HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
-KHOA CÔNG NGHỆ THÔNG TIN-

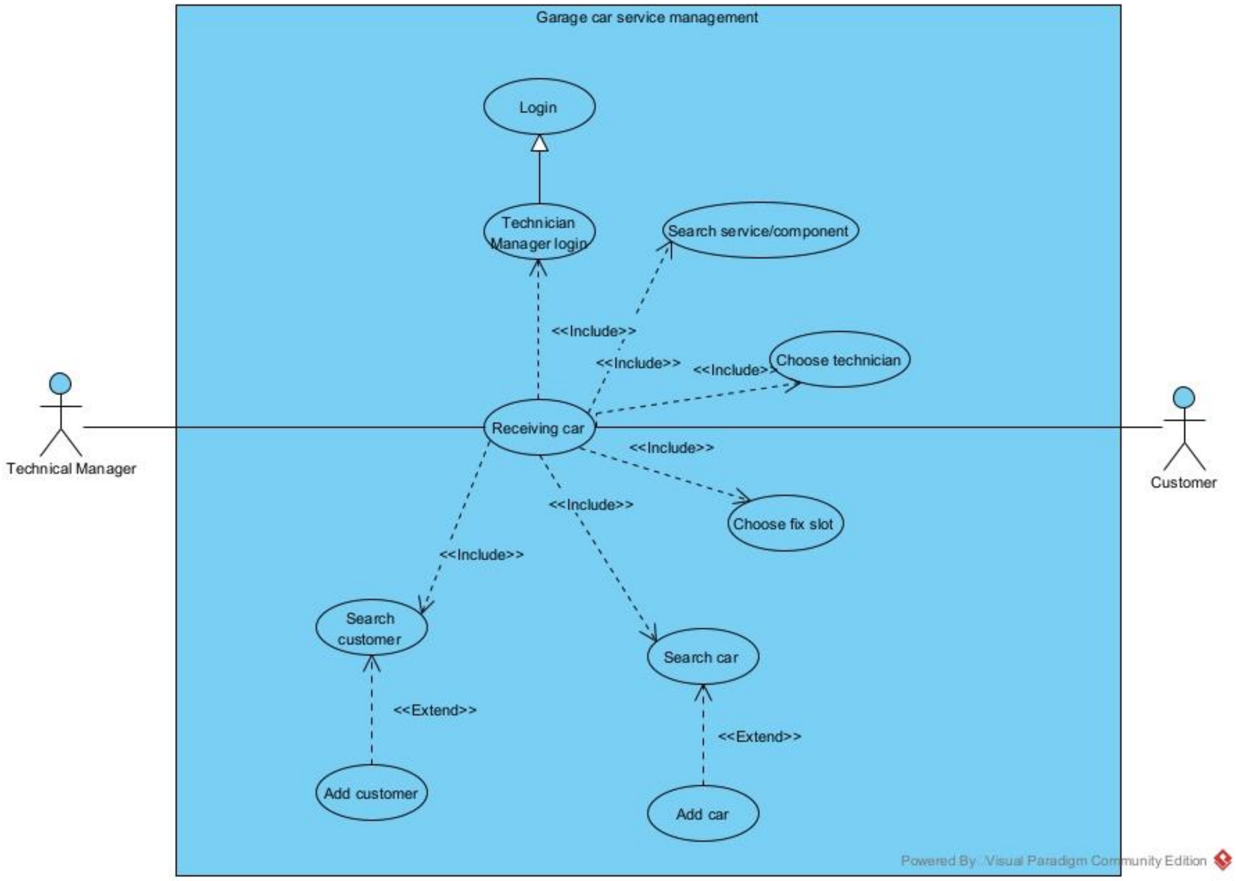
Môn học:	Nhập môn CNPM
Tên đề tài bài tập lớn:	Garage Car Service Management
Lớp:	E22CQCN03-B
Nhóm:	1
Sinh viên thực hiện:	Trần Tuấn Cường-B22DCVT073 Vũ Nam Dương-B22DCVT119 **Phạm Thế Hùng-B22DCVT229** Nguyễn Quang Vũ-B22DCVT595
Yêu cầu tuần	Báo cáo tổng hợp
Tên module	Receiving a car

Hà Nội, 2025

1.Detailed module UC diagram + description of module UCs.

Let’s review this function description: Customer brings the car → The Technical manager receives the car and does a technical check, then selects the menu to receive the car → the page to find the customer by name appears → enters the customer's name and find → the interface of the list of customers whose name contains the entered keyword appears (if not, add a new customer) → click on the correct customer → the interface of the list of cars of that customer that has been repaired appears (if not yes, then click add a new car for the customer) → the interface to enter services and spare parts appears -> repeats the following steps until the end of service/parts at the customer's request: click more services/parts event → service/component search interface appears → enter the name and search → interface list of services/components with the name containing the keyword you just entered appears → click select service/component + enter quantity (price is available at the moment) and confirmation → service/component added to customer's temporary invoice → Choose the fix slot → Choose the Technician service the Car received→
After above steps, click confirm → the system saves and prints a temporary invoice to attach car.

So, in order to receive a car , the Technician Manager have to : log in to the system, search the customer by name appears to check whether the customer exists in the system (if not, add a new customer), select a car in the list of cars of that customer that has been repaired appears(if not yes, add a new car for the customer),search service/component,then comfirm the temporary invoice (this could be separated as a sub use case or integrated in the receiving use case). So we have the use case as follow:



Use case description:

-Technician Login: this use case enables the technician manager to log in to the system .

-Search customer: this use case enables the technician manager to search the customer.

-Add customer: this use case enables the technician manager to add the new customer.

-Search car: this use case enables the technician manager to search the customer’s car.

-Add car: this use case enables the technician manager to add new car into the list of customer’s cars.

-Search service/component : this use case enables the technician manager to search service/component.

-Choose fix slot: this use case enables the technician manager to choose the fix slot for each received car.

-Choose technician : this use case enables the technician manager to choose the Technician(s) for each received car.

2. Standard module scenario

<table>
<tr><td>Scenario</td><td>Receiving a car</td></tr>
<tr><td>Actor(s)</td><td>Customer, Technician Mananger</td></tr>
<tr><td>Pre-condition</td><td>Technician Mananger has a Technician Mananger account</td></tr>
<tr><td>Post-condition</td><td>Technician Mananger received succesfully a car</td></tr>
<tr><td>Main-events</td><td>1.A Technician Manager A login with username = abc, password =12345 into the system to receive a car C from a customer B at the garage.
2.The main Technician Manager home interface appears with the option to receive a car.
3. The Technician Manger asks the customer to find the customer by name appears.
4.The client provides infor: B/HN/123456789.
5. The Technician Manager enters the customer's name and clicks on search button to find.
6. The interface of the list of customers whose name contains the entered keyword appears:
<table>
<tr><th>Code</th><th>Name</th><th>Address</th><th>Tel</th><th>Note</th></tr>
<tr><td>1</td><td>B</td><td>Ha Noi</td><td>123456</td><td></td></tr>
<tr><td>2</td><td>BC</td><td>Sai Gon</td><td>223344</td><td></td></tr>
<tr><td>3</td><td>BB</td><td>Da Nang</td><td>2342423</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>
7. A recognizes that B is already in the system on the line 1. A clicks on the line 1.
8. the interface of the list of cars of that customer that has been repaired appears
<table>
<tr><th>Code</th><th>Number plate</th><th>Model</th><th>Make</th><th>Description</th></tr>
<tr><td>1</td><td>30A-11111</td><td>Camry</td><td>Toyota</td><td></td></tr>
<tr><td>2</td><td>29B-99999</td><td>CX-5</td><td>Mazda</td><td></td></tr>
<tr><td>3</td><td>30K-56789</td><td>Corolla Cross</td><td>Toyota</td><td></td></tr>
</table>
9. A recognizes that C is already in the system on the line 3. A clicks on the line 3.</td></tr>
</table>

10. the interface to enter compont/service appears(At the bottom are the “Search " and "Next" buttons)::

Code	Name	Unit price	Quantity	To_Money

11. A reports the car condition to B and asks B to add component/service(s) as requested by customer.

12.B inform A about component/service(s).

13.To add the component/service, A clicks “search” button and search.

14. Service/component search interface appears:

Code	Name	Unit price
1	D	1000
2	DC	1234
3	DD	13
4	DCB	32

A recognizes that D is already in the system on the line 1. A clicks on the line 1.

15. A enters quantity and clicks button to comfirm.

16.The “To_Money” column will auto-fill.

17. After adding service(s),A click “next” button.

18. the interface to choose fix slot appears:

ID	Note
1	Slot 1
1	Slot 2

Choose the car receieved and click the fix slot .

19.After adding fix slot , click next.

20. The interface to choose fix slot appears:

ID	Name	Note
1	Pham The A	
2	Nguyen Quang B	

Choose the car receieved and click the fix slot .

21.After adding technician, click next.

22. The confirmation UI is appeared with information:

Invoice ID:1

TechnicianManger:A

Customer:B

Address: Ha Noi

Customer’s phone number:123456

Customer’s car:

1.

