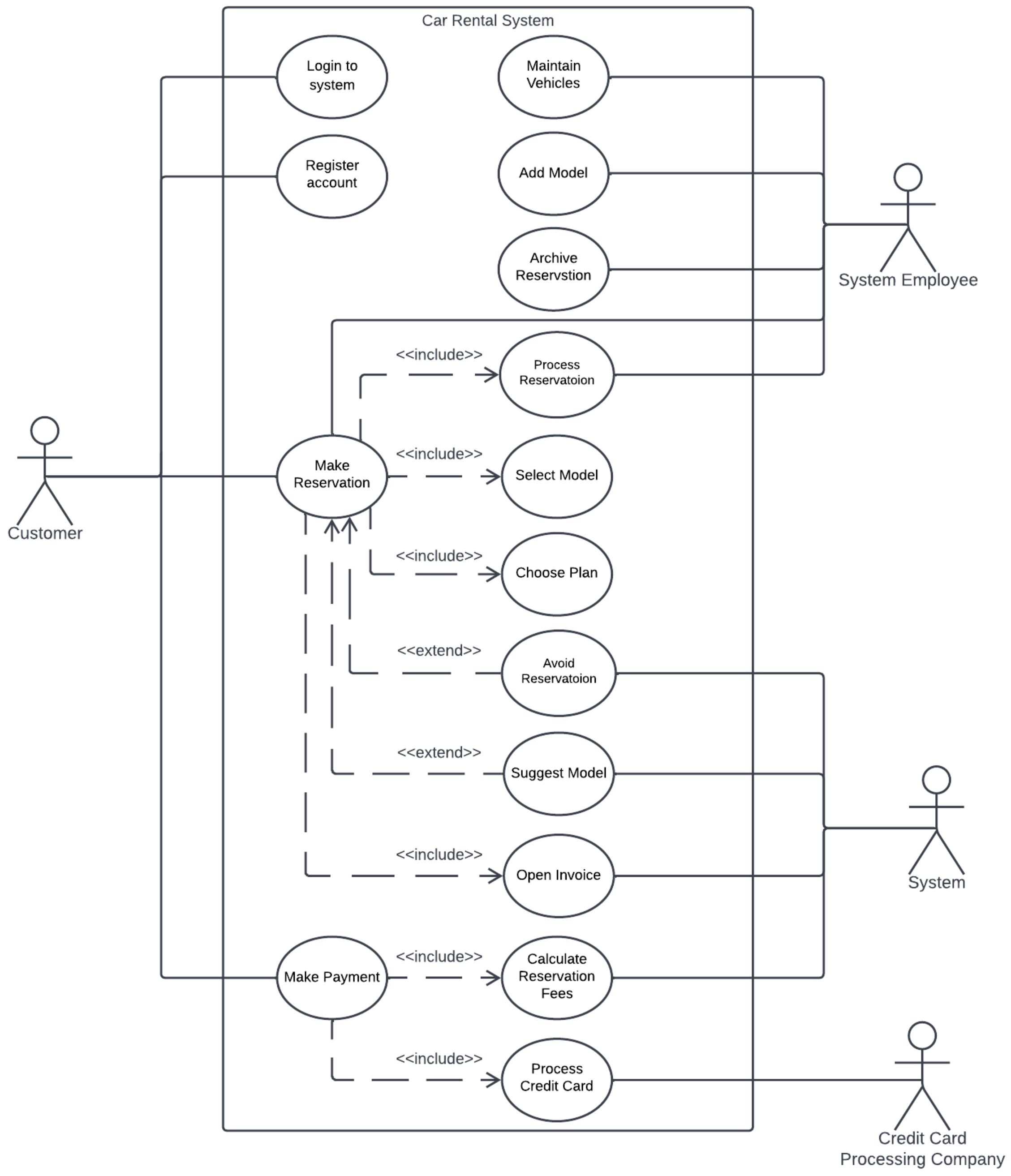
1. Use-Cases and scenarios

1.1. Use Case Diagram



1.2. Use Case Scenarios

Use case name:	Add model
Participating actors:	System Employee
Entry conditions:	The System Employee wants to index a new vehicle model to the system.
Flow of events:	1. The System Employee logs in to the company website admin dashboard and then enters the add model section. 2. The System Employee starts to fill in the vehicle information in the add model form, e.g., manufacturer, model name, model type, release year, transmission mode, price class, and the number of doors. After completing that form, the System Employee submits the vehicle information by pressing the save model button. 3. The save model button triggers the add model control that validates and processes the information presented in the add model form.
Exit conditions:	The System Employee has received a positive acknowledgment from the add model control object indicating successful indexing of a new vehicle model to the system and the models' database has been updated to accommodate changes.

Exception Conditions:	Invalid vehicle information exception initiated by add model control object.

Use case name:	Make a reservation
Participating actors:	1. Customer 2. System Employee
Entry conditions:	1. The Customer enters the company website searching for a vehicle for rental. 2. The Customer goes to the company location and asks the System Employee for a vehicle reservation.
Flow of events:	1. The Customer or System Employee starts to fill in the required vehicle model preferences section on the make reservation form, e.g, manufacturer, model name, model type, release year, transmission mode, price class, and number of doors. 2. Customer or System Employee invokes "select model" use case to find required models. Otherwise, the customer or System Employee can press the cancel reservation button. 3. If the filtering preferences have no search results, the Customer or System Employee will invoke the "suggest

	model" use case. 4. Customer or System Employee can continue selecting other models for reservation by repeating the above 3 steps. 5. After selecting the required model(s), the Customer or System Employee selects the vehicle rental plan by invoking the “select plan” use case and then, the Customer or System Employee should write down the rental start and end dates. 6. The System Employee processes the reservations manually using a reservation form and archives reservations in the file cabinet. Also, System sends a receipt for the customer with the reservation details. 7. The Customer should go to the company to sign the contract and get the vehicle (s). 8. As soon as a car is checked out to a customer, an invoice is opened by the system. 9. If the customer does not show up to sign the contract in the specified period of time, the reservation is avoided.
Exit conditions:	1. The customer has received a receipt of reservation details.

	2. Customer or system employee pressed on the cancel reservation button. 3. The Customer does not show up to sign the contract in the specified period of time.
Exception Conditions:	None

Use case name:	Select model
Participating actors:	1. Customer 2. System Employee
Entry conditions:	The customer or System Employee is logged in to the company website and the make reservation landing page is currently opened.
Flow of events:	1. After the customer or System Employee writes the filtering preferences for the available model for rentals, the customer should press on the show models button. 2. The show models button will invoke the select model control to create an interface of available models fetched from the system database that match customer filtering criteria.
Exit conditions:	1. The customer has selected a vehicle model for reservation. 2. There is no vehicle model that

	matches customer needs.
Exception Conditions:	None

Use case name:	Suggest model
Participating actors:	1. Customer 2. System Employee
Entry conditions:	1. Customer has not found vehicle models that match his needs. 2. System Employee has not found vehicle models that match customer needs.
Flow of events:	1. Customer or System Employee press on the suggest model button to search for a model that best meets vehicle filtering preferences. 2. The system traverses the models database and filters models using suggest model control. 3. An interface of vehicle models returned back to the customer to choose from.
Exit conditions:	Customer or System Employee has found a model that best matches model filtering criteria.
Exception Conditions:	Suggest model control returns negative

	acknowledgement indicating no search results for specified model selection criteria.

Use case name:	Make payment
Participating actors:	1. Customer 2. Credit Card Processing Company 3. System Employee
Entry conditions:	1. Customer returned the vehicle back to the company and wants to check out the invoice.
Flow of events:	1. For making a payment, customers can go to the company location to pay rental fees in cash, otherwise, customers can pay rental fees anywhere using a credit card. 2. If Customer goes to the company website, the System Employee checks the opened rental invoice on the system to get the reservation fees. 3. Otherwise, if the customer used a credit card for paying rental fees, customer credit card details will be passed for a credit processing company. That company is responsible for transferring the rental fees for customer bank accounts to the car rental company account after

	processing and validating customer credit card information.
Exit conditions:	1. The Customer paid the rental fees in cash for the system employee. 2. Rental charges have been withdrawn from the customer bank account by the credit card processing company.
Exception Conditions:	Credit card processing company returns negative acknowledgement indicating that the customer bank account is out of money.

2. Identifying entity, boundary, and control objects of Use -Cases.

2.1. Add model

Entity Object	• System Employee
Boundary Object	• add model. • save model button
Control Object	• add model control

2.2. Make a reservation

Entity Object	• System Employee. • Customer. • File cabinet
Boundary Object	• make reservation form. • cancel reservation.
Control Object	• make reservation control.

2.3. Select model

Entity Object	• System Employee. • Customer.

Boundary Object	● show model’s button.
Control Object	● select model control.

2.4. Suggest model

Entity Object	● System Employee. ● Customer.
Boundary Object	● suggest model button.
Control Object	● suggest model control.

2.5. Make payment

Entity Object	● Customer.
Boundary Object	● Return car button.
Control Object	● Payment control.

3. Identifying Class, Responsibilities, and Collaborators (CRC) cards

Customer	
Responsibilities	Collaborators
Track all information related to a single Customer. Make Reservations. Make Payment.	User

Vehicle	
Responsibilities	Collaborators
Track all information related to a single Single.	SystemEmployee

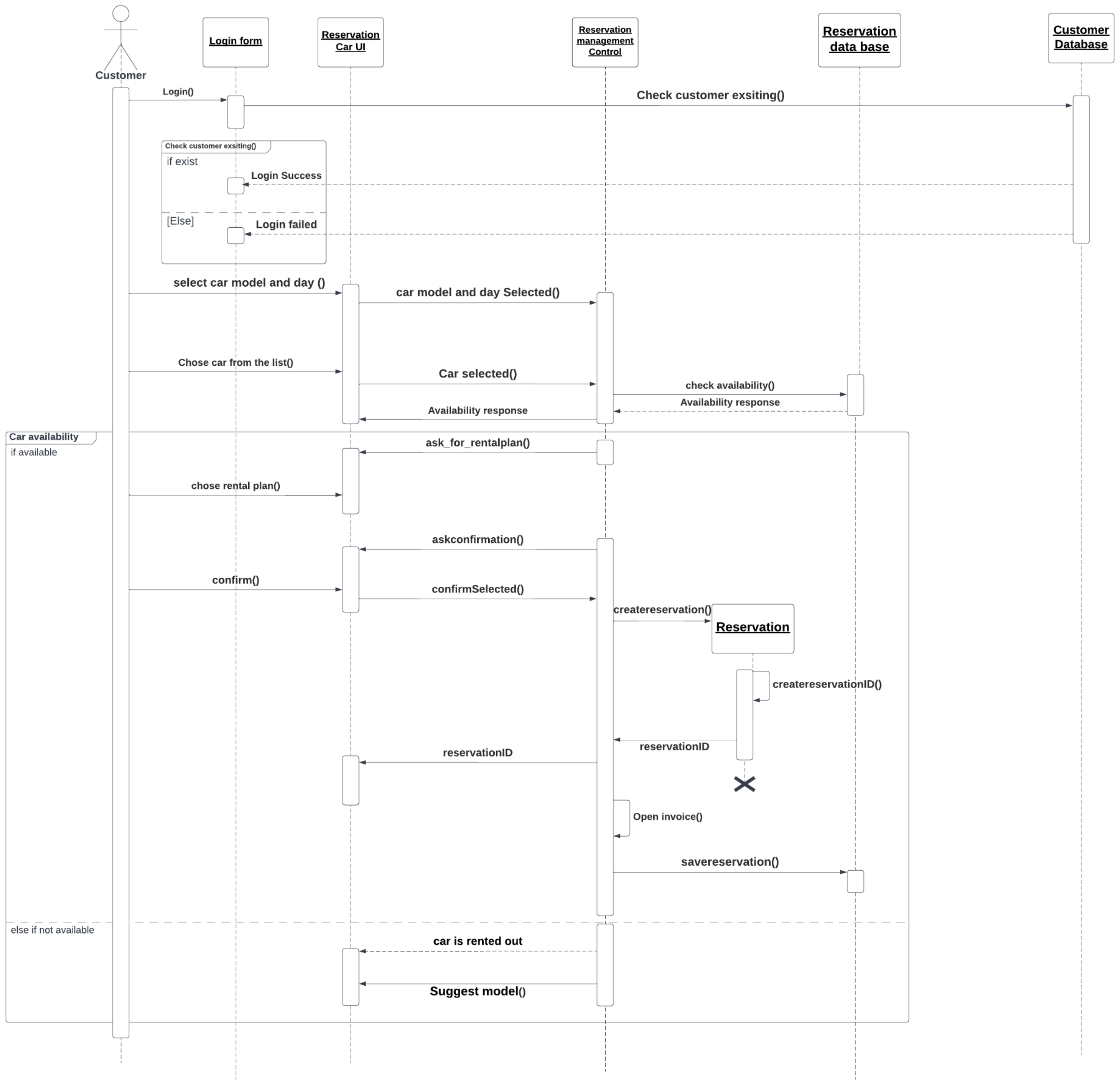
Reservations	
Responsibilities	Collaborators
Controls sequence of making new Reservations. Track all information related to a Reservations.	User SystemEmployee

Payment	
Responsibilities	Collaborators
Controls sequence of making new Payment. Track all information related to a Payment. CalculateReservationFees.	User Reservations

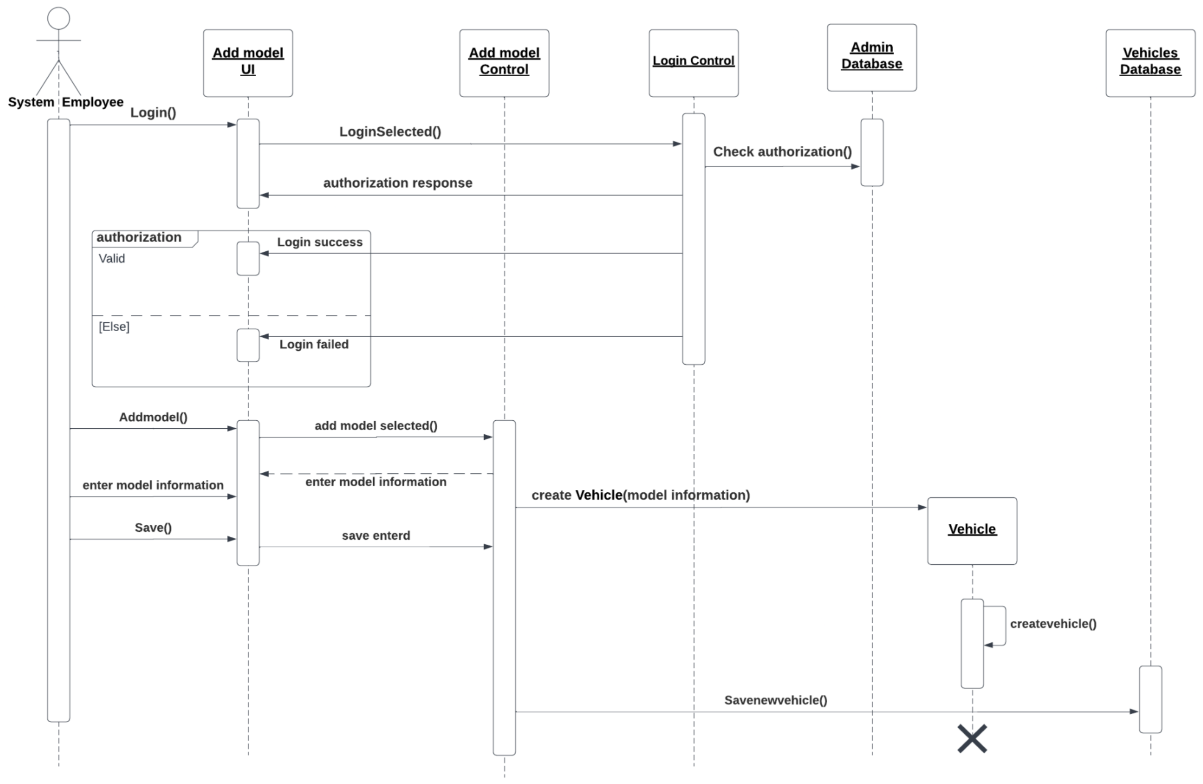
SystemEmployee	
Responsibilities	Collaborators
track and all control the information of the system. MaintainVehicles AddModel ArchiveReservstion	User

4. Sequence diagrams

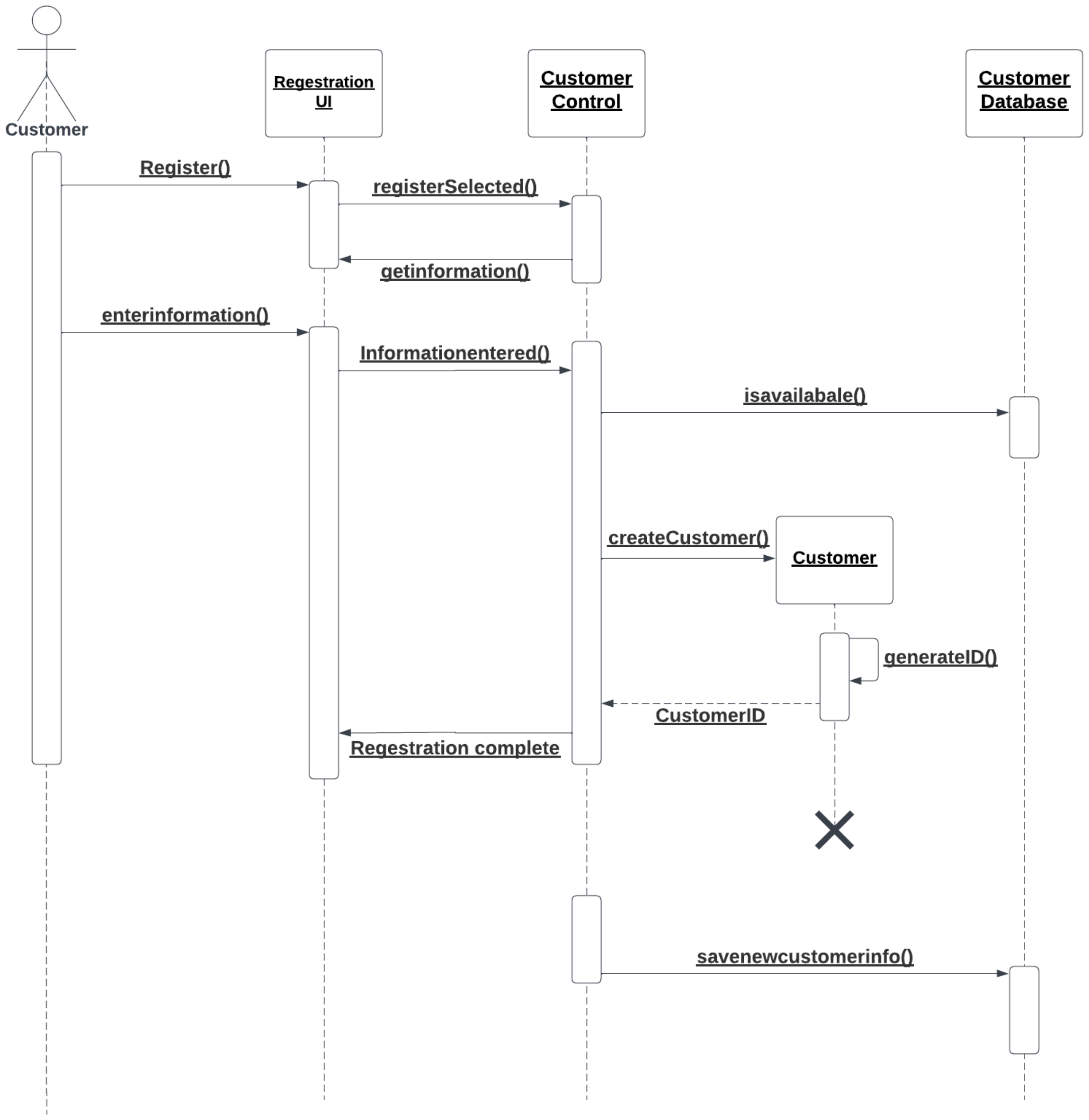
4.1. Reservation Sequence diagram



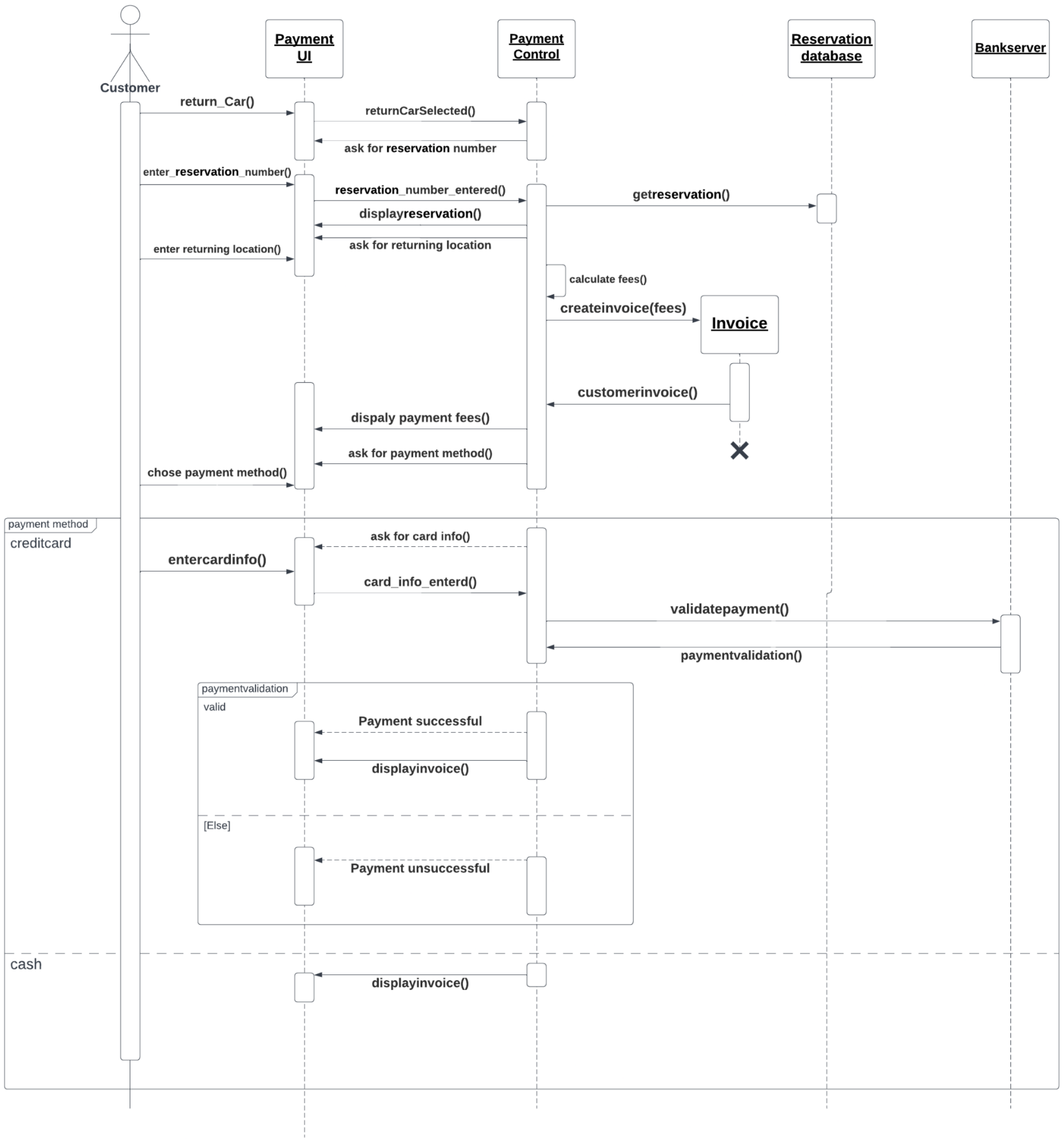
4.2. Add Model Sequence diagram



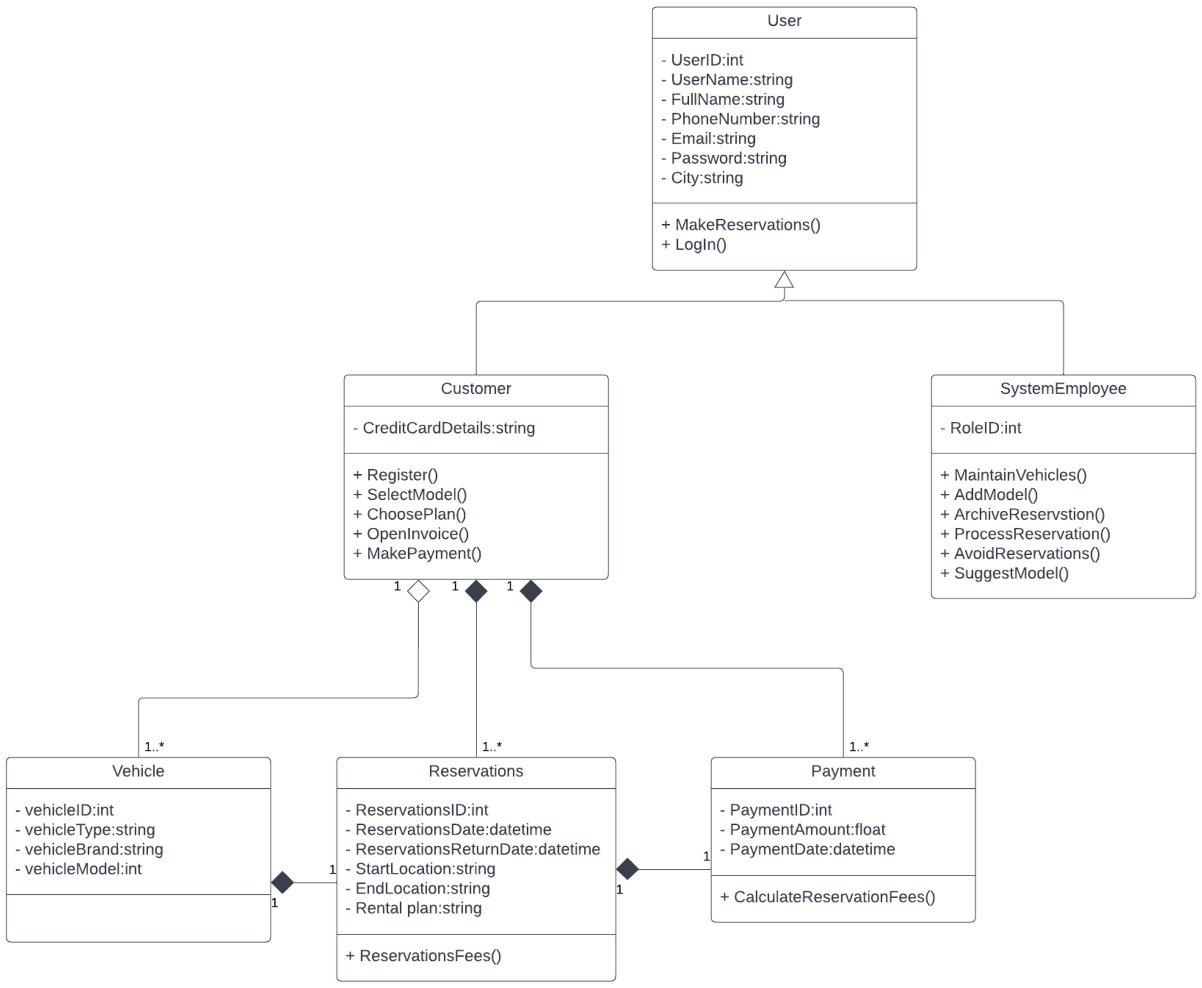
4.3. Registration Sequence diagram



4.4. Payment Sequence diagram



5. Class diagram


User
- UserID:int
- UserName:string
- FullName:string
- PhoneNumber:string
- Email:string
- Password:string
- City:string
+ MakeReservations()
+ LogIn()
Customer
- CreditCardDetails:string
+ Register()
+ SelectModel()
+ ChoosePlan()
+ OpenInvoice()
+ MakePayment()
SystemEmployee
- RoleID:int
+ MaintainVehicles()
+ AddModel()
+ ArchiveReservstion()
+ ProcessReservation()
+ AvoidReservations()
+ SuggestModel()
1
1
1
1..*
1..*
1..*
Vehicle
- vehicleID:int
- vehicleType:string
- vehicleBrand:string
- vehicleModel:int
Reservations
- ReservationsID:int
- ReservationsDate:datetime
- ReservationsReturnDate:datetime
- StartLocation:string
- EndLocation:string
- Rental plan:string
+ ReservationsFees()
Payment
- PaymentID:int
- PaymentAmount:float
- PaymentDate:datetime
+ CalculateReservationFees()
1
1
1
1