CS 6359 Individual Assignment #2:

Name: Naveenraj Palanisamy **NetID: nxp154130**

Car Rental System:

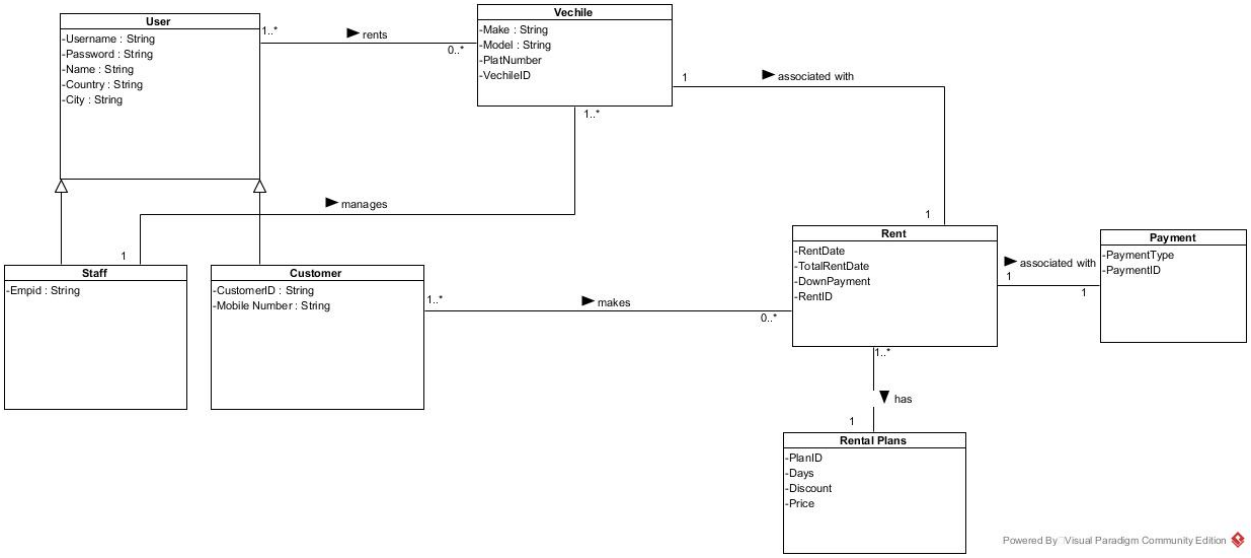
1. A) Requirements of the system:

1) System must allow customer to register to the application.
2) System must allow customer to login to the application.
3) System must allow customer to select car for reservation by selecting car make and model.
4) System must notify customer in case of the unavailability of car in particular selection.
5) In case of unavailability of particular selection system must suggest different make and model of similar price.
6) System must allow customer to cancel reservation.
7) On returning system must allow customer to pay bill by direct or by company or through credit card.
8) System must allow company to view car repairs, rental purchases and taxes.
9) On Selecting car for reservation system should allow customer to select rental plans.
10) System must allow company to register new cars.
11) System must allow company to delete car from list of available cars and also system allows company to cancel reservation.
12) System must allow company to update rental plans.

1. B) Constraints

- Every user must logged in to use the system.
- System should be able to handle many request and always available 24*7.

2) UML Class diagram as domain model:

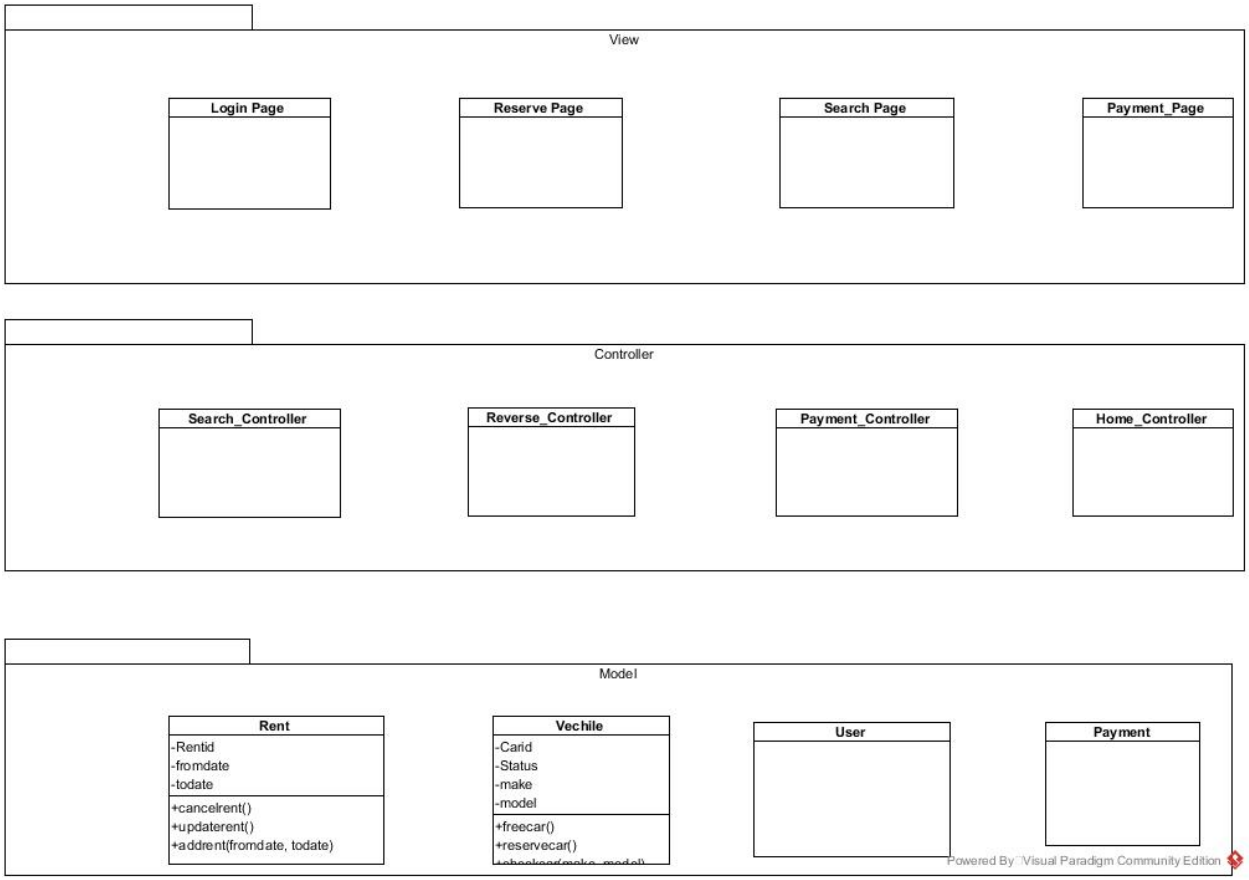


3. A) Type of the system:

From the requirements we can see that system will be of the type web application.

3. B) Architecture of the system:

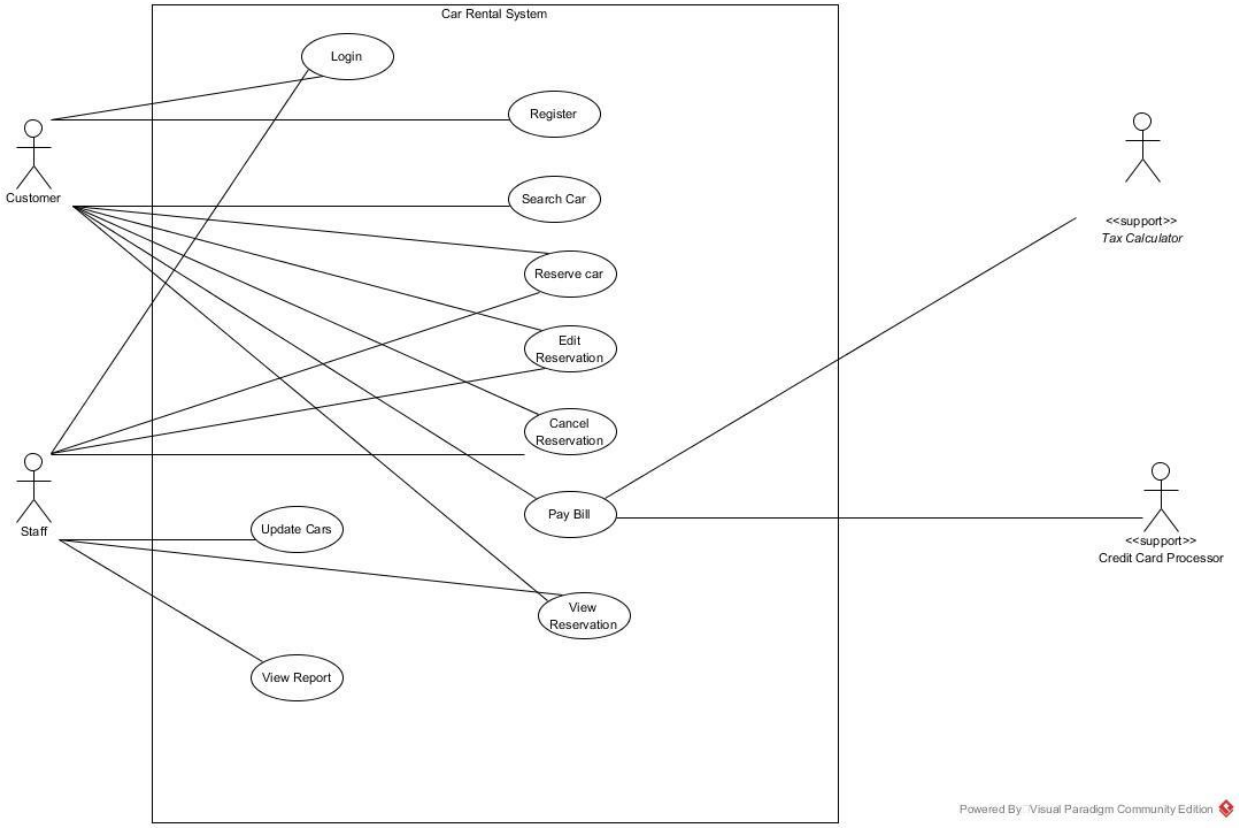
Will be using MVC architecture. Since in analysis stage came up with the rough architecture diagram. In the design stage below can see detailed design for the some use cases.



4. A) Derive use-cases from the requirements.

Login, Register, Search Car, Reserve Car, Edit Reservation, Update Reservation, View Reservation, View Report, Add vehicle, Update Rent and so on.

4. B) Use case diagram for overall system.



5. Fully Dressed Use Case text and Sequence Diagram for the below use cases.
Search Car, Reserve Car, Edit Reservation, and Candle Reservation

1) **Search Car.**

Scope: Car Rental application
Level: User goals
Primary Actor: Customer, Staff
Stakeholders and Interests:

Registered user: Can search car using make and model and reserve for a car.

General user: Can maintain the entire car rental system and see reservation and also can generate report

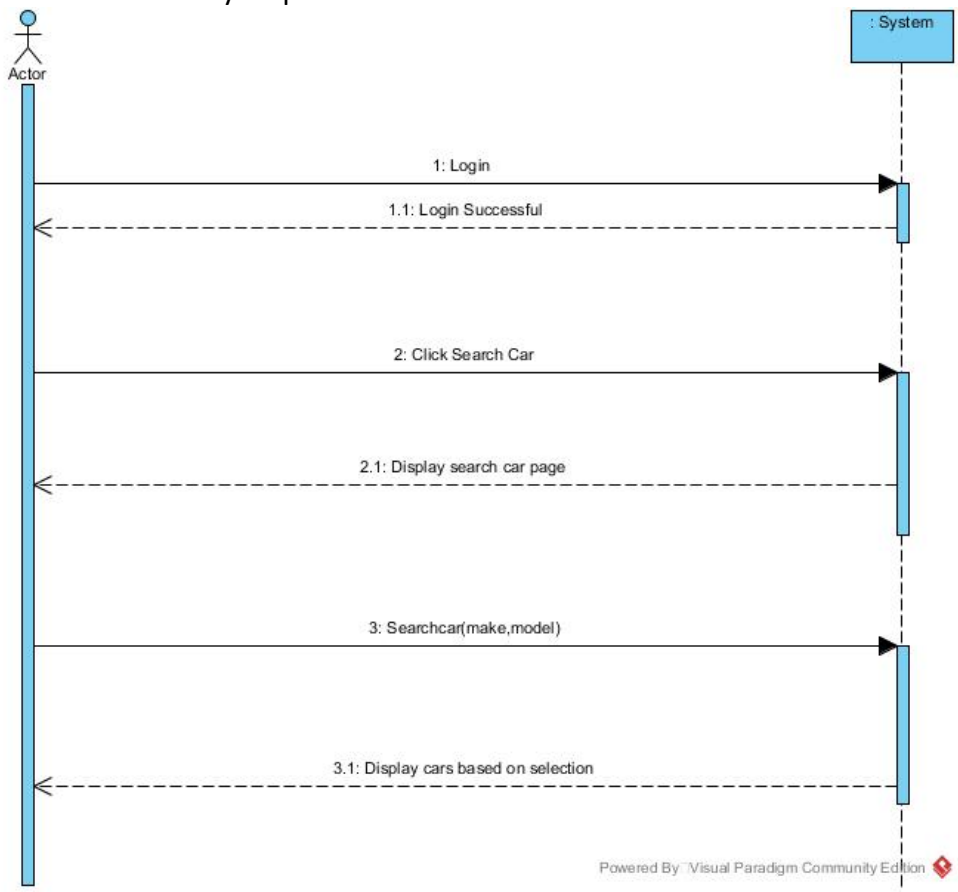
Pre-conditions: The user should be logged in.
Post-conditions: The cars are sorted and displayed based on the selected filter.
Main Success Scenario (Basic Flow):

1. The user selects the filter (by category or by date) by which cars should be sorted and displayed
2. The system displays the list of cars based on the selected filter
3. User can see list of cars based on their filter selection.

Extensions (or Alternative Flows):

*a. At any time system fails,

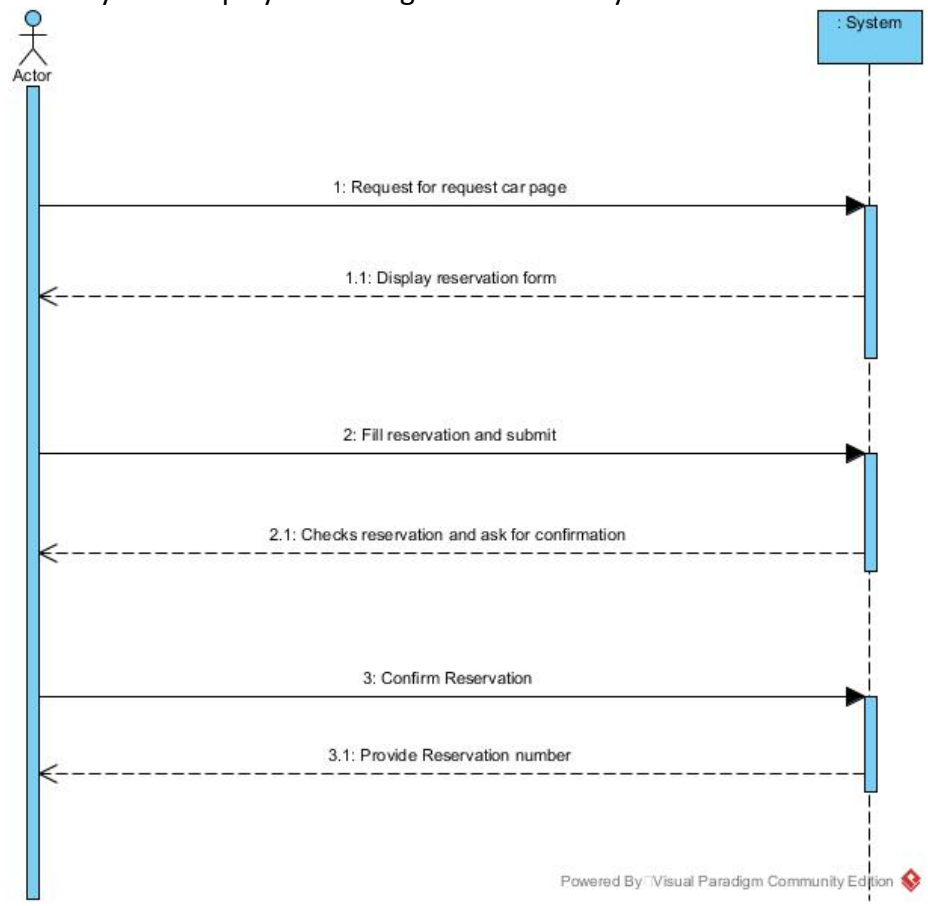
To support recovery, ensure the system's state is not changed and all the cars are recovered from any step of the scenario.



2) Reserve Car.

Scope: Car Rental application
Level: User goals
Primary Actor: Customer and Staff
Stakeholders and Interests:
Registered user: Can search car using make and model and reserve for a car.
General user: Can maintain the entire car rental system and see reservation and also can generate report
Pre-conditions: The user should be logged into the application and select a car.
Post-conditions: The selected car is reserved for the user.
Main Success Scenario (Basic Flow):
1. The user clicks on the reserve car button on the car page
2. The system displays a confirmation form for car reservation
3. The user confirms by filling his details in the form and submits it
4. The system sends an email to the user regarding the confirmation of car reservation
5. The system displays a message “Car is reserved”
6. User Receives message as Car is reserved.

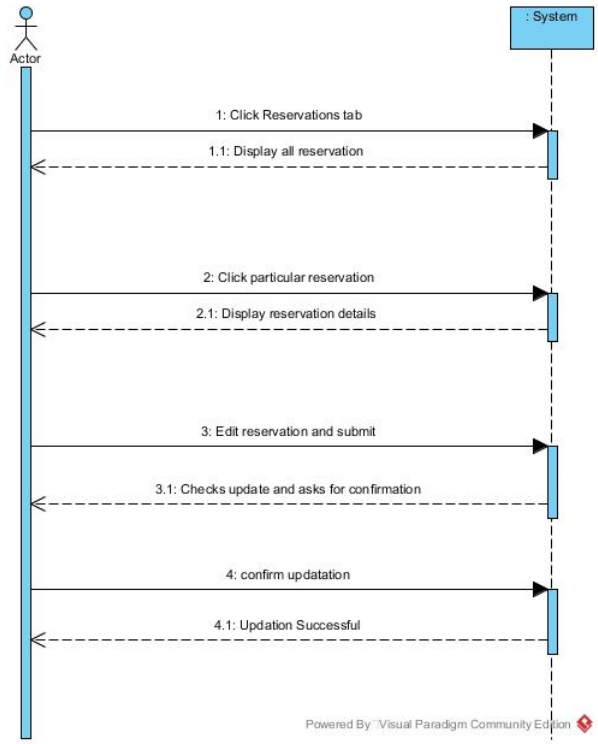
Extensions (or Alternative Flows):
2a. The system displays a message “Car is already booked”



3) Edit Reservation.

Scope: Car Rental application
Level: User goals
Primary Actor: Customer and Staff.
Stakeholders and Interests:
Registered user: Can search car using make and model and reserve for a car.
General user: Can maintain the entire car rental system and see reservation and also can generate report
Pre-conditions: The user should have reserved a car already.
Post-conditions: The reserved car is updated for the user

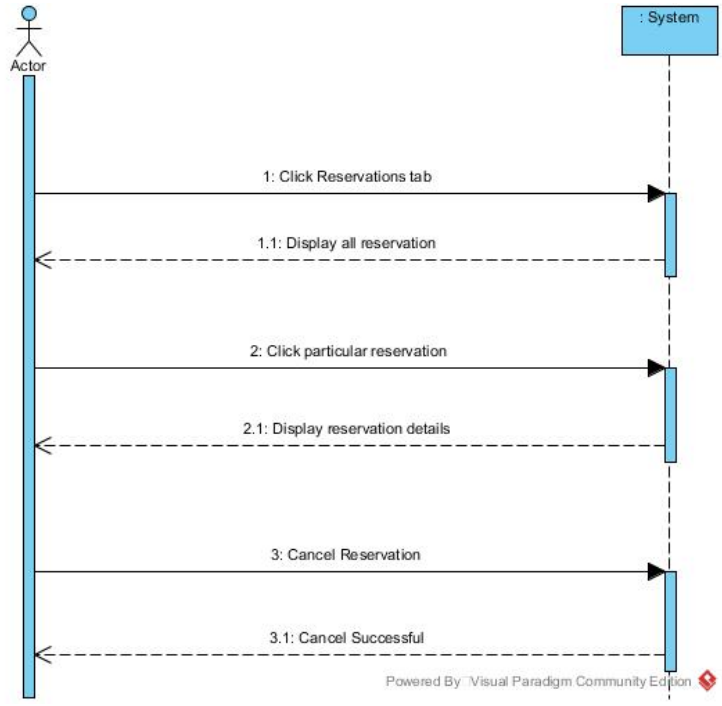
Main Success Scenario (Basic Flow):
1. The user selects the Car reservation history
2. The system displays the list of reserved cars
3. The user selects the particular car reservation to update.
4. The user updates reservation clicks on the update car reservation button
5. The system sends an email notification to the user regarding the reservation updating.
6. The system displays the message “Reserved Car is updated”
7. User receives message as Car Reservation is updated.



4) Cancel Reservation.

Scope: Car Rental application
Level: User goals
Primary Actor: Customer and Staff.
Stakeholders and Interests:
Registered user: Can search car using make and model and reserve for a car.
General user: Can maintain the entire car rental system and see reservation and also can generate report
Pre-conditions: The user should have reserved a car already.
Post-conditions: The reserved car is canceled for the user

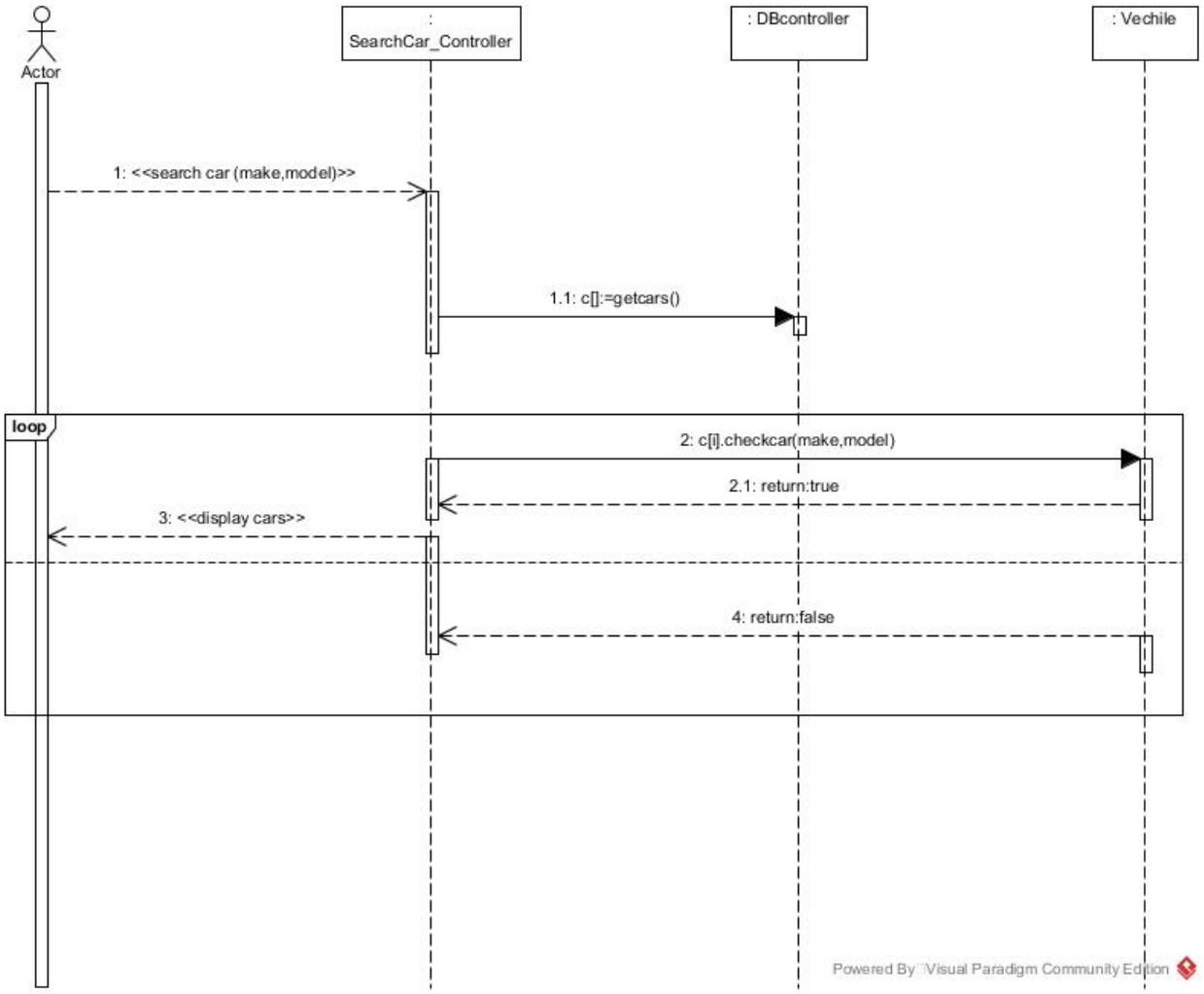
Main Success Scenario (Basic Flow):
1. The user selects the Car reservation history
2. The system displays the list of reserved cars
3. The user selects the particular car reservation to cancel.
4. The user clicks on the cancel car reservation button
5. The system sends an email notification to the user regarding the reservation cancelled.
6. The system displays the message “Reserved Car is cancelled”
7. User receives message as Car Reservation is cancelled.



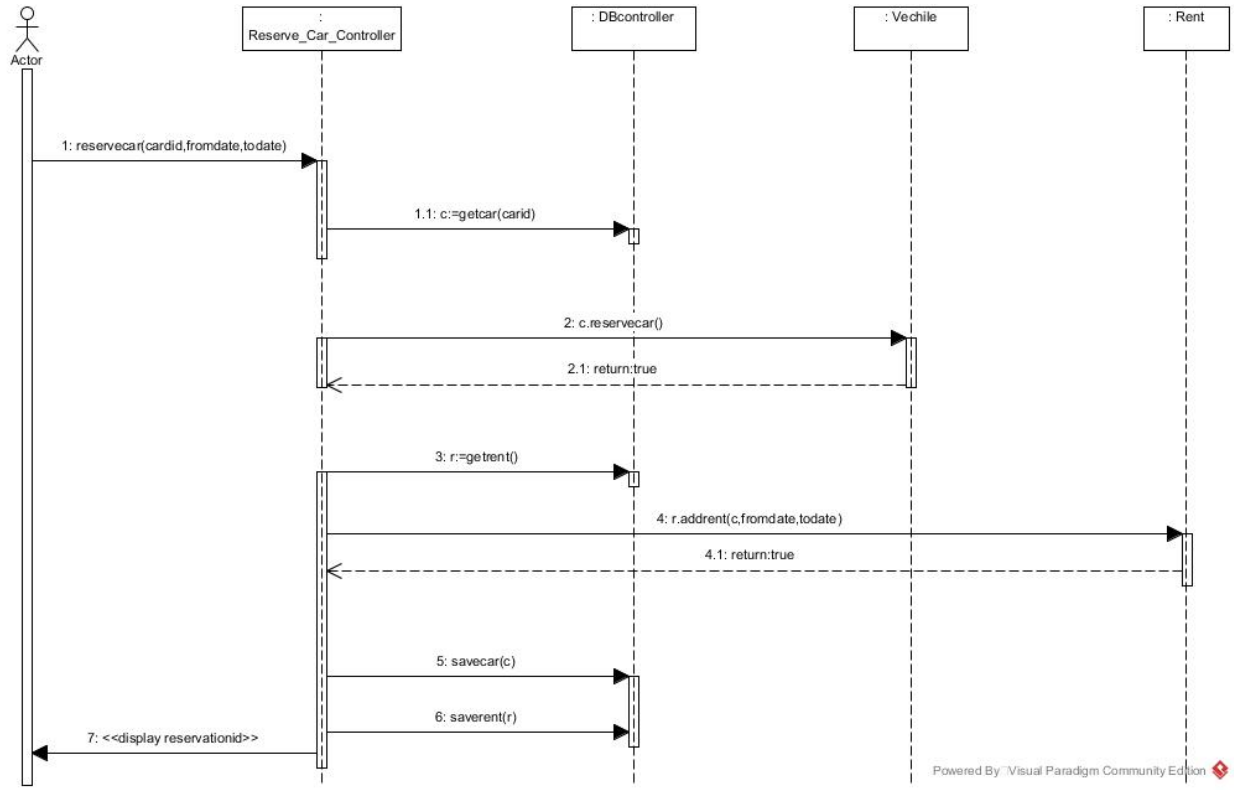
6. **Class diagram from Interaction diagrams.**

First to come up with class diagram we need interaction diagram. So we are coming up with the interaction diagram for the above addressed use case.

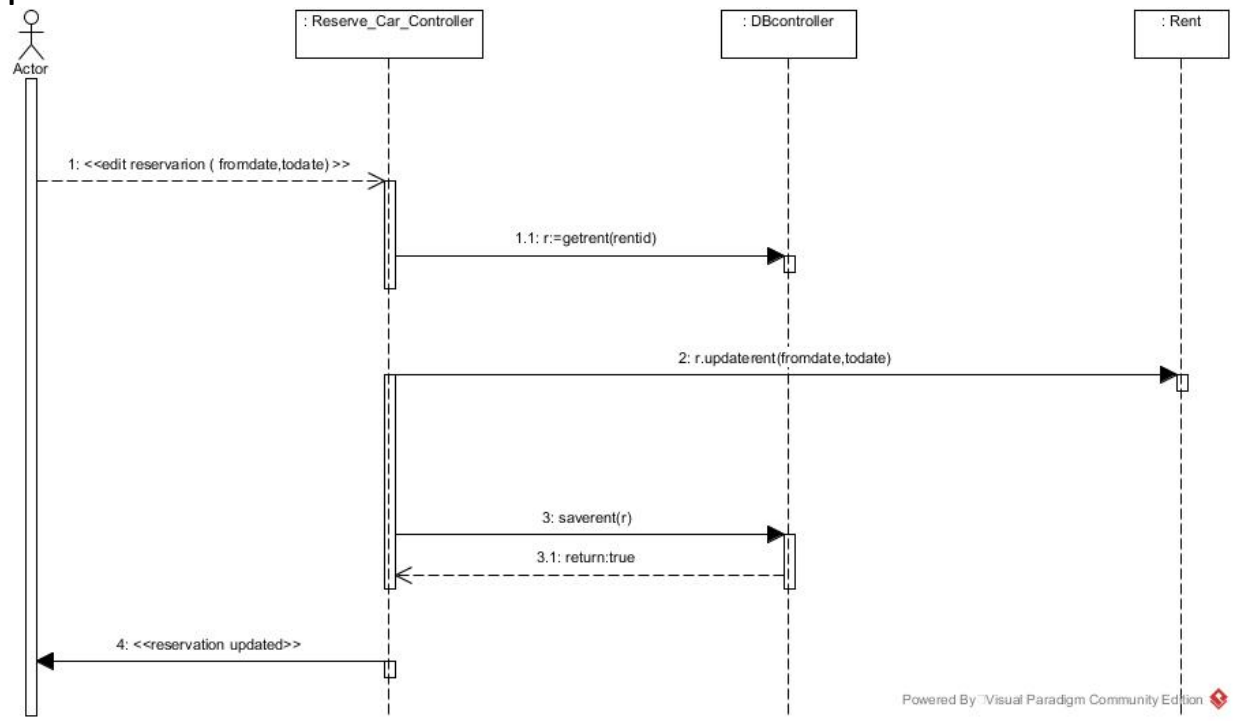
Search Car:



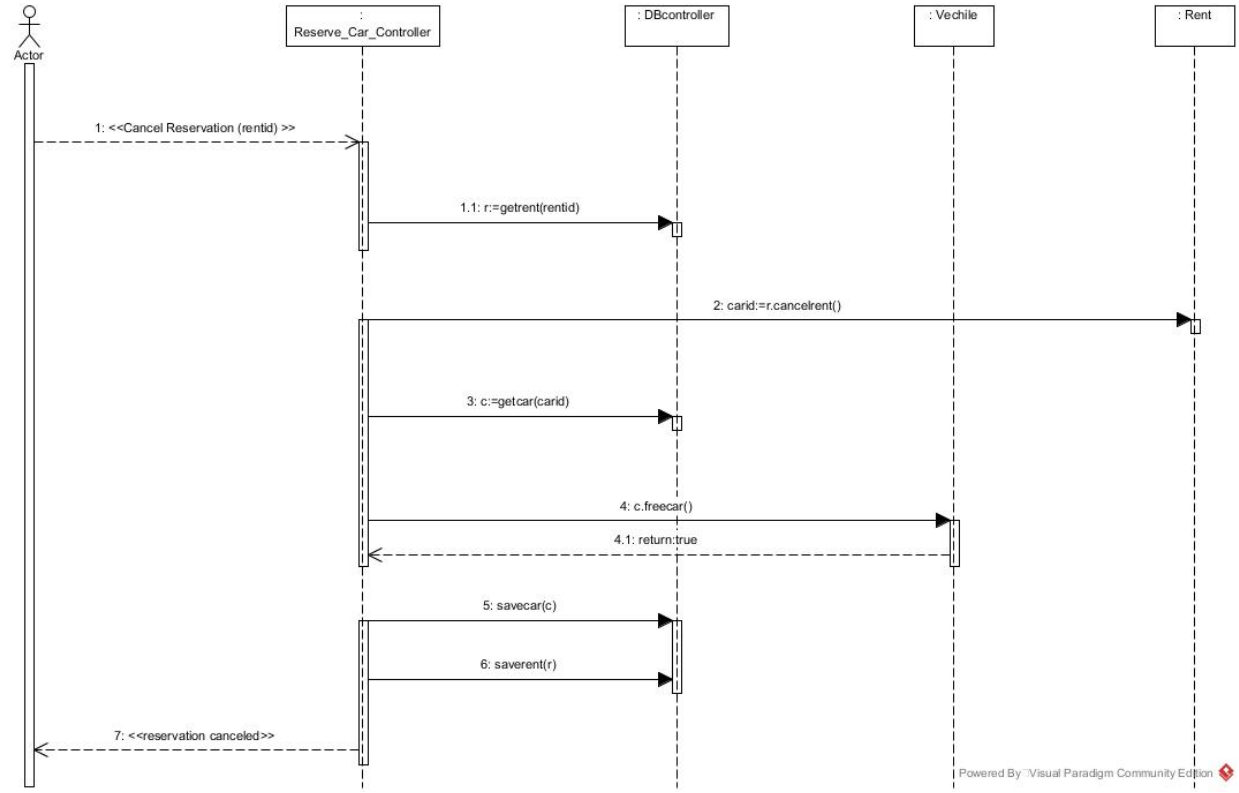
Reserve Car:



Update Reservation:

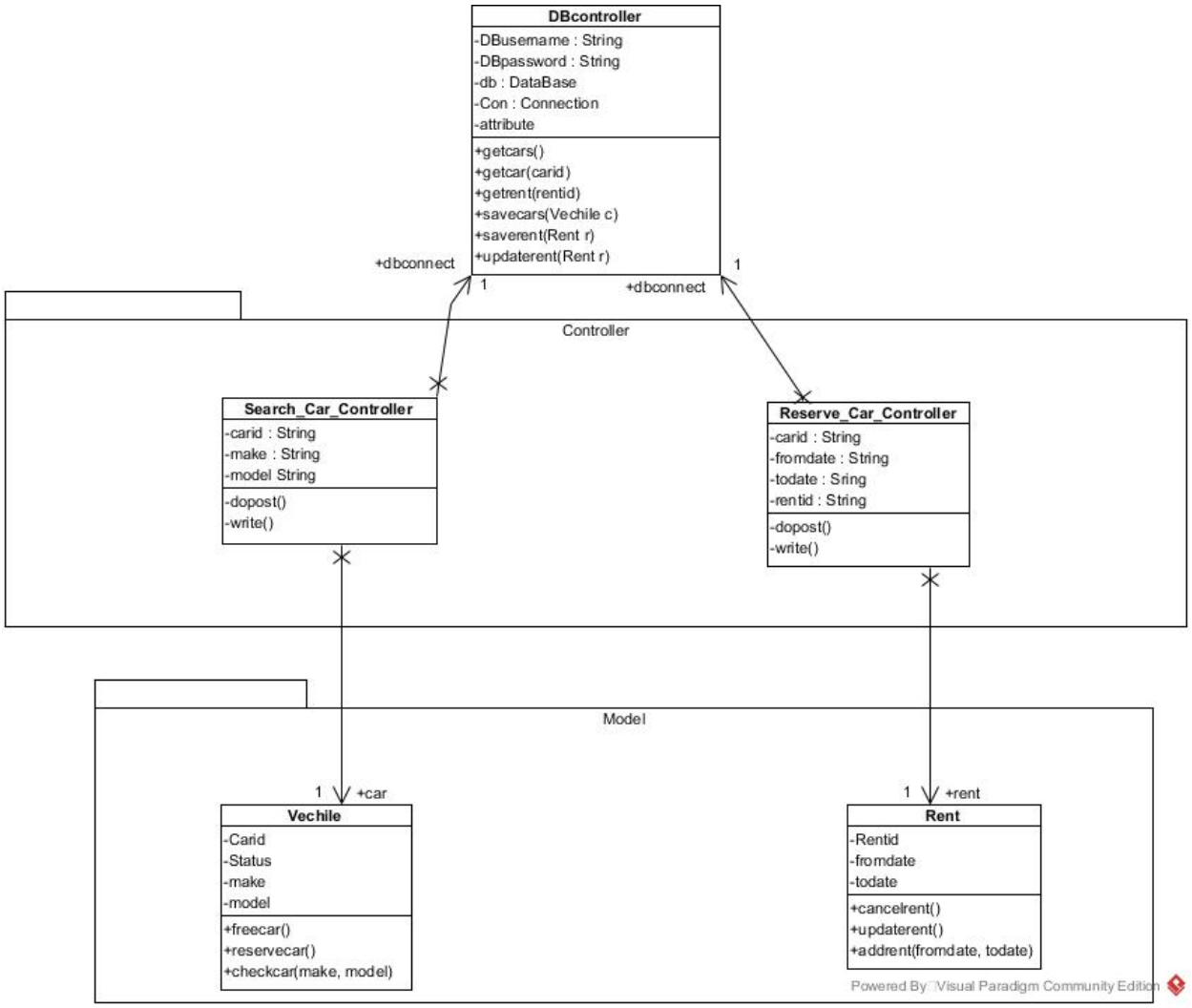


Cancel Reservation:



Class Diagram:

Class diagram is for the above mentioned use cases only so came up with only needed classes.



Patterns Applied:

1) Controller in GRASP pattern. To Avoid bolted controller, we came with 2 controller so one controller is not over loaded with jobs.

2) High Cohesion: System is highly cohesive, each class is doing its own job.

3) Expert pattern: Expert pattern can be seen my seeing Data Base controller. DB is providing you the objects and DB is updating objects.

4) Low Coupling: Coupling is maintained very low as low as possible.

5) Creator pattern: Creator pattern is applied so that each creator only handling all the response related to the objects.