Number plate :30K-56789

Model : Corolla Cross

Made : Toyota.

Code	Name	Unit price	Quantity	ToMoney
TotalAmount				

Information about component/service(s):Name is D ,Unit price 1000, Total amount of the component.

A confirm and a cancel button.

23.A repeats the infor to the B and asks to confirm

24.B agrees with the infor + confirms

25.A clicks on the confirm button

26. The system displays the receiving success alert and returns to the main seller UI.

27. A informs to B that the booking is success

Exceptional scenarios:

	6. There is no customer found or the customer B is not in the system yet. 6.1. A clicks “Add new customer” button. 6.2. System returns the interface similar to step 8 in the main-events. 8. There is no available car found 8.1.A clicks “ Add new car” button. 8.2 System returns the interface similar to step 10 in the main-events. 18.There is no free fix slot. 18.1.the system will auto add fix slot from slot 1. 20.There is no free technician. 20.1.the system will auto add technicina from technician 1.

3. Entity class (Analysis phase).

Step 1: describe the system within a paragraph**.**

The system enables the Technician Manager to receive cars from customers, checking the car's condition, selecting repair services and parts,fixslots and technician. Once a receving is completed , a temporary invoice is created by the system with the information about the customer, the received car, and the service that the customer used for the car.

Step 2+3: extract nouns and classify them.

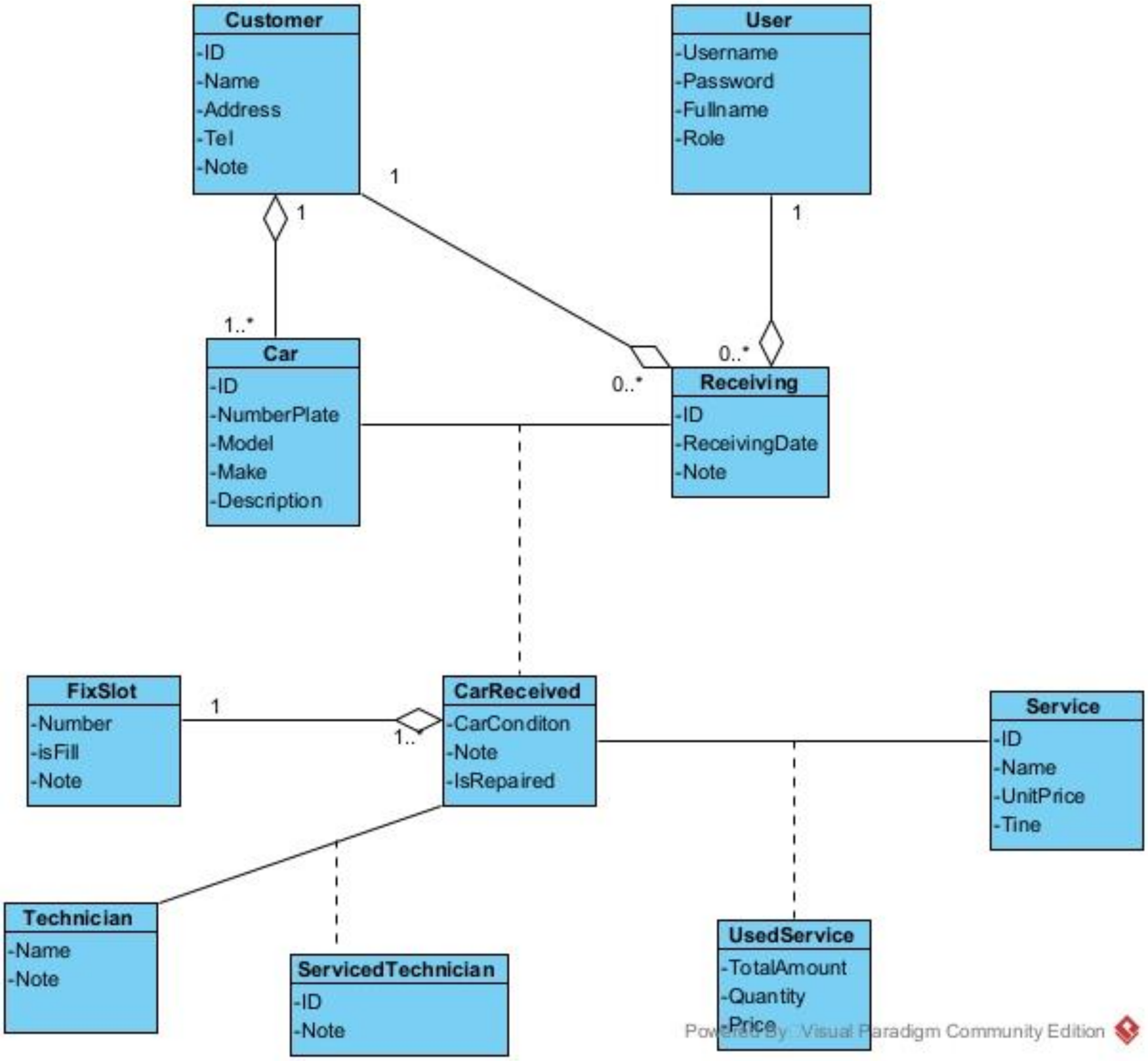
- System: an abstract noun →reject.
- Information: a common noun→reject.
- Technician Manager : a kind of class User
- Customer: need to be managed → a class: Customer
- Car: need to be managed→ a class: Car
- Services/Parts: need to be managed→ a class: Service
- Username,Password,Position,FullName : attributes of User
- Code,Name,Address,Tel.Note: attributes of Customer

- FixSlot: need to be managed→ a class: FixSlot
- Techncian: need to be managed→ a class: Technician

Step 4+5: quantity and object relationships among classes

The relationships among the classes are defined like this:

-A customer can have many cars, and a car belongs only to one customer. So, Customer – Car is 1 – n.
-A Technician Manager(User) may receives many cars, a car may be recevied by many Technician Managers(Users) at different times: so User-Car is n-n .So we could propose a class between them: Receiving.
-A customer may have many Receivings, but a receiving is created for only one Customer . So the relationship between Customer-Receiving is 1-n.
-A Car may also be received in many Receiving (in different times). Moreover, in a Receiving, the TechnicianManger may receive many cars (for only one Customer). So, Receiving – Car is still n-n. We need a more class between them: CarReceived. A Receiving and Car determine an unique CarReceived. This association relationship determines also some information: CarStatus.
-A Service may have been used many time for a CarReceived (in different times). A ReceivedCar may also use many services. The relationship ReceivedCar – Service is n-n. So we create a class UsedService between ReceivedCar and Service.
-A ReceivedCar is serviced in many FixSlots(in many times). So the relationship ReceivedCar-FixSlot is n-1.
-A ReceivedCar may be serviced by many Technician.
A Technician may service many ReceiviedCar in different times.So ReceivedCar-Technicians : n-n. So we create a class ServicedTechnician between ReceivedCar and Technician.



4.Class Diagram(Analysis phase).

Receiving a car

- Enter the system -> The login interface is appeared -> need a class: LoginView
  - input for username -> inUsername
  - input for password -> inPassword
  - a submit to login -> subLogin
- Enter the username/password -> the system must check if the login is correct -> need a method:

  - name: checkLogin()
  - input: username, password (of the class User)
  - output: boolean
  - assign to the entity class: User.
- Once login is successful -> the main interface of the Technician Manager is appeared -> need a class: TechnicianManagerHomeView which has at least:
  - an option to choose to receive car -> subReceiving.
- When a customer bring a car to receive -> the Technician Manager chooses the receiving option -> the interface to search customer is appeared -> need a class: SearchCustomerView
  - input name to search -> inName
  - input code to search -> inCode
  - input address to search -> inAddress
  - input tel to search -> inTel
  - input note to search -> inNote
  - a button search -> subSearch
  - a button to add new customer (in the case of new customer)-> subAddCustomer
  - a list of results -> outsubListCustomer
- Enter the Customer name to search -> The system has to search all customers whose name contains the entered keyword -> need a method:
  - name: searchCustomer()
  - input: a keyword
  - output: a list of customer
  - assign this method to the entity class: Customer.
- The results are returned to (and displayed on) the SearchCustomerView.
- The Technician Manager chooses the correct customer (in the case of new customer, chooses to add new customer and the addCustomerView appeared, that needs a method to add new customer into the DB:
  - name: addCustomer()
  - input: Customer
  - output: boolean
  - assign this method to the entity class: Customer.

).

- The interface to search car is appeared -> need a class: SearchCarView
    - input numberPlate to search -> inNumberPlate
    - input code to search -> inCode
    - input model to search -> inModel
    - input make to search -> inMake
    - input description to search -> inDescription
    - a button search -> subSearch
    - a button to add new car (in the case of new car)-> subAddCar
    - a list of results -> outsubListCar
- Enter the Car name to search -> The system has to search all cars whose name contains the entered keyword -> need a method:
    - name: searchCar()
    - input: a keyword
    - output: a list of car
    - assign this method to the entity class: Car.
- The Technician Manager chooses the correct car (in the case of new customer, chooses to add new cuar and the addCarView appeared, that needs a method to add new car into the DB:
    - name: addCar()
    - input: Car
    - output: boolean
    - assign this method to the entity class: Car.

)

- The interface to search service is appeared -> need a class: SearchServiceView
    - input code to search -> inCode
    - input name to search -> inName
    - input Unit price to search -> inUnitPrice
    - a button search -> subSearch

    - a button to comfirm after adding service(s) -> subNext
    - a list of results -> outsubListService

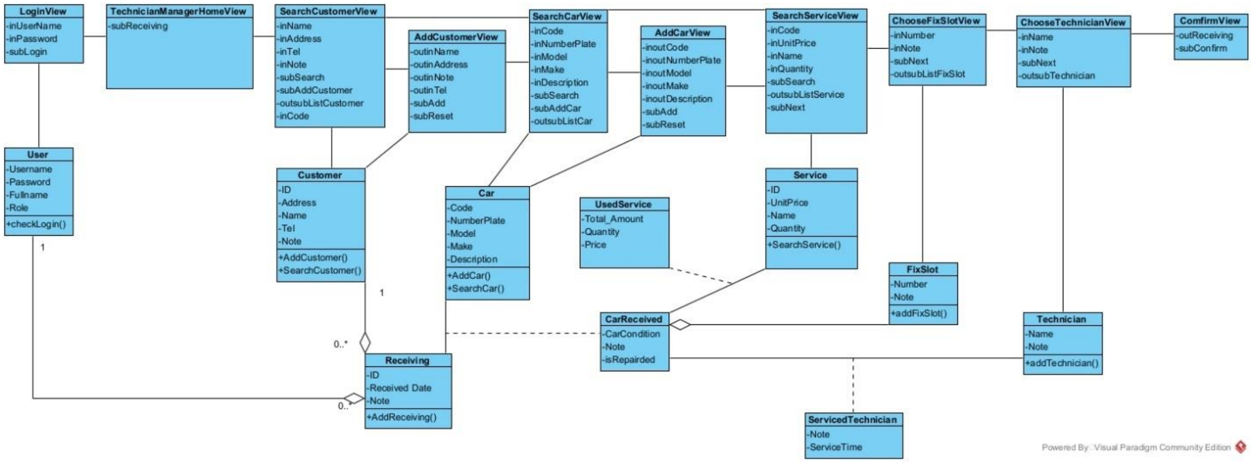
- The interface to choose FixSlot is appeared -> need a class: ChooseFixSlotView

    - input number to search -> inNumber
    - input Note to search -> inNote
    - a button to comfirm after adding fixslot(s)-> subNext
    - a list of results -> outsubListFixSlot

- The interface to choose Technican is appeared -> need a class: ChooseTechnicianView

    - input name to search -> inName
    - input Note to search -> inNote
    - a button to comfirm after adding technician(s)-> subNext
    - a list of results -> outsubListTechnician

- The confirm interface is appeared -> need a class ConfirmView
    - display all information about the Receiving -> outReceiving
    - a confirm button -> subConfirm.
- The Technicina Manager chooses to confirm after having the aggregation from the customer -> The system has to save the receiving into the DB -> need a method:
    - name: addReceiving()
    - input: an object of Receiving
    - output: none or boolean
    - -> assign this method to the entity class: Receiving.
- After saving to the DB, the system returns to the TechnicianManagerHomeView.



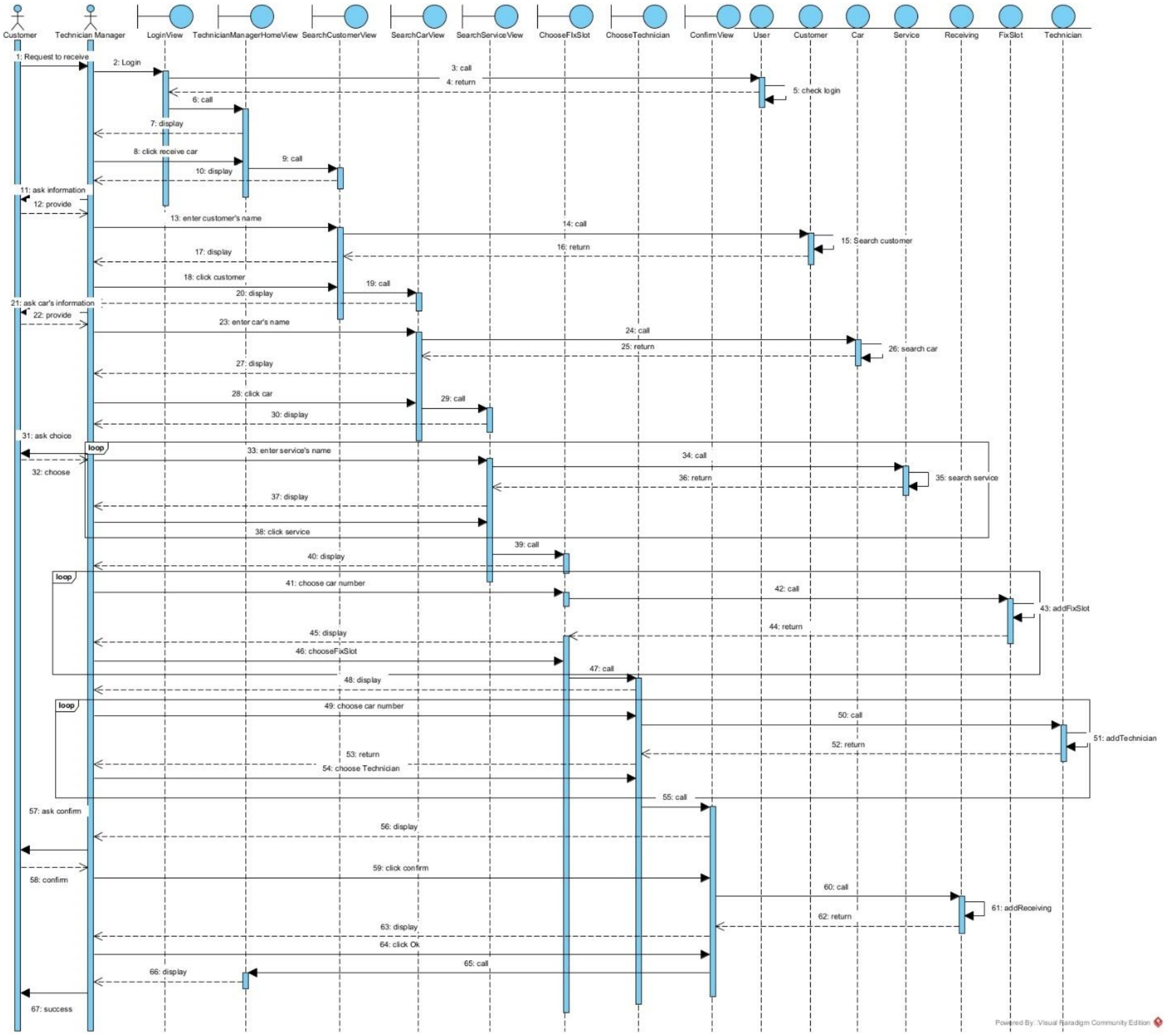
5. Sequence diagram(Analysis phase).

Scenario version 2

1. A customer brings a car to the garage.
2. The Technician Manager enters username/password and then clicks on the Login button.
3. The class LoginView calls the class User to process.
4. The class User calls the method checkLogin(). The login is successful.
5. The class User returns the results to the class LoginView.
6. The class LoginView calls the class TechnicianManagerHomeView.
7. The class TechnicianManagerHomeView displays itself to the Technician Manager.
8. The Technician Manager chooses the option to receive car on the interface TechnicianManagerHomeView.
9. The class TechnicianManagerHomeView calls the class SearchCustomerView.
10. The class SearchCustomerView displays itself to the Technician Manager.
11. The Technician Manager asks personal information about the Customer.
12. The Customer provides them to Technician Manager .
13. The Technician Manager enters the customer’s name and clicks on the search button.
14. The class SearchCustomerView calls the class Customer to process.
15. The class Customer calls the method searchCustomer().
16. The class Customer returns the results to the class SearchCustomerView.
17. The class SearchCustomerView displays the results to the Technician Manager.

18. The Technician Manager chooses the row that corresponds to the current Customer.
19. The class SearchCustomerView calls the class SearchCarView.
20. The class SearchCarView displays itself to the Technician Manager.
21. The Technician Manager asks informations about the Car.
22. The Customer provides them to Technician Manager .
23. The Technician Manager enters the car’s name and clicks on the search button.
24. The class SearchCarView calls the class Car to process.
25. The class Car calls the method searchCar().
26. The class Car returns the results to the class SearchCarView.
27. The class SearchCarView displays the results to the Technician Manager.
28. The Technician Manager chooses the row that corresponds to the current Car.
29. The class SearchCarView calls the class SearchServiceView.
30. The class SearchServiceView displays itself to the Technician Manager.
31. The seller informs the car’s status to the Customer and requires the customer to choose Service(s).
32. The Customer provides the service(s)’s name to Technician Manager.
33. The Technician Manager enters the service’s name and clicks on the search button.
34. The class SearchServiceView calls the class Service to process.
35. The class Service calls the method searchService().
36. The class Service returns the results to the class SearchServiceView.
37. The class SearchServiceView displays the results to the Technician Manager.
38. The Technician Manager chooses the service(s) which satisfies the customer’s requirement and click next button. (Repeat steps 33 to 38 until the required number of service is added.)
39. The class SearchserviceView calls the class ChooseFixSlotView.
40. The class ChooseFixSlotView displays itself to the Technician Manager.
41. The Technician choose car’s number.
42. The class ChooseFixSlotView calls the class FixSlot to process.
43. The class FixSlot calls the method addFixSlot().
44. The class FixSlot returns the results to the class ChooseFixSlotView
45. The class ChooseFixSlotView displays the results to the Technician Manager.
46. The Technician Manager chooses the fixslot for each car received and click next button. (Repeat steps 41 to 46 until the required number of fixslot is added.)

47. The class ChooseFixSlotView calls the class ChooseTechnicianView.
48. The class ChooseTechnicianView displays itself to the Technician Manager.
49. The Technician choose car’s number.
50. The class ChooseTechnicianView calls the class Technician to process.
51. The class Technician calls the method addTechnician().
52. The class Technician returns the results to the class ChooseTechnicianView
53. The class ChooseTechnicianView displays the results to the Technician Manager.
54. The Technician Manager chooses the technician(s) for each car received and click next button. (Repeat steps 49 to 54 until the required number of technician is added.
55. The class ChooseFixSlot calls the class ConfirmView.
56. The class ConfirmView displays all receiving information to the Technician Manager.
57. The Technician Manager repeats these informations to the customer and requires the customer to confirm.
58. The customer confirms them.
59. The Technician Manager clicks of the confirm button.
60. The class ConfirmView call the class Receiving to process.
61. The class Receiving calls the method addReceiving().
62. The class Receiving returns to the class ConfirmView.
63. The class ConfirmView displays a successful message to the Technician Manger.
64. The Technician Manager clicks on the OK button of the message.
65. The class ConfirmView calls the class TechnicianMangerHomeView.
66. The class TechnicianMangerHomeView displays itself to the Technician Manger.
67. The Technician Manager informs the successful receiving to the customer and gives customer the temporary invoice’s code.



6.Entity Classes(Design phase).

- *Step 1*: Complete class and attributes:

  -Rename class if needed

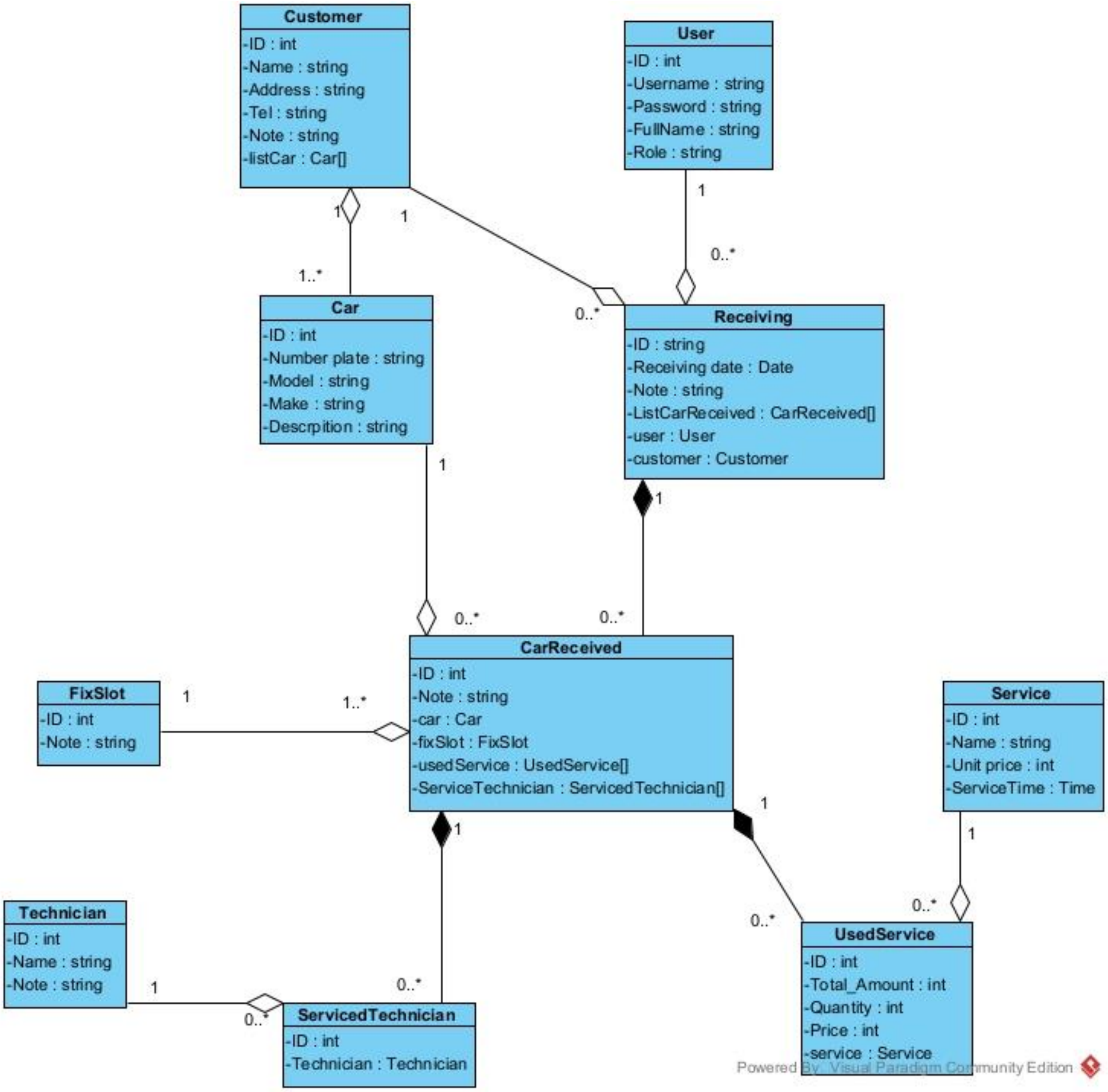
  - Rename attribute if needed

  -Add the id attribute for the classes which DO NOT inherit from other class: Customer, Receving,CarReceived, User, Service, UsedService,Temporary Invoice,Car,Technician,FixSlot.

- *Step 2*: Add the type of each attribute in all classes

- *Step 3*: Convert all association relationships to correspond aggregation/composition relationships:
    - Car + Receiving -> CarReceived is converted to: Car is a component of CarReceived, CarReceived is a component of Receiving.
    - CarReceived + Service -> UsedService is converted to: Service is a component of UsedService, UsedService is a component of CarReceived.
- *Step 4*: Add the object attributes which correspond to the aggregation/composition relationships:
    - Car is a component of Customer, of type n-1 -> Customer has a list of Car.
    - Car is a component of CarReceived, of type 1-n -> CarReceived has a Car
    - CarReceived is a component of Receiving, of type n-1 -> Receiving has a list of CarReceived
    - Customer is a component of Receiving, of type 1-n -> Receiving has a Customer
    - User is a component of Receiving, of type 1-n -> Receiving has an User
    - Service is a component of UsedService, of type 1-n -> UsedService has a Service
    - UsedService is a component of CarReceived, of type n-1 -> CarReceived has a list of UsedService
    - FixSlot is a component of CarReceived, of type 1-n -> CarReceived has a FixSlot
    - Technician is a component of ServiceTechnician, of type 1-n -> ServiceTechnician has a Technician
    - ServiceTechnician is a component of CarReceived, of type n-1 -> CarReceived has a list of ServiceTechnician

(Service include Service and part)



7. Database design.

- Step 1: For each entity class, create a corresponding table. The table name should have the prefix “tbl” (table) + the name of the entity class. For example, from an entity class Customer, the corresponding table should be tblCustomer.Therefore, we have tables:

tblCustomer,tblCar,tblUser,tblReceiving,tblFixSlot,tblTechnician,tblCarReceived,tblFixSlot,tblTechnician,tblServicedTechnician,tblService,tblUserService.

- Step 2: For each entity class, transfer all NON-OBJECT attributes to contribute as the columns of the corresponding table. For example, in the class Customer, the attribute listCar has a type of Car[] is a OBJECT attribute, so it will be removed, the remain attributes (id, name, address, tel,Note) will play the columns of the tblCustomer.
- Step 3: Quantitative relationship among classes -> quantitative relationship among
corresponding :

  -tables 1-1: Regroup/Keep

  -1-n: Keep

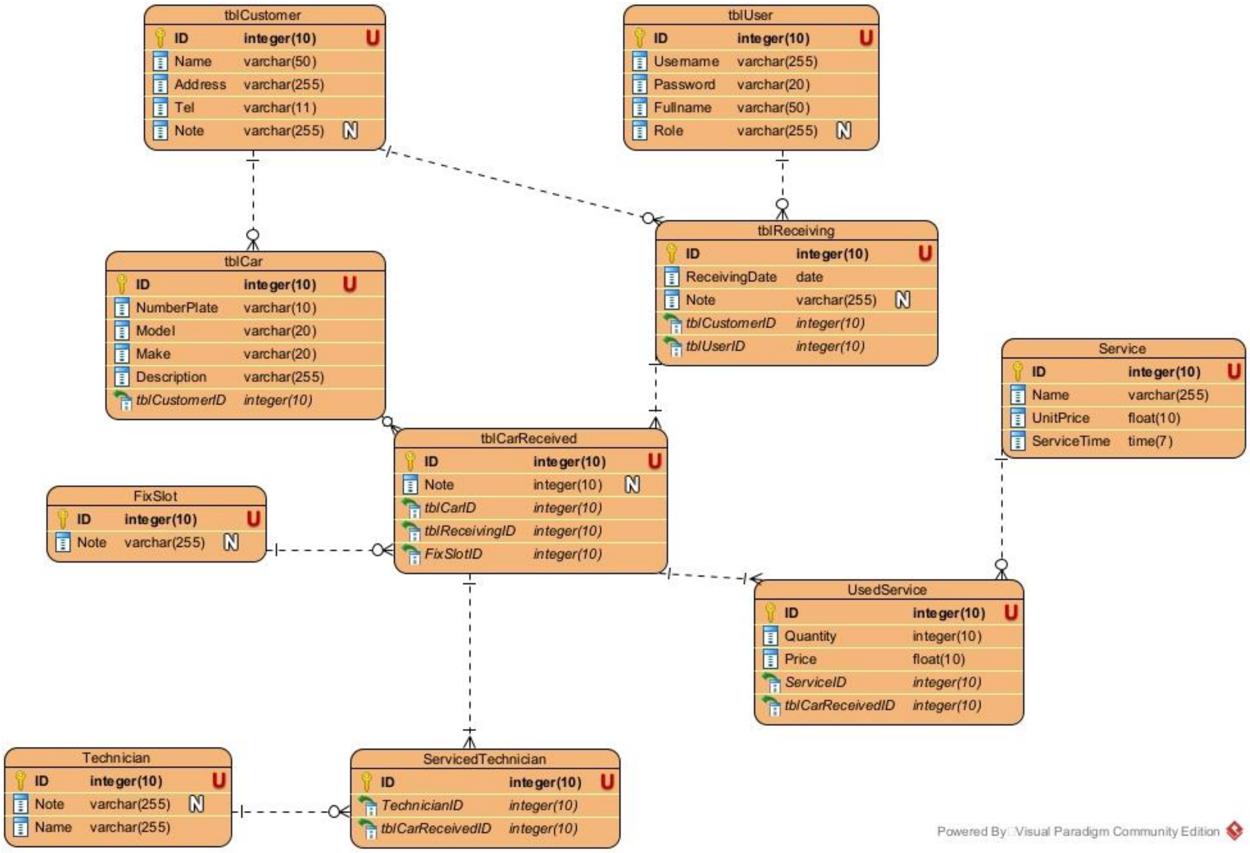
  -n-n: Incorrect -> Return to analyze phase to correct

- Step 4: Add key attribute into tables:

  Primary key: If table has an id -> set it as PK

  Foreign key: If tblA - tblB (1-n) =>tblB has to have a foreign key refering to the PK of tblA

Step 5: Remove redundant attributes: Duplicate, Secondary



8.Class Diagram (Design phase).

In this module, the login processing is omitted:

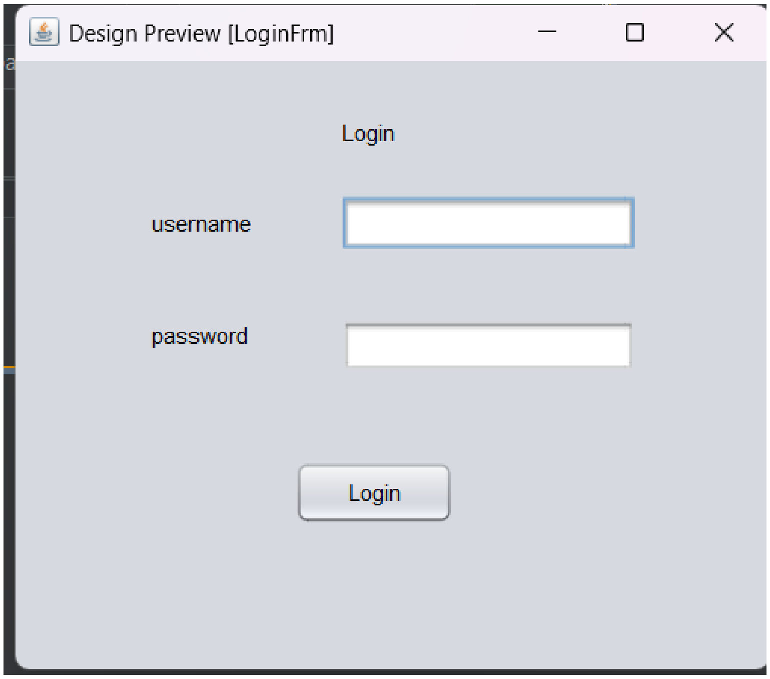
- View classes:
    - LoginFrm is the interface to login. It needs a text field to enter the username, a text field to enter password, and a button to login.
    - TechnicianManagerHomeFrm is the home interface of the TechnicianManager. It needs at least a button to go to the receiving function, a label to display the user’s fullname
    - SearchCustomerFrm is the interface to search and select customer. It needs 4 text fields to enter the customer’name ,address ,tel and note, a search button and a table to show the results and button add new customer to add new customer with informations in 4 text fields if the customer does not exist in the DB.

  - SearchCarFrm is the interface to search and select the customer’s car(s). It needs 1 text field to enter the Customer’ID ,a button search,a table to show the list of customer’cars, a button to add new car if the receiving car does not exist in the DB and a button next when select all the car.
  - AddCarFrm is the interface to add a new car . It needs 4 text field to enter the numberplate,model,make,Description ,a button to add .
  - SearchServiceFrm is the interface to search and select the service. It needs a comboBox to choose car,2 text fields to enter the service’s name and to choose the quantity,a button to search, a table to show the list of services a button next ,a button to add service .
  - ChooseFixSlot is s the interface to search and select the free fix slot. It needs a combobox to choose the car to add fix slot ,a button to search, a table to show the list of fix slots , a button next .
  - ChooseTechnician is s the interface to search and select the free technician . It needs 1 text fields to enter carID to choose the car to add technician ,a button to search, a table to show the list of fix slots, a button next .
  - ConfirmFrm is the interface to confirm the receiving information.It need 2 Jlabel to inform Customer’s name and fix slot,2 Jtable to present UsedService(s) and ServicedTechnician(s),a button confirm , a button cancel.
- Control (DAO) classes:
  - UserDAO is the class for manipulating with DB related to the User object. In this module, it needs a method to verify whether the login information is correct or not, it is checkLogin() method.
  - CustomerDAO has two methods:
    - searchCustomer(): to search customer whose name contains the entered key
    - addCustomer(): to add new customer into the DB
  - CarDAO has two methods:
    - searchCar(): to search by the customer’s id
    - addCar(): to add new car into the DB
  - ReceivingDAO has a method to add a new receiving addReceving().

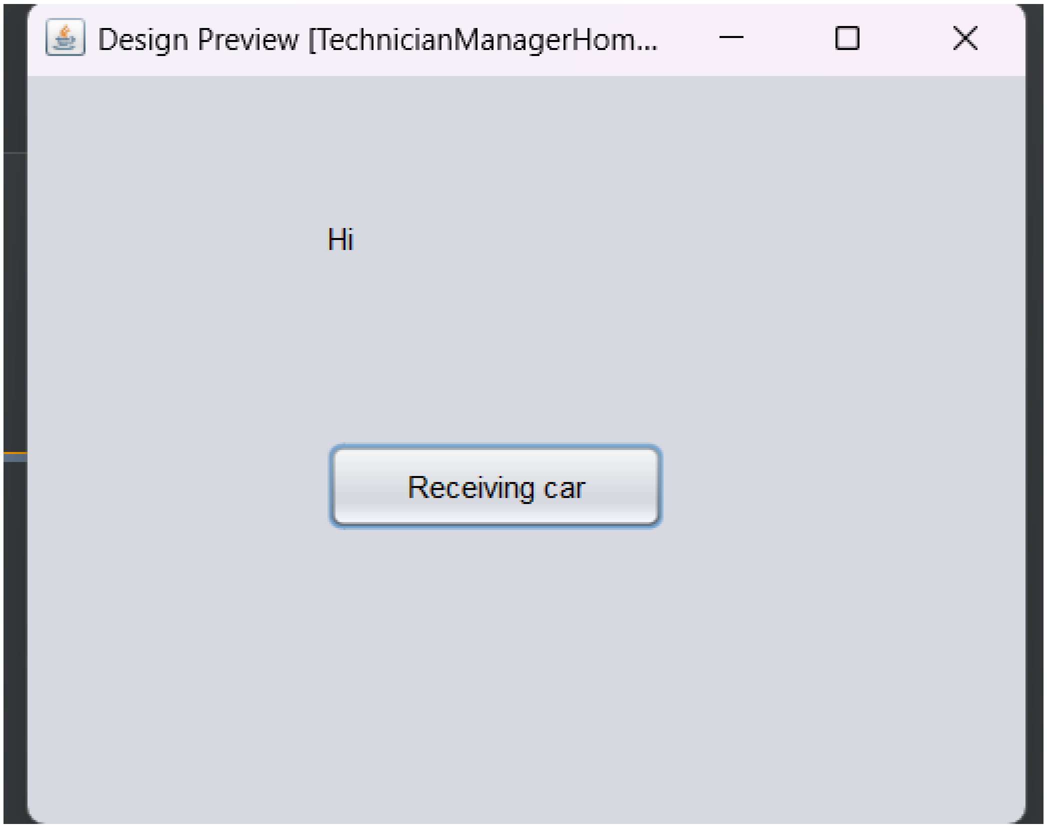
    - ServiceDAO has a method to search available service searchService().
    - FixSlotDAO has a method to search available fix slot searchFreeFixSlot().
    - TechnicinaDAO has a method to search available fix slot searchFreeTechnician().
- Entity classes: Car, Customer,Receiving,CarReceived ,UsedService, Service, FixSlot,Technician and User,ServiceTechnician.

Design interface:

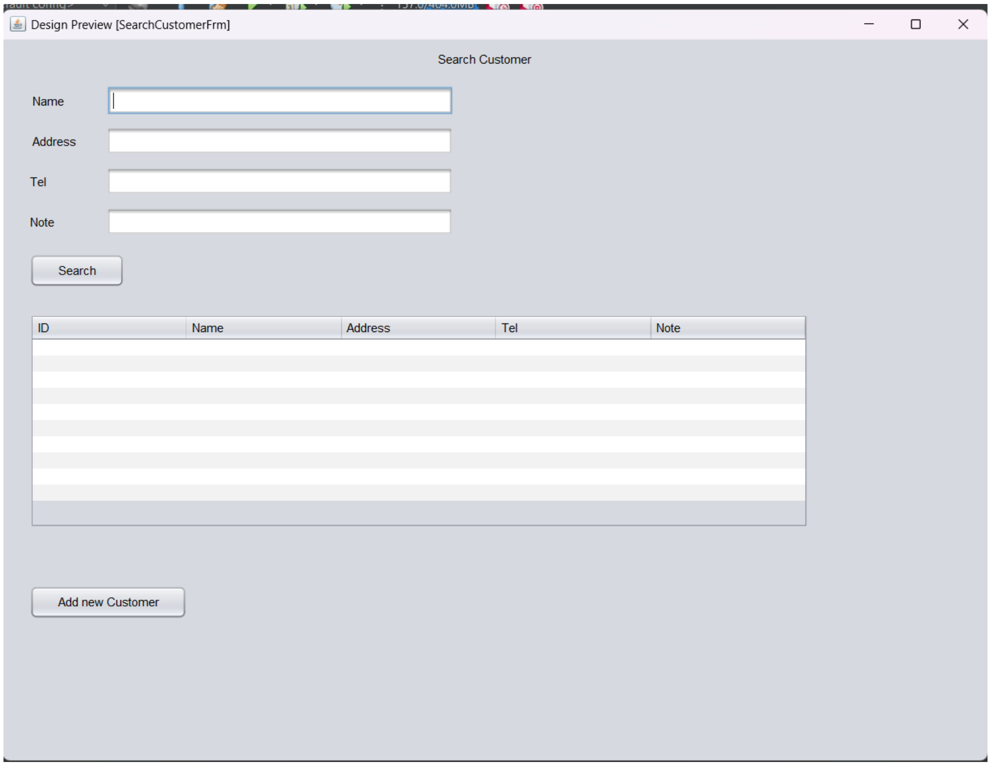
-LoginFrm:



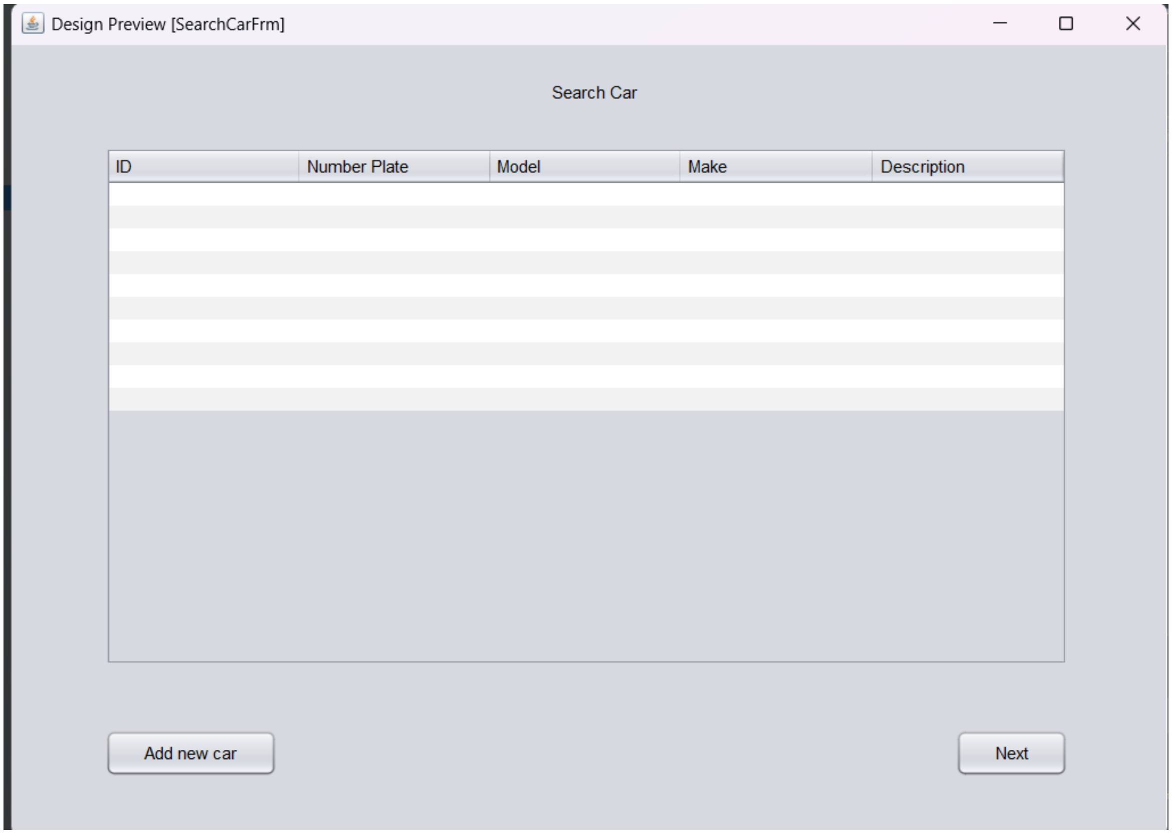
-TechnicianManagerHomeFrm:



-SearchCustomerFrm:

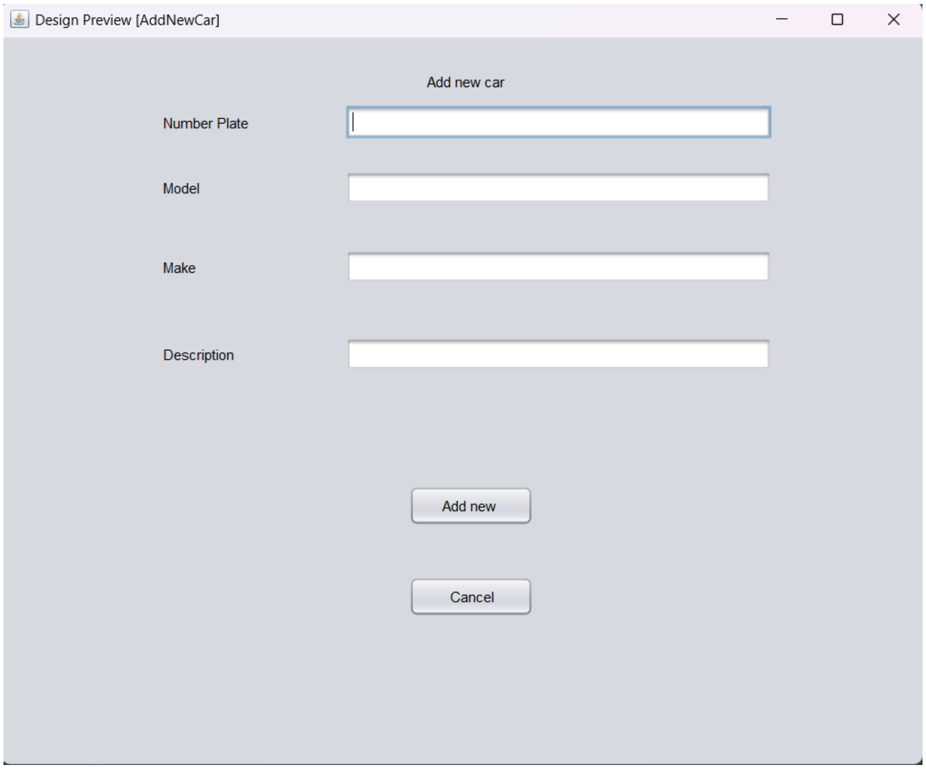


-SearchCarFrm:

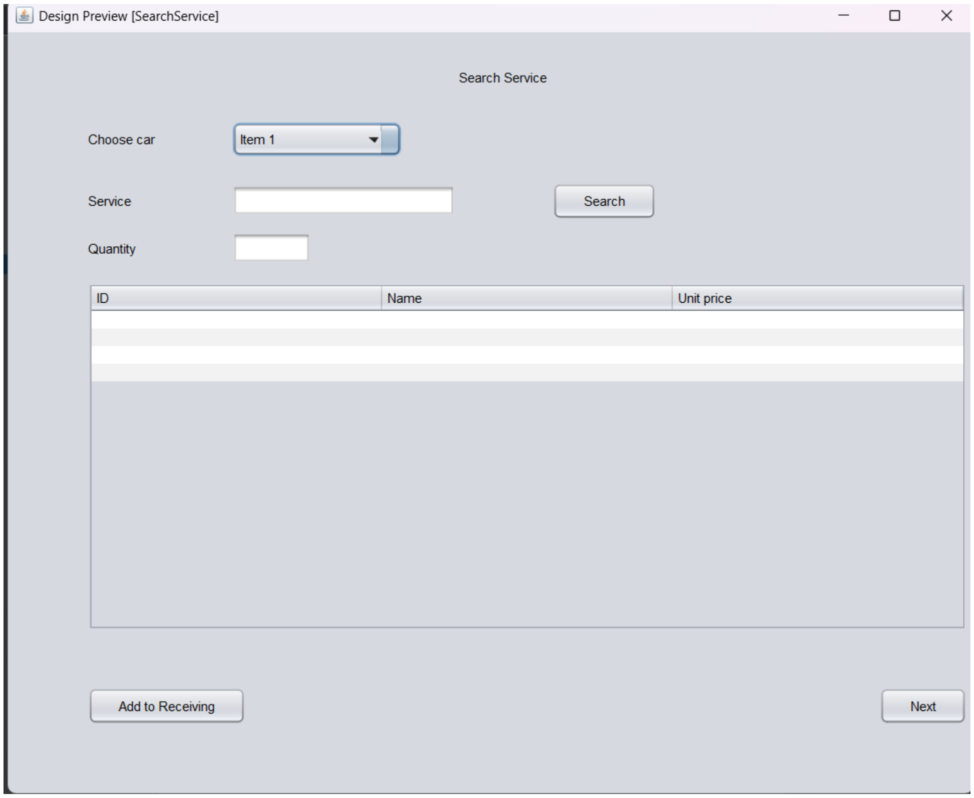


ID	Number Plate	Model	Make	Description

-AddCarFrm:

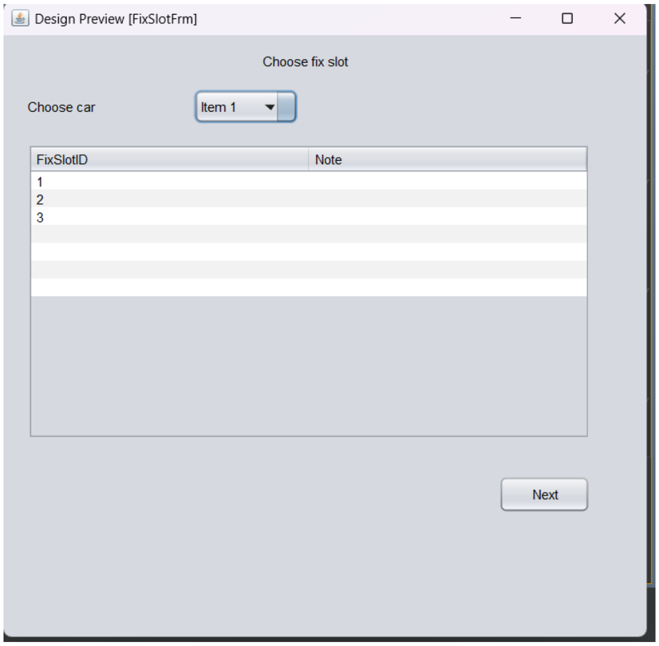


SearchServiceFrm:



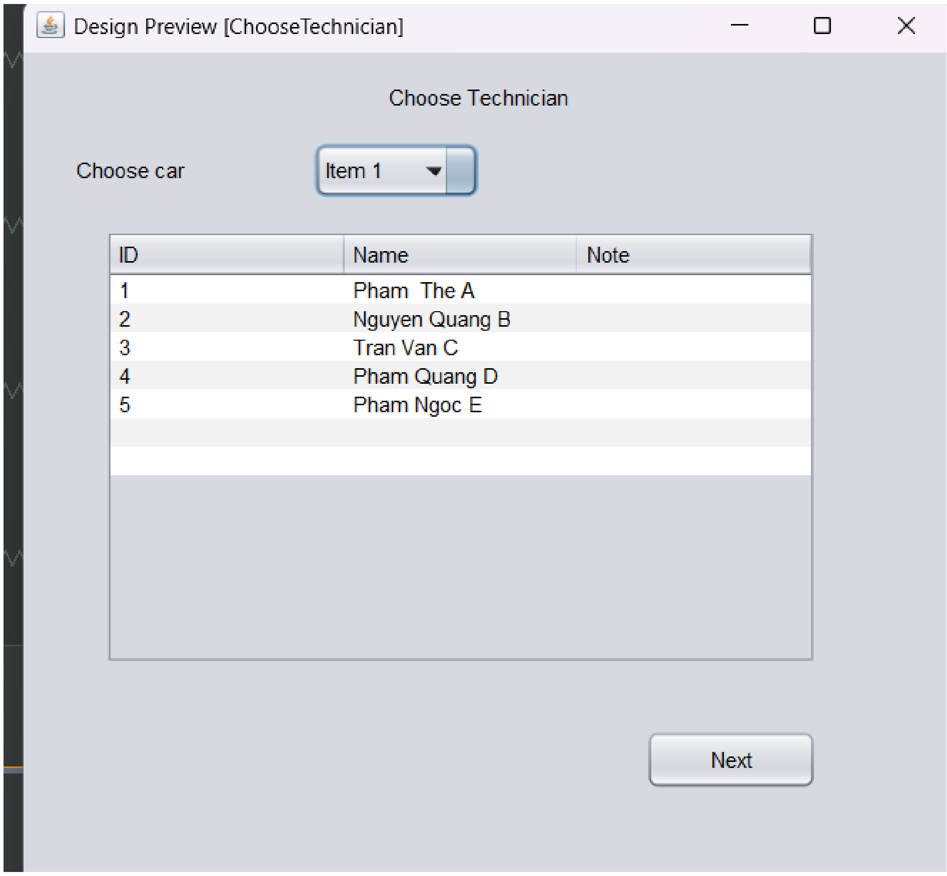
ID	Name	Unit price

Choose FixSlot:



FixSlotID	Note
1	
2	
3	

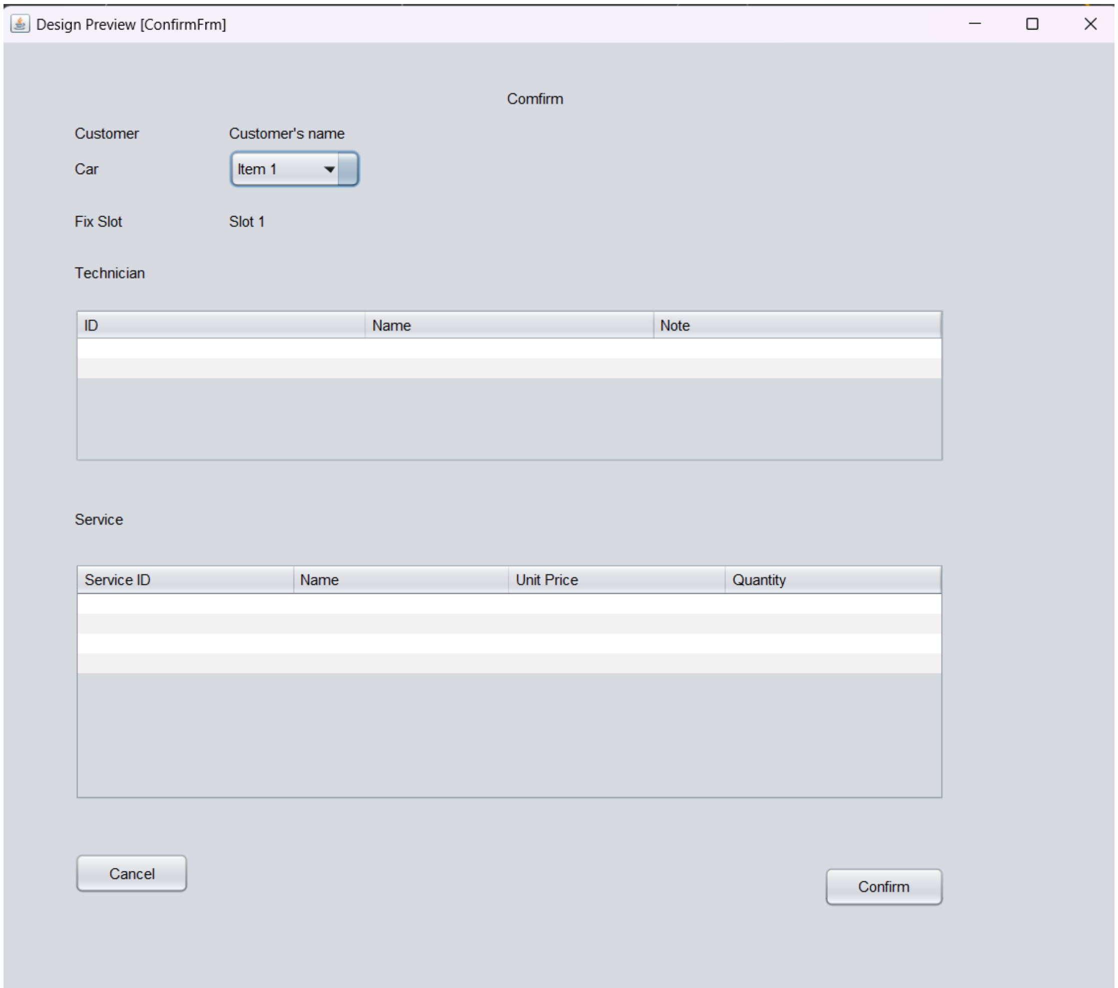
ChooseTechnicianFrm



Choose car Item 1

ID	Name	Note
1	Pham The A	
2	Nguyen Quang B	
3	Tran Van C	
4	Pham Quang D	
5	Pham Ngoc E	

ConfirmFrm:



Design Preview [ConfirmFrm]

Comfirm

Customer Customer's name

Car Item 1

Fix Slot Slot 1

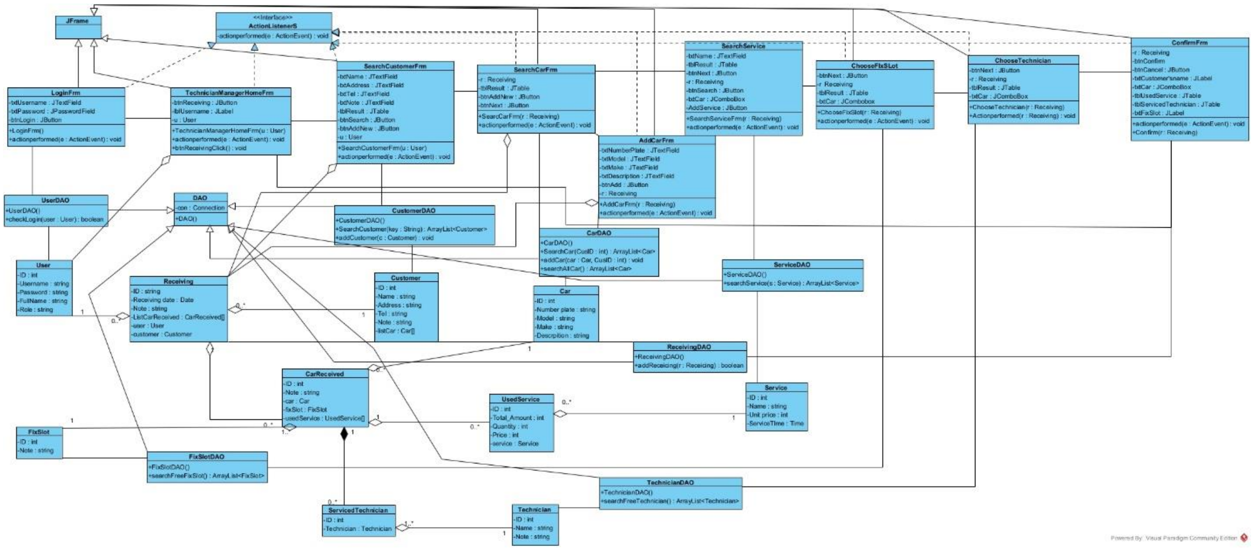
Technician

ID	Name	Note

Service

Service ID	Name	Unit Price	Quantity

Class Diagram:



9. Sequence diagram(Design phase).

Scenario v.3 for receiving car function

1. A Customer request TechnicianManager to receive his car(s).
2. A TechnicianManager enters his username, password, and clicks on the login button on LoginFrm.
3. The method actionPerformed() of LoginFrm is called.
4. The method actionPerformed() calls User to create an User object.
5. The class User packs the information into an User object
6. The class User returns User object to the method actionPerformed().
7. The method actionPerformed() calls method checkLogin() of the class UserDAO.
8. The method checkLogin() checks the login information.
9. The method checkLogin() calls the class User set more 3 attributes id,fullname, role.
10. The class User calls its method setUsername(), setRole(),setID().
11. The class User returns the User object to the method checkLogin().
12. The method checkLogin() returns the results to the actionPerformed().
13. The method actionPerformed() calls the class TechnicianManagerHomeFrm.
14. The constructor TechnicianManagerHomeFrm() is called.
15. The interface SearchFreeRoomFrm is shown to the seller.
16. The TechnicianManager clicks on the Receiving car button.
17. The method actionPerformed() is called
18. The method actionPerformed() calls the class SearchCustomerFrm.
19. The constructor SearchCustomerFrm() is called
20. The interface SearchCustomerFrm is shown to the manager.
21. The Technician Manager asks personal information about the Customer.
22. The Customer provides them to Technician Manager .
23. The Technician Manager enters the customer’s information and clicks on the search button.
24. The method actionPerformed() of class SearchCustomerFrm is called.

25. The method actionPerformed() calls the method searchCustomer() of class CustomerDAO.
26. The method searchCustomer() executes.
27. The method searchCustomer() calls the class Customer to pack the results.
28. The class Customer pack each result into a Customer object
29. The class Customer returns the object to the method searchCustomer().
30. The method searchCustomer() returns the results to the method actionPerformed()
31. The method actionPerformed() displays the results on the interface SearchCustomerFrm to the TechnicianManager.
32. The Technician Manager chooses the row that corresponds to the current Customer.
33. The method actionPerformed() of the class SearchCustomerFrm is called.
34. The method actionPerformed() calls the class Receiving to pack information to transfer to other interface.
35. The class Receiving calls constructor Receiving(), method setUser(), setCustomer()
36. The class Receiving returns the Receiving object to the method actionPerformed().
37. The method actionPerformed() calls the class SearchCarFrm.
38. The constractor SearchCarFrm() is called.
39. The interface SearchCarFrm is shown to the TechnicianManager.
40. The Technician Manager asks informations about the Car.
41. The Customer provides them to Technician Manager .
42. The Technician Manager enters the customer and clicks on the search button.
43. The method actionPerformed() of the class SearchCarFrm is called.
44. The method actionPerformed() calls the method searchCar() of class CarDAO.
45. The method searchCar() executes.
46. The method searchCar() calls the class Car to pack the results.
47. The class Car pack each result into a Car object.
48. The class Car returns the object to the method searchCar().
49. The method searchCar() returns the results to the method actionPerformed()
50. The method actionPerformed() displays the results on the interface SearchCarFrm to the TechnicianManager.

51. The Technician Manager choose car(s) and clicks on the next button.
52. The method actionPerformed() of class SearchCarFrm is called.
53. The actionPerformed() method calls the CarReceived class.
54. The CarReceived class calls constructor CarReceived () and its setters.
55. The CarReceived class return an CarReceived object to actionPerformed() method.
56. The actionPerformed() method calls the Receiving class
57. The Receiving class calls getListCarReceived().add() method
58. The getListCarReceived().add() method returns to the actionPerformed() method
59. The class actionPerformed() calls the class SearchServiceFrm.
60. The constructor SearchServiceFrm() is called
61. The interface SearchServiceFrm is shown to the Technician.
62. The seller informs the car's status to the Customer and requires the customer to choose Service(s).
63. The Customer provides the service(s)'s name to Technician Manager.
64. The Technician Manager enters the service(s)'s name , car(s)'s id(may be the customer brings many car)and clicks on the search button.
65. The method actionPerformed() of class SearchServiceFrm is called.
66. The method actionPerformed() calls the method searchService() of class ServiceDAO.
67. The method searchService() executes.
68. The method searchService() calls the class Service to pack the results.
69. The class Service pack each result into a Service object.
70. The class Service returns the object to the method searchService().
71. The method searchService() returns the results to the method actionPerformed()
72. The method actionPerformed() displays the results on the interface SearchService to the TechnicianManager.
73. The Technician Manager choose service(s)(change the car's ID if change the ReceivedCar) and clicks on the next button after adding service(s) for car(s).
74. The method actionPerformed() of class SearchServiceFrm is called.
75. The actionPerformed() method calls the UsedService class.
76. The UsedService class calls constructor UsedService () and its setters.