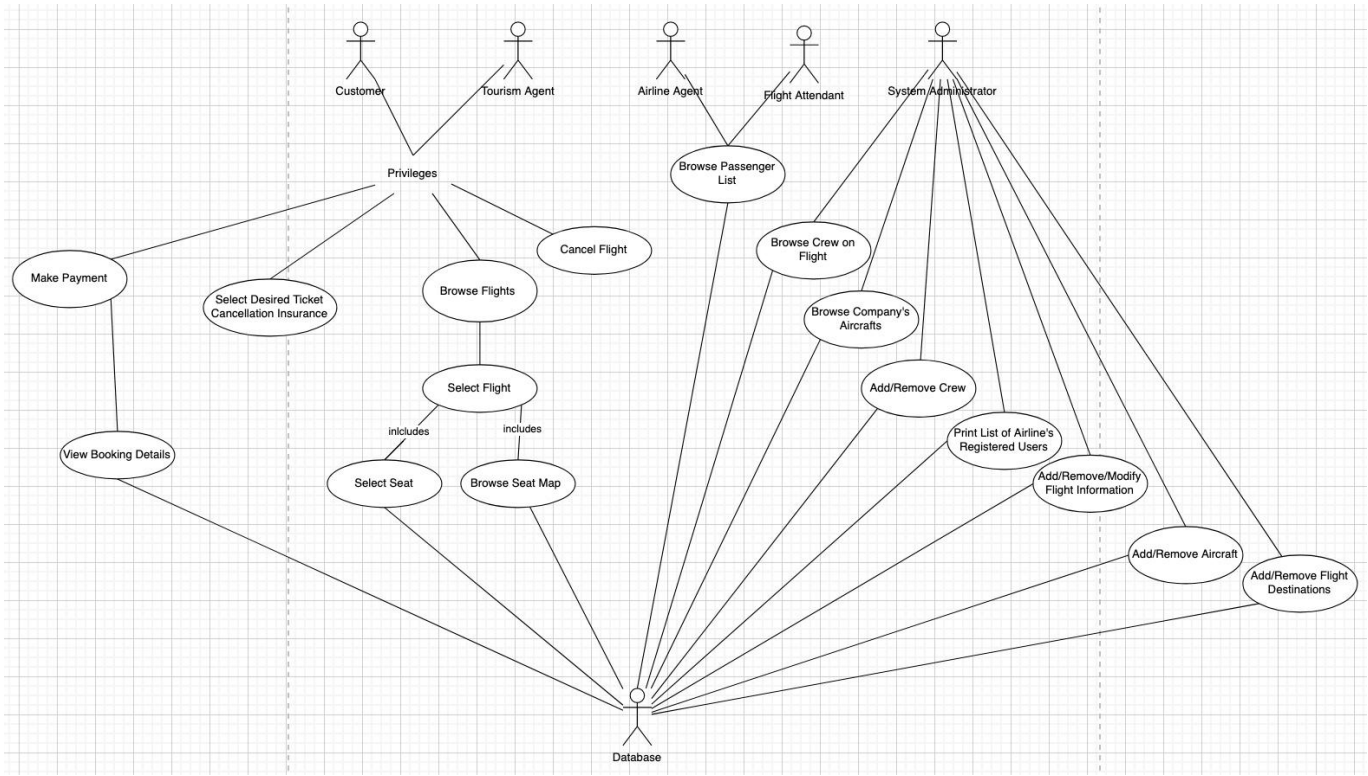
Group 6 - Caue Rodrigues, Jacob Lever, Rhishik Roy, Sahiti Akella

PART A:

System Description:

The Flight Reservation Web Application is an interactive system that caters to various types of users. The application is built in Java in order to implement core functionality of the system and is connected to a MySQL database to store and modify system information. The user interface will provide the user with options to use the application as a guest user, or create a new user. If a customer decided to create a new user, they will be considered a registered user and will be able to create a username and password which they can use to sign into the system. Other types of users can also create new accounts. Once a registered customer logs in, they are asked whether they would like to sign up for a company credit card. Customers are able to browse through available flights, select flight, view the seat map for the selected flight, select a seat and opt in for ticket cancellation insurance. Once the user selects their desired flight and seat, they are presented with a payment form that must validate their payment information in order to proceed to the next step. Once validated, the customer is able to view their trip information and receipt details. Tourism agents have the same functionality. However, they are able to select a customer that they would like to book a flight for. As for airline agents and flight attendants, they are able to browse through the passenger list of their selected flight. The system administrator is able to browse through flights, crews, aircrafts, and modify aircraft and flight information. They are also able to print a list of registered users.

System's Use Case Diagram:



System's Use Case Scenarios:

Browse Flight: Customers and tourism agents select the "Browse Flights" option. The system presents a list of available flights. The system waits for the user to select a flight.

Select Flight: Customers and tourism agents browse through available flights and selects a specific flight from the list. Once the user selects the desired flight, the system proceed to the graphical seat map.

Browse Seat Map: Customers and tourism agents are presented with the seat map for the selected flight. The system displays a graphical view of the seat map. The user browses through the available seats.

Select Seat: Customers and tourism agents select a desired seat (Ordinary, Comfort, and Business-Class) on the graphical map. The system then proceeds to the next step.

Select Desired Ticket Cancellation Insurance: Customers and tourism agents are given the option to select ticket cancellation insurance. User either selects 'Yes' or 'No.' If the user chooses to select cancellation insurance, the system adds the cancellation fee to the ticket price. If the user decides not to select cancellation insurance, the system does not update the ticket price. The system proceeds to the next step.

Make Payment: Customers and tourism agents are presented with a payment form, where they are asked to input their card details (Card holder name, card number, expiry date, and CVV). Once the user confirms payment, the system validates the credentials and present a success/error message accordingly.

View Booking Details: After successful payment, the system generates a receipt and ticket for the user. The user is able to view selected flight/seat/insurance information, as well as the total price of their trip.

Cancel Flight: User clicks the 'Cancel Flight' option on the system. The user browses through booked flights and selects the flight they would like to book. The system displays a pop up window to verify trip cancellation. The system cancels the flight and updates the database accordingly.

Browse Passenger List: Airline agents and flights attendants are able to select a specific flight and then view the information of passengers on that specific flight.

Browse Crew on Flight: System administrators will be able to select a specific flights and view the list of crew members that are working on the selected flight.

Browse Company's Aircrafts: System administrators will be able to view the list of aircrafts that are registered with the airline.

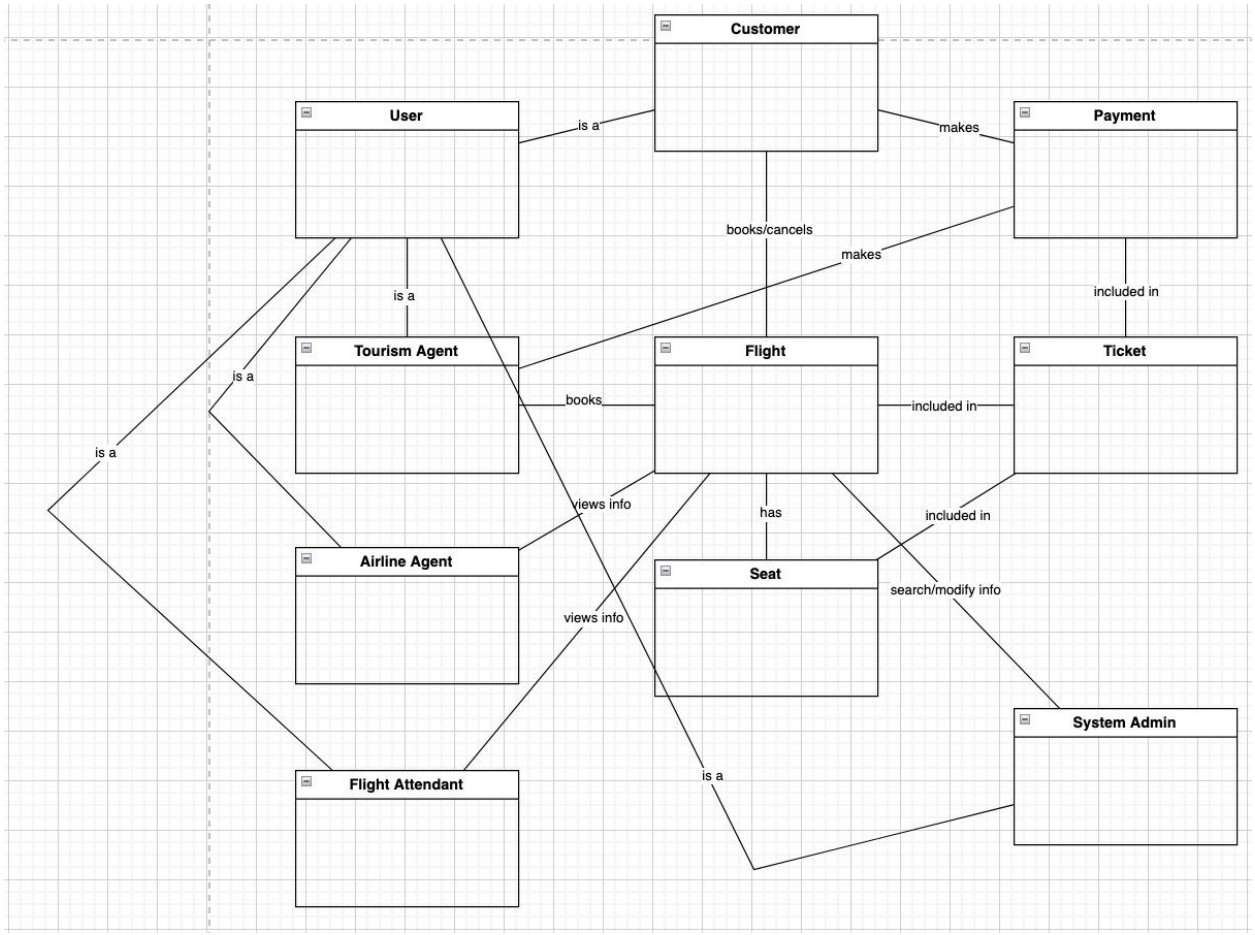
Add/Remove Crew: System administrators will be able to select a flight and add, remove, or modify crew member information of the flight crew.

Print List of Airlines Registered Users: System administrators will be able to view the list of users that have registered accounts with the airline system.
Add/Remove/Modify Flight Information: System administrators will be able to select a flight and add, remove, or modify the origin, destination, or departure date of the flight

Add/Remove Aircraft: System administrators will be able to add or remove aircrafts that are registered with the airline.

Add/Remove Flight Destinations: System administrators will be able to select a flight in the database and add, remove or modify it's destination

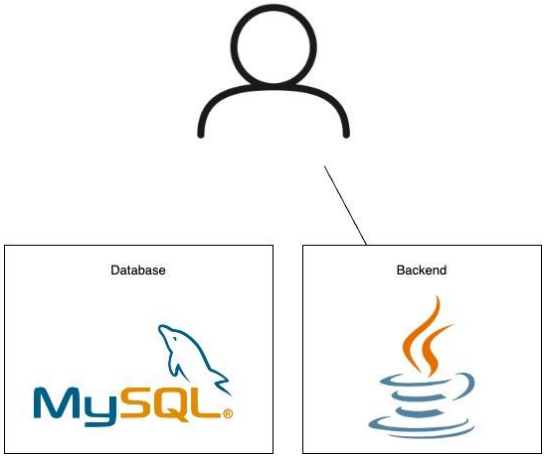
System's Conceptual Model:



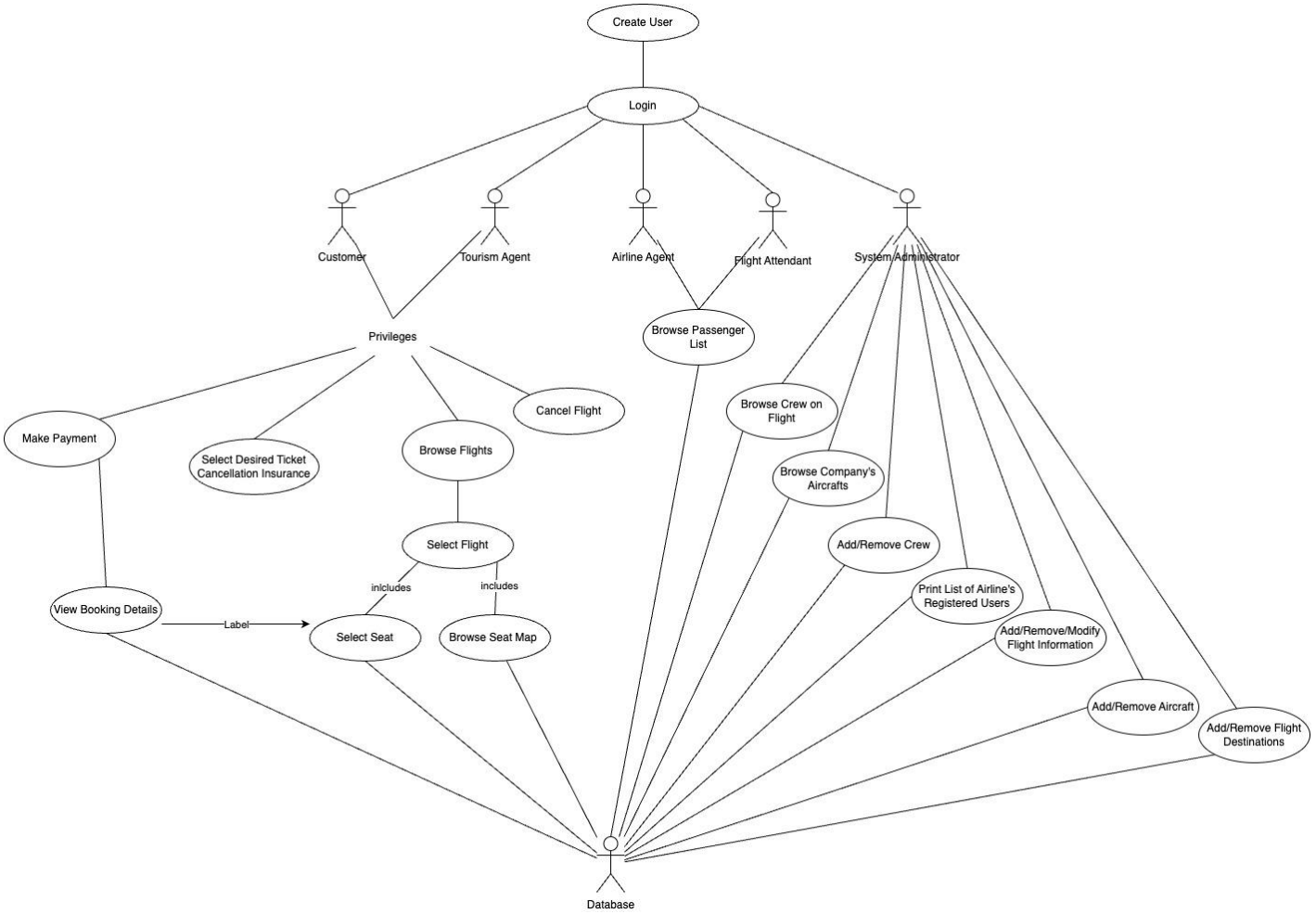
PART B:

Highlights of the system's architecture:

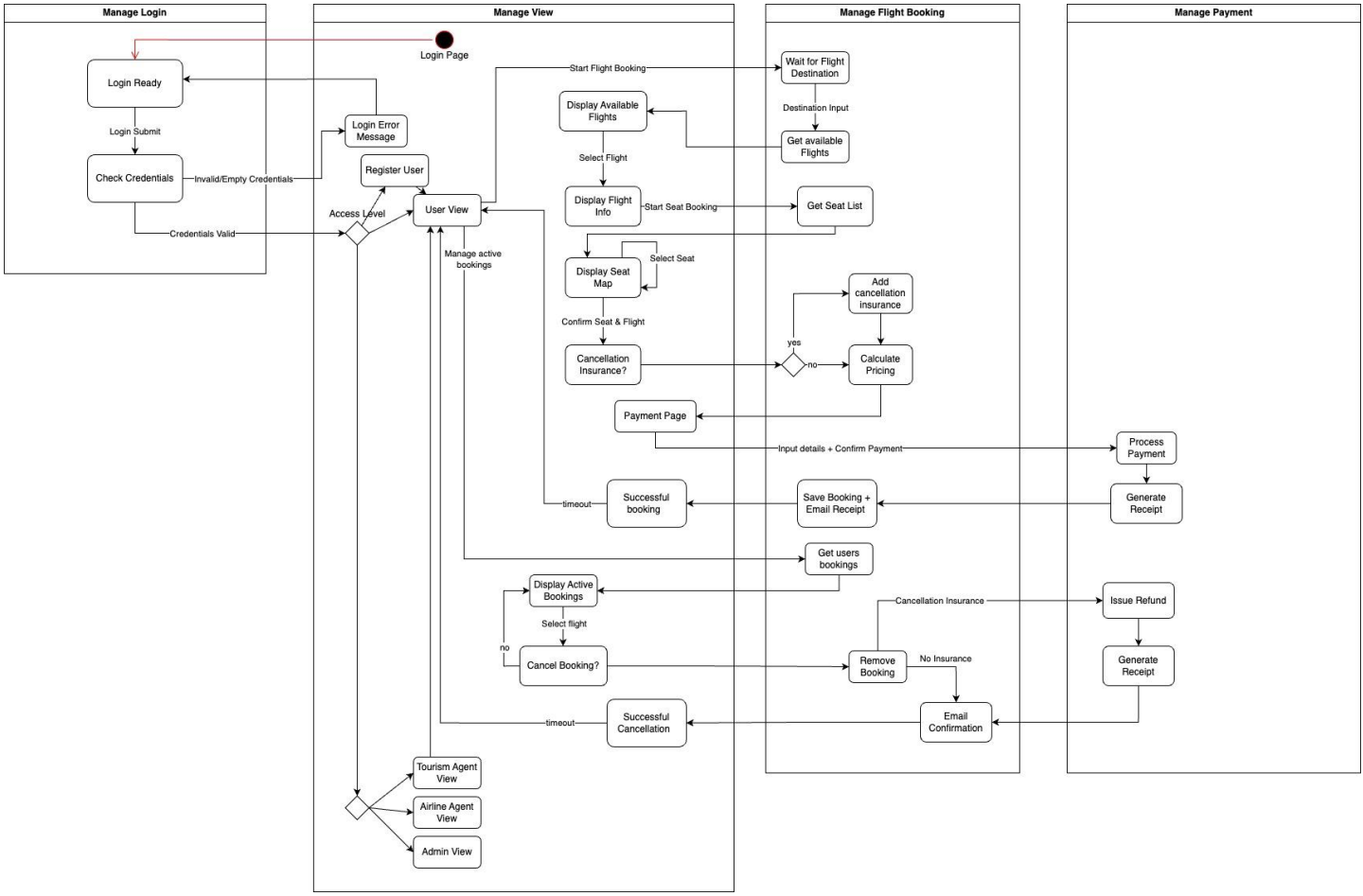
In this project, we have used Java for core functionality and MySQL for the database. We have used JDBC (MySQL Connector) in order to connect the application to the SQL database.



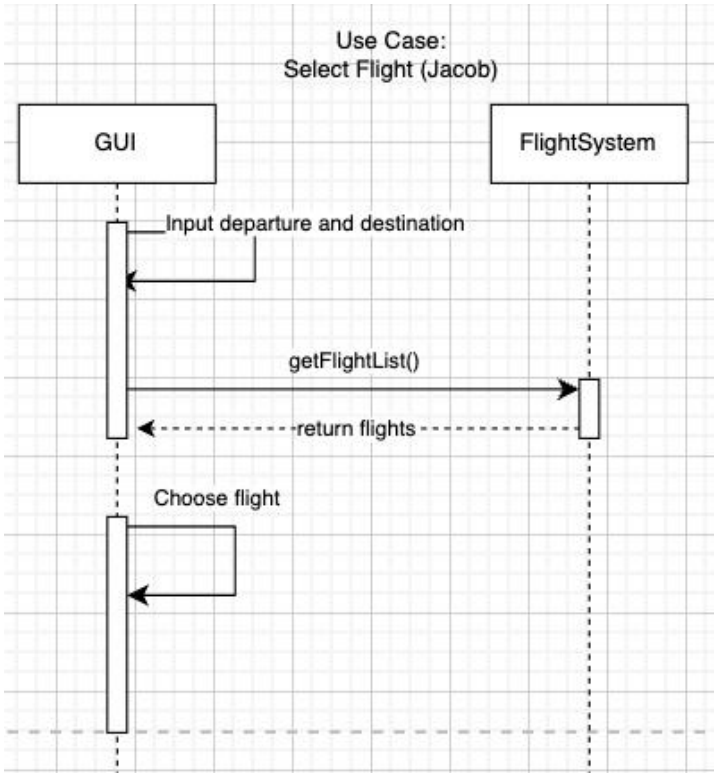
Updated use case diagram:

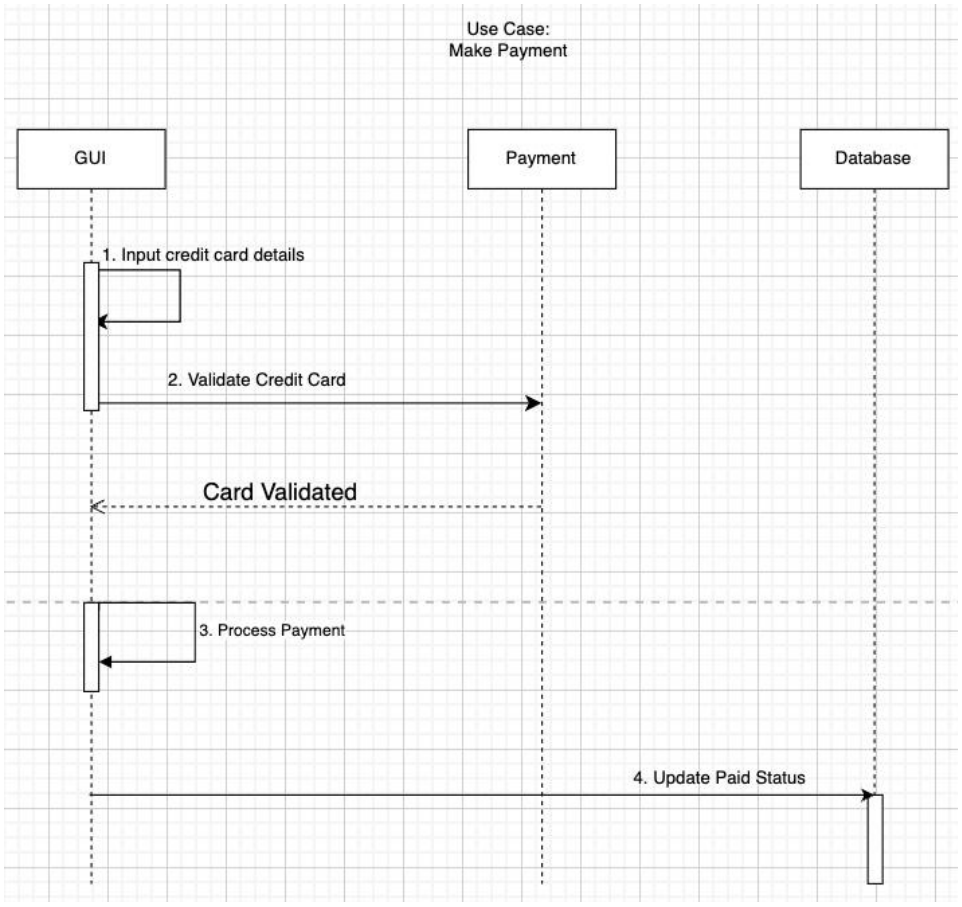


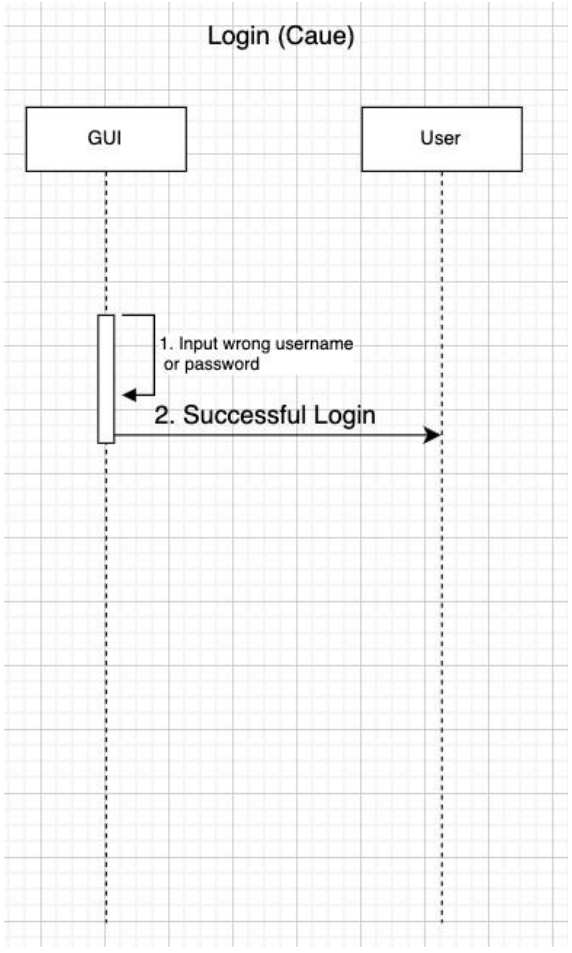
Systems activity diagram:

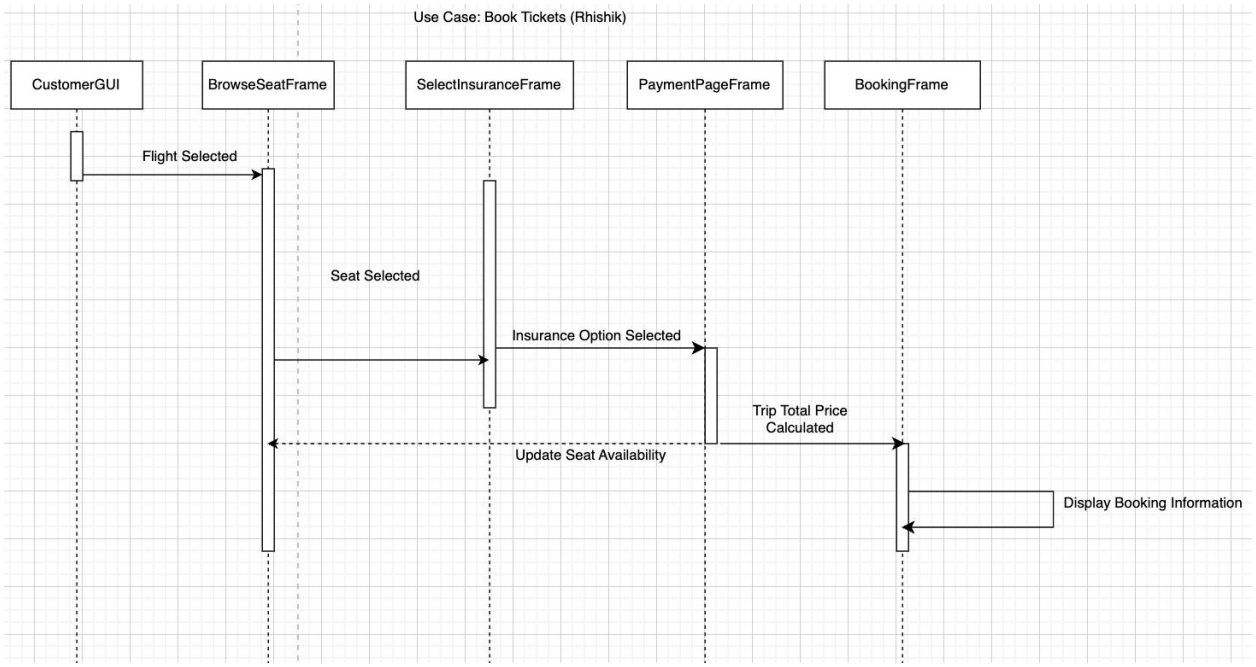


Sequence diagrams:

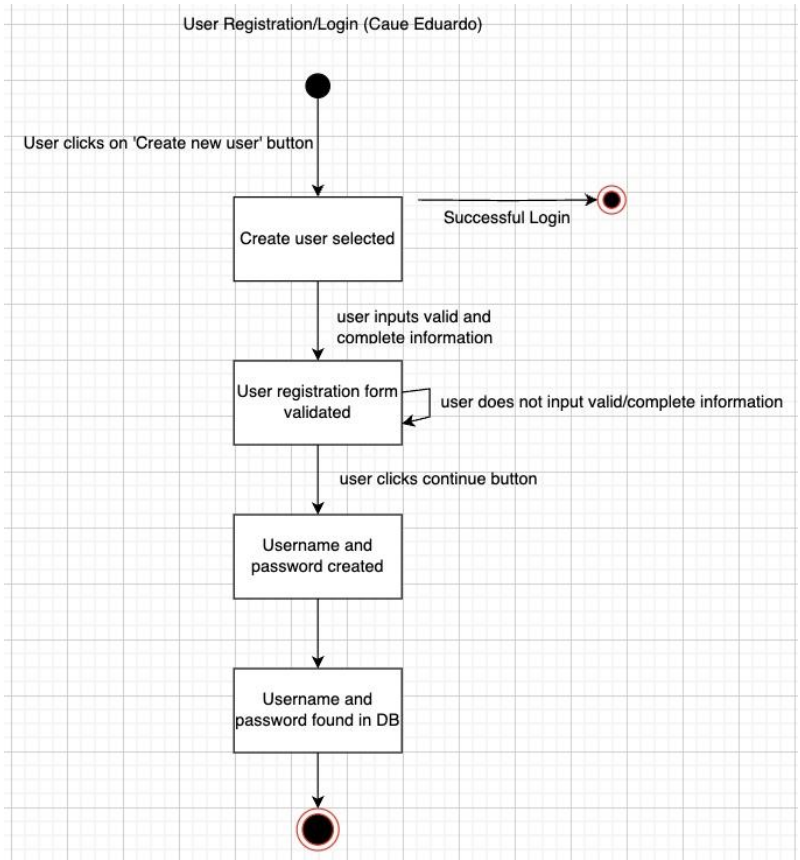


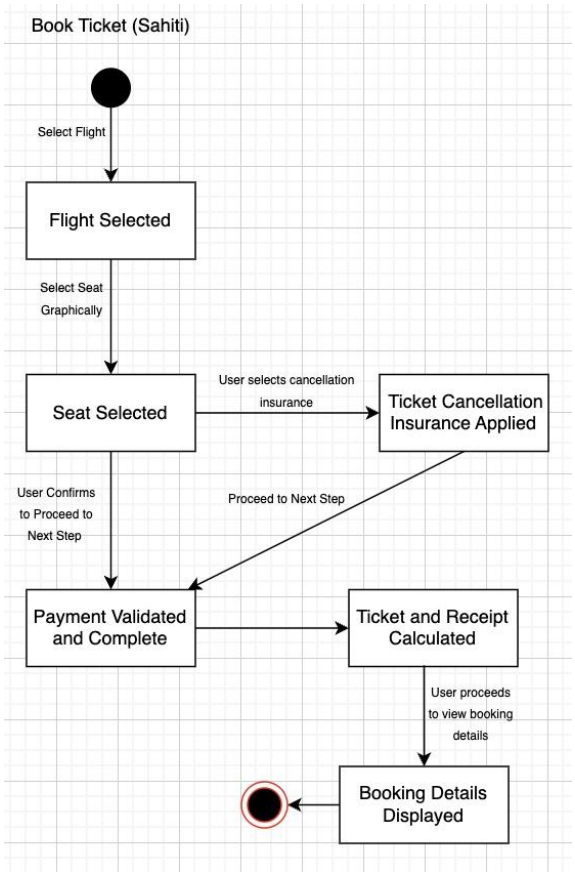

Use Case:
Make Payment
GUI
Payment
Database
1. Input credit card details
2. Validate Credit Card
Card Validated
3. Process Payment
4. Update Paid Status

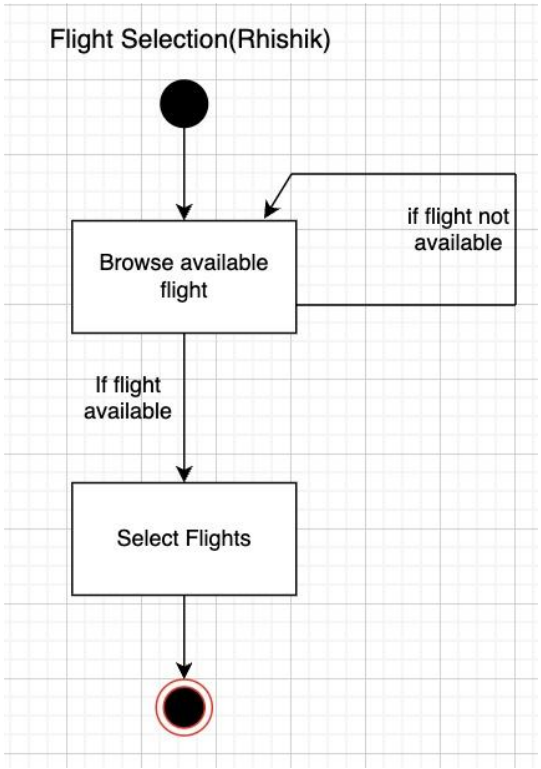

Login (Caue)
GUI
User
1. Input wrong username
or password
2. Successful Login

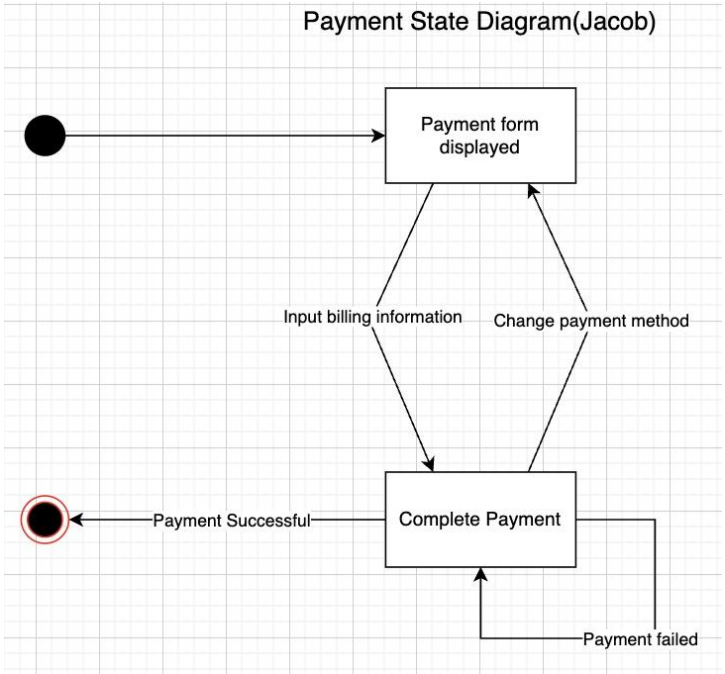


State transition diagrams:

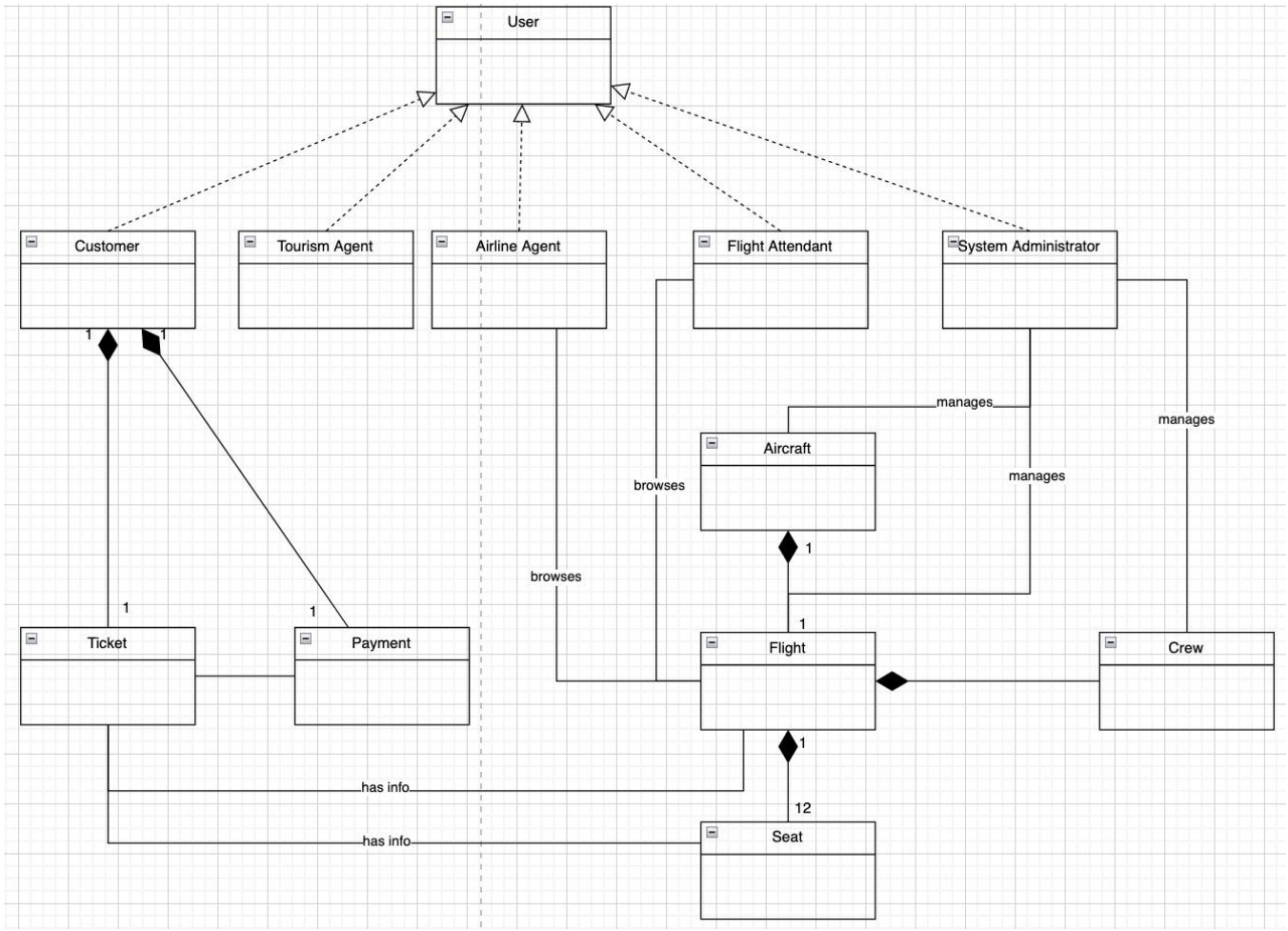



Book Ticket (Sahiti)
Select Flight
Flight Selected
Select Seat Graphically
Seat Selected
User selects cancellation insurance
Ticket Cancellation Insurance Applied
User Confirms to Proceed to Next Step
Proceed to Next Step
Payment Validated and Complete
Ticket and Receipt Calculated
User proceeds to view booking details
Booking Details Displayed

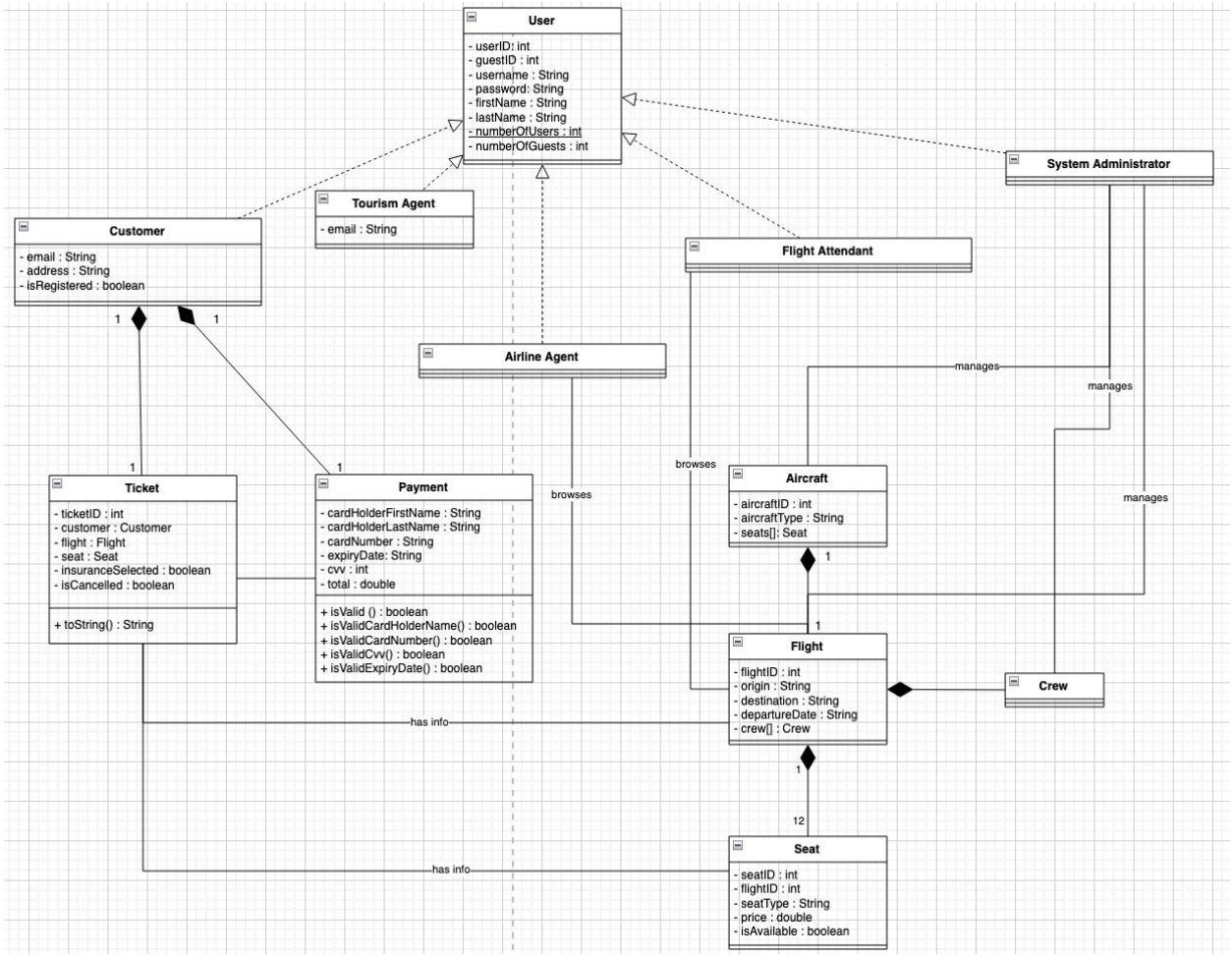

Flight Selection(Rhishik)
Browse available flight
if flight not available
If flight available
Select Flights



System's Domain class diagram (without attributes/functionalities):



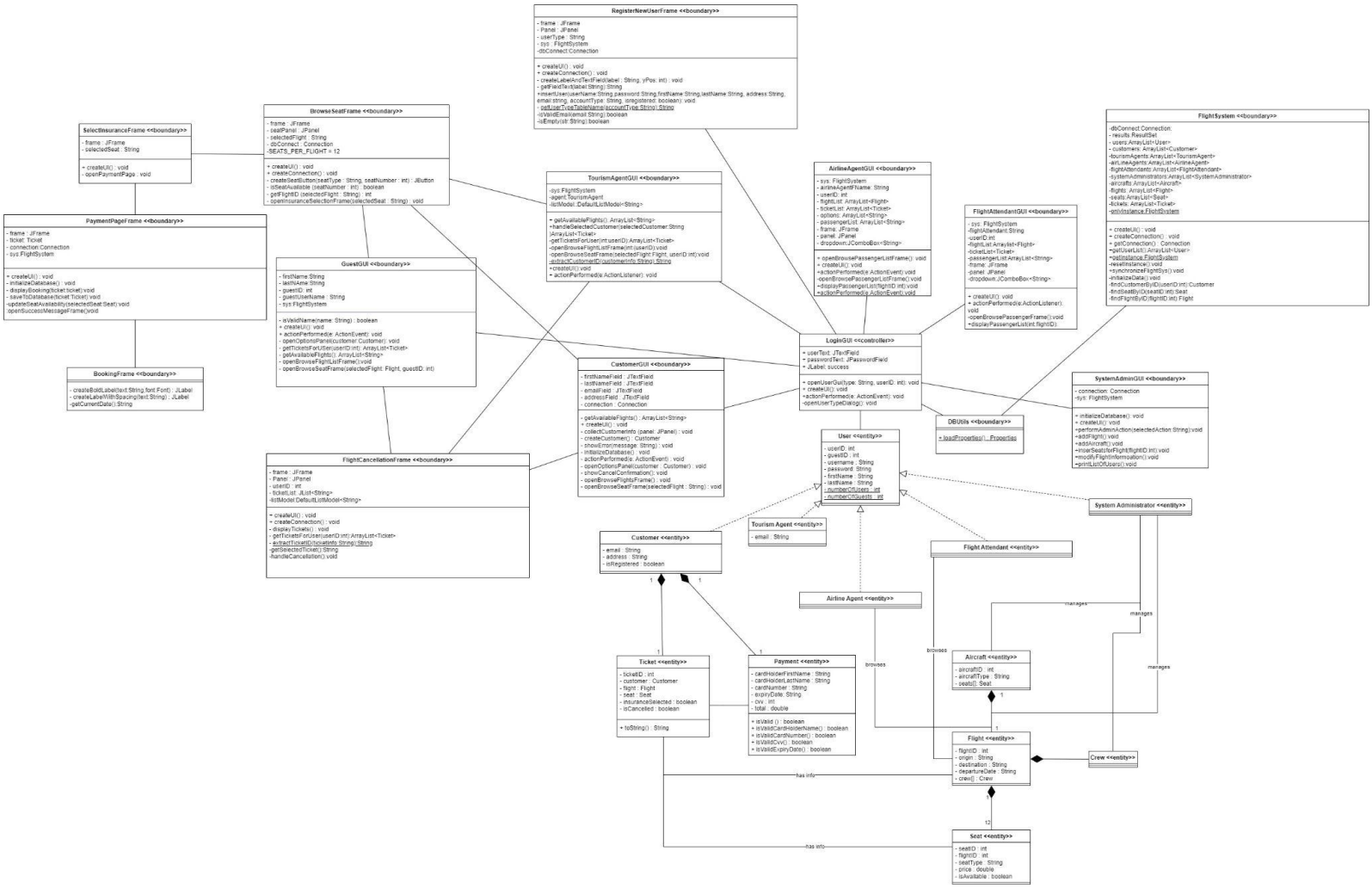
System's Domain class diagram(with attributes/functionalities):



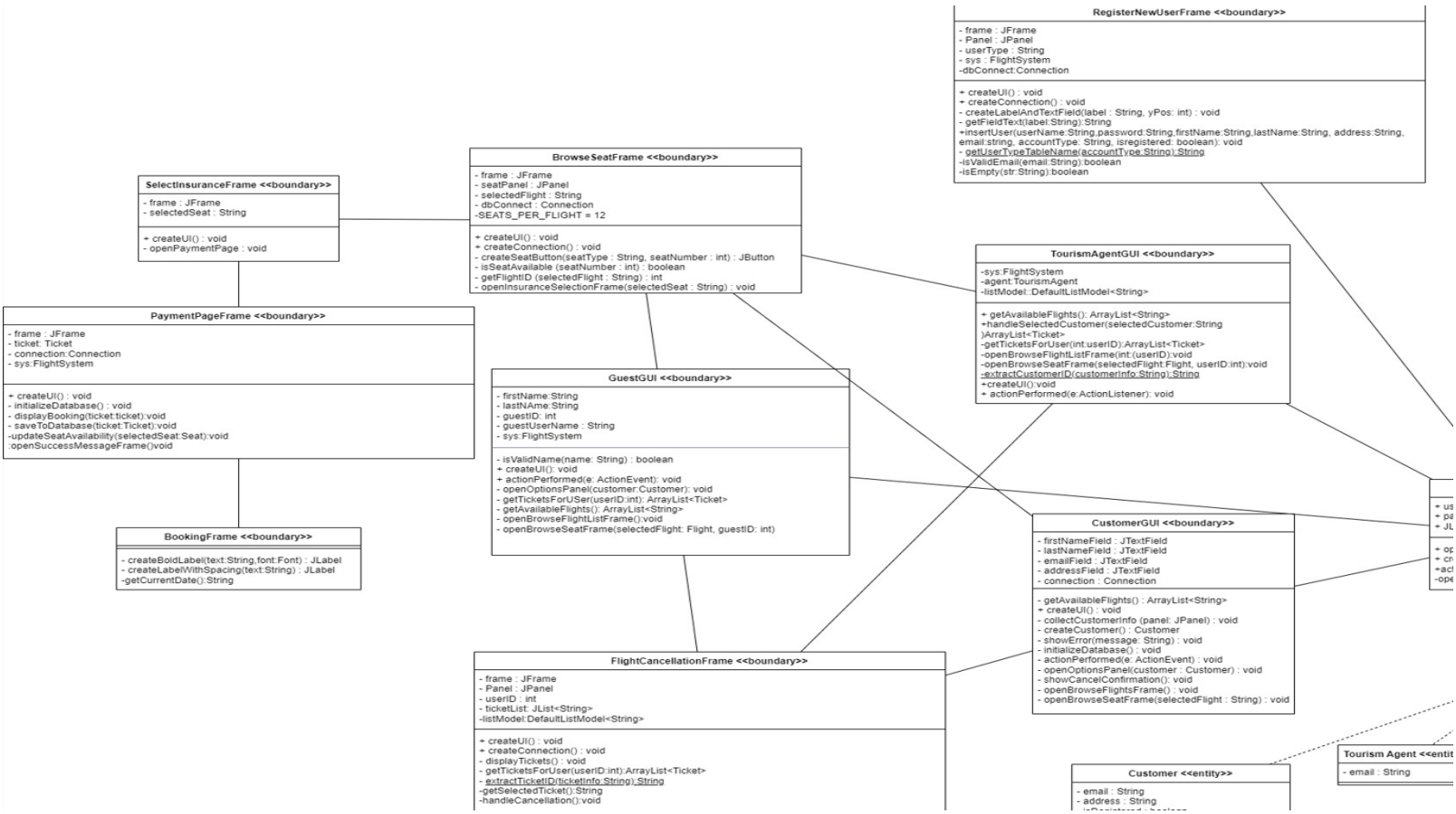
PART C:

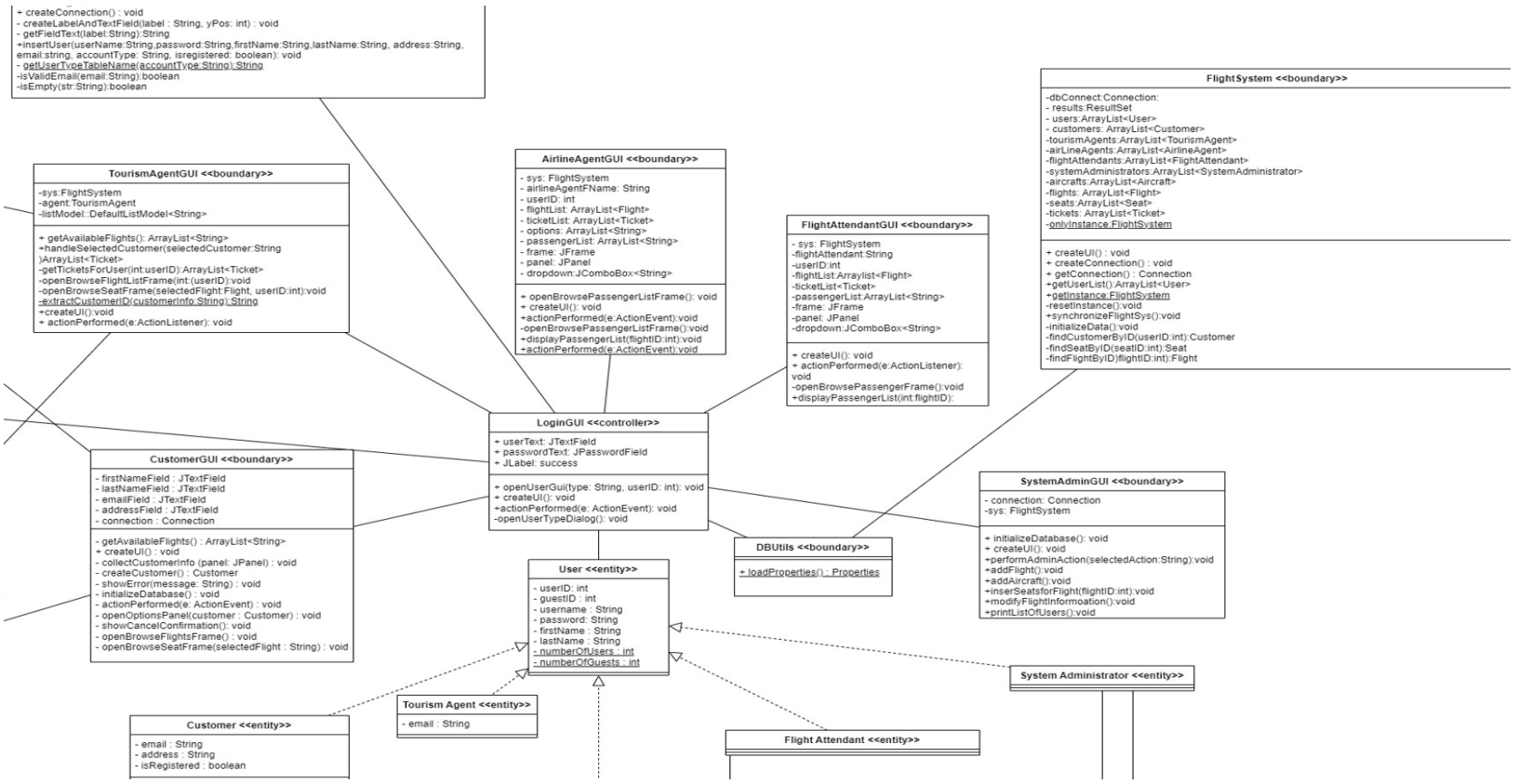
System's Detailed Design-Class Diagram:

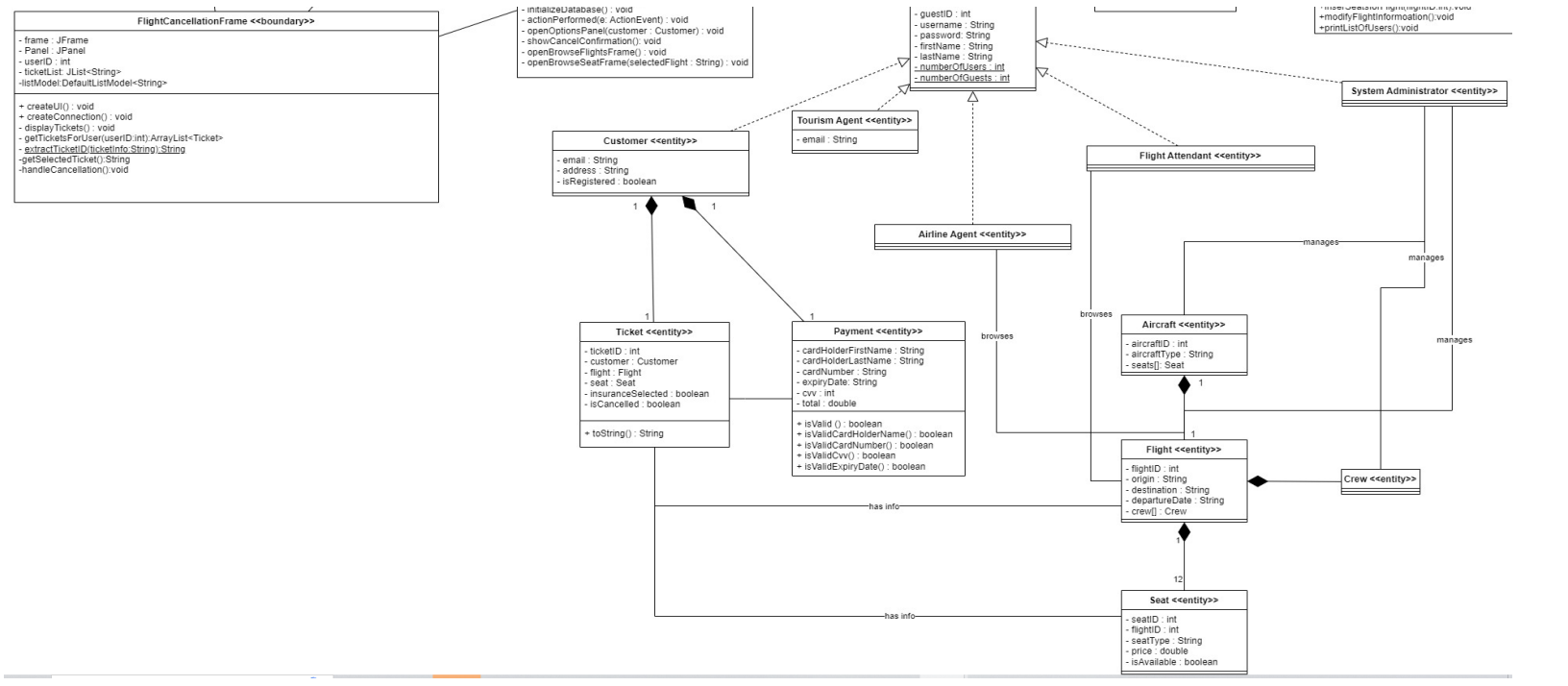
The full picture:



Zoomed view:

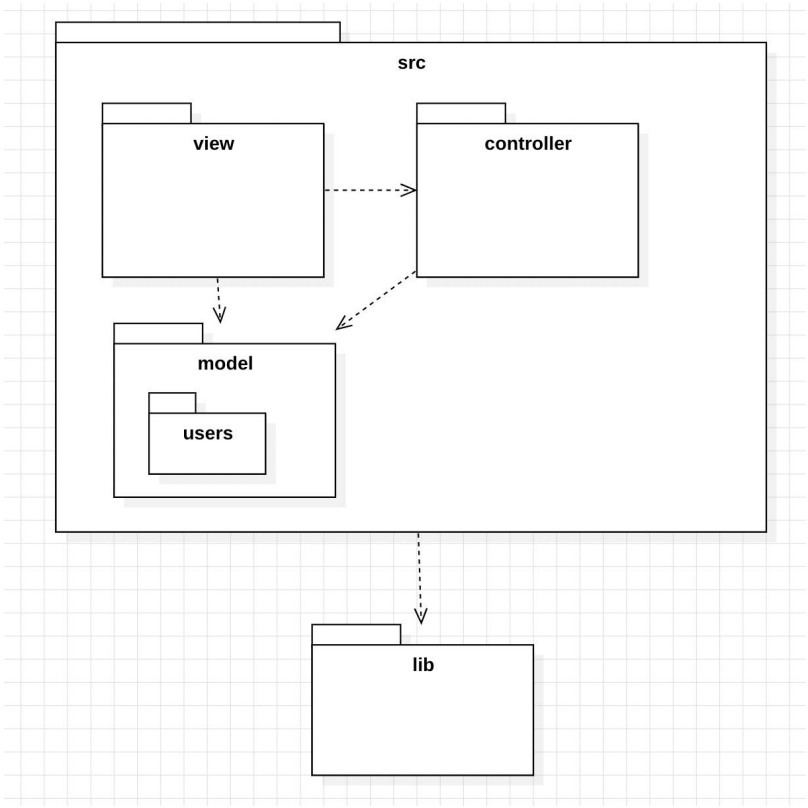

RegisterNewUserFrame <<boundary>>
- frame : JFrame
- Panel : JPanel
- userType : String
- sys : FlightSystem
-dbConnect:Connection
+ createUI() : void
+ createConnection() : void
- createLabelAndTextField(label : String, yPos: int) : void
- getFieldText(label:String):String
+insertUser(userName:String,password:String,firstName:String,lastName:String, address:String, email:string, accountType: String, isregistered: boolean): void
- getUserTypeTableName(accountType:String):String
-isValidEmail(email:String):boolean
-isEmpty(str:String):boolean
SelectInsuranceFrame <<boundary>>
- frame : JFrame
- selectedSeat : String
+ createUI() : void
- openPaymentPage : void
BrowseSeatFrame <<boundary>>
- frame : JFrame
- seatPanel : JPanel
- selectedFlight : String
- dbConnect : Connection
-SEATS_PER_FLIGHT = 12
+ createUI() : void
+ createConnection() : void
- createSeatButton(seatType : String, seatNumber : int) : JButton
- isSeatAvailable (seatNumber : int) : boolean
- getFlightID (selectedFlight : String) : int
- openInsuranceSelectionFrame(selectedSeat : String) : void
TourismAgentGUI <<boundary>>
-sys:FlightSystem
-agent:TourismAgent
-listModel::DefaultListModel<String>
+ getAvailableFlights(): ArrayList<String>
+handleSelectedCustomer(selectedCustomer:String
)ArrayList<Ticket>
-getTicketsForUser(int:userID):ArrayList<Ticket>
-openBrowseFlightListFrame(int:(userID):void
-openBrowseSeatFrame(selectedFlight:Flight, userID:int):void
-extractCustomerID(customerInfo:String):String
+createUI():void
+ actionPerformed(e:ActionListener): void
PaymentPageFrame <<boundary>>
- frame : JFrame
- ticket: Ticket
- connection:Connection
- sys:FlightSystem
+ createUI() : void
- initializeDatabase() : void
- displayBooking(ticket:ticket):void
- saveToDatabase(ticket:Ticket):void
-updateSeatAvailability(selectedSeat:Seat):void
:openSuccessMessageFrame()void
GuestGUI <<boundary>>
- firstName:String
- lastNAme:String
- guestID: int
- guestUserName : String
- sys:FlightSystem
- isValidName(name: String) : boolean
+ createUI(): void
+ actionPerformed(e: ActionEvent): void
- openOptionsPanel(customer:Customer): void
- getTicketsForUSer(userID:int): ArrayList<Ticket>
- getAvailableFlights(): ArrayList<String>
- openBrowseFlightListFrame():void
- openBrowseSeatFrame(selectedFlight: Flight, guestID: int)
BookingFrame <<boundary>>
- createBoldLabel(text:String,font:Font) : JLabel
- createLabelWithSpacing(text:String) : JLabel
-getCurrentDate():String
CustomerGUI <<boundary>>
- firstNameField : JTextField
- lastNameField : JTextField
- emailField : JTextField
- addressField : JTextField
- connection : Connection
- getAvailableFlights() : ArrayList<String>
+ createUI() : void
- collectCustomerInfo (panel: JPanel) : void
- createCustomer() : Customer
- showError(message: String) : void
- initializeDatabase() : void
- actionPerformed(e: ActionEvent) : void
- openOptionsPanel(customer : Customer) : void
- showCancelConfirmation(): void
- openBrowseFlightsFrame() : void
- openBrowseSeatFrame(selectedFlight : String) : void
FlightCancellationFrame <<boundary>>
- frame : JFrame
- Panel : JPanel
- userID : int
- ticketList: JList<String>
-listModel:DefaultListModel<String>
+ createUI() : void
+ createConnection() : void
- displayTickets() : void
- getTicketsForUser(userID:int):ArrayList<Ticket>
- extractTicketID(ticketInfo:String):String
-getSelectedTicket():String
-handleCancellation():void
Customer <<entity>>
- email : String
- address : String
Tourism Agent <<entit
- email : String


+ createConnection() : void
- createLabelAndTextField(label : String, yPos: int) : void
- getFieldText(label:String):String
+insertUser(userName:String,password:String,firstName:String,lastName:String, address:String, email:string, accountType: String, isregistered: boolean): void
- getUserTypeTableName(accountType:String):String
-isValidEmail(email:String):boolean
-isEmpty(str:String):boolean
TourismAgentGUI <<boundary>>
-sys:FlightSystem
-agent:TourismAgent
-listModel::DefaultListModel<String>
+ getAvailableFlights(): ArrayList<String>
+handleSelectedCustomer(selectedCustomer:String)ArrayList<Ticket>
-getTicketsForUser(int:userID):ArrayList<Ticket>
-openBrowseFlightListFrame(int:(userID):void
-openBrowseSeatFrame(selectedFlight:Flight, userID:int):void
-extractCustomerID(customerInfo:String):String
+createUI():void
+ actionPerformed(e:ActionListener): void
AirlineAgentGUI <<boundary>>
- sys: FlightSystem
- airlineAgentFName: String
- userID: int
- flightList: ArrayList<Flight>
- ticketList: ArrayList<Ticket>
- options: ArrayList<String>
- passengerList: ArrayList<String>
- frame: JFrame
- panel: JPanel
- dropdown:JComboBox<String>
+ openBrowsePassengerListFrame(): void
+ createUI(): void
+actionPerformed(e:ActionEvent):void
-openBrowsePassengerListFrame():void
+displayPassengerList(flightID:int):void
+actionPerformed(e:ActionEvent):void
FlightAttendantGUI <<boundary>>
- sys: FlightSystem
-flightAttendant:String
-userID:int
-flightList:Arraylist<Flight>
-ticketList<Ticket>
-passengerList:ArrayList<String>
-frame: JFrame
-panel: JPanel
-dropdown:JComboBox<String>
+ createUI(): void
+ actionPerformed(e:ActionListener): void
-openBrowsePassengerFrame():void
+displayPassengerList(int:flightID):
FlightSystem <<boundary>>
-dbConnect:Connection:
- results:ResultSet
- users:ArrayList<User>
- customers: ArrayList<Customer>
-tourismAgents:ArrayList<TourismAgent>
-airLineAgents:ArrayList<AirlineAgent>
-flightAttendants:ArrayList<FlightAttendant>
-systemAdministrators:ArrayList<SystemAdministrator>
-aircrafts:ArrayList<Aircraft>
-flights: ArrayList<Flight>
-seats:ArrayList<Seat>
-tickets: ArrayList<Ticket>
-onlyInstance:FlightSystem
+ createUI() : void
+ createConnection() : void
+ getConnection() : Connection
+getUserList():ArrayList<User>
+getInstance:FlightSystem
-resetInstance():void
+synchronizeFlightSys():void
-initializeData():void
-findCustomerByID(userID:int):Customer
-findSeatByID(seatID:int):Seat
-findFlightByID(flightID:int):Flight
LoginGUI <<controller>>
+ userText: JTextField
+ passwordText: JPasswordField
+ JLabel: success
+ openUserGui(type: String, userID: int): void
+ createUI(): void
+actionPerformed(e: ActionEvent): void
-openUserTypeDialog(): void
CustomerGUI <<boundary>>
- firstNameField : JTextField
- lastNameField : JTextField
- emailField : JTextField
- addressField : JTextField
- connection : Connection
- getAvailableFlights() : ArrayList<String>
+ createUI() : void
- collectCustomerInfo (panel: JPanel) : void
- createCustomer() : Customer
- showError(message: String) : void
- initializeDatabase() : void
- actionPerformed(e: ActionEvent) : void
- openOptionsPanel(customer : Customer) : void
- showCancelConfirmation(): void
- openBrowseFlightsFrame() : void
- openBrowseSeatFrame(selectedFlight : String) : void
SystemAdminGUI <<boundary>>
- connection: Connection
-sys: FlightSystem
+ initializeDatabase(): void
+ createUI(): void
+performAdminAction(selectedAction:String):void
+addFlight():void
+addAircraft():void
+inserSeatsforFlight(flightID:int):void
+modifyFlightInformoation():void
+printListOfUsers():void
DBUtils <<boundary>>
+ loadProperties() : Properties
User <<entity>>
- userID: int
- guestID : int
- username : String
- password: String
- firstName : String
- lastName : String
- numberOfUsers : int
- numberOfGuests : int
Customer <<entity>>
- email : String
- address : String
- isRegistered : boolean
Tourism Agent <<entity>>
- email : String
Flight Attendant <<entity>>
System Administrator <<entity>>


FlightCancellationFrame <<boundary>>
- frame : JFrame
- Panel : JPanel
- userID : int
- ticketList: JList<String>
-listModel:DefaultListModel<String>
+ createUI() : void
+ createConnection() : void
- displayTickets() : void
- getTicketsForUser(userID:int):ArrayList<Ticket>
- extractTicketID(ticketInfo:String):String
-getSelectedTicket():String
-handleCancellation():void
- initializeDatabase() : void
- actionPerformed(e: ActionEvent) : void
- openOptionsPanel(customer : Customer) : void
- showCancelConfirmation(): void
- openBrowseFlightsFrame() : void
- openBrowseSeatFrame(selectedFlight : String) : void
- guestID : int
- username : String
- password: String
- firstName : String
- lastName : String
- numberOfUsers : int
- numberOfGuests : int
+modifyFlightInformoation():void
+printListOfUsers():void
Customer <<entity>>
- email : String
- address : String
- isRegistered : boolean
Tourism Agent <<entity>>
- email : String
System Administrator <<entity>>
Flight Attendant <<entity>>
Airline Agent <<entity>>
manages
manages
manages
browses
browses
Ticket <<entity>>
- ticketID : int
- customer : Customer
- flight : Flight
- seat : Seat
- insuranceSelected : boolean
- isCancelled : boolean
+ toString() : String
Payment <<entity>>
- cardHolderFirstName : String
- cardHolderLastName : String
- cardNumber : String
- expiryDate: String
- cvv : int
- total : double
+ isValid () : boolean
+ isValidCardHolderName() : boolean
+ isValidCardNumber() : boolean
+ isValidCvv() : boolean
+ isValidExpiryDate() : boolean
Aircraft <<entity>>
- aircraftID : int
- aircraftType : String
- seats[]: Seat
Flight <<entity>>
- flightID : int
- origin : String
- destination : String
- departureDate : String
- crew[] : Crew
Crew <<entity>>
has info
has info
Seat <<entity>>
- seatID : int
- flightID : int
- seatType : String
- price : double
- isAvailable : boolean
1
1
1
1
1
1
1
12

PART D:

Package Diagram:



Deployment Diagram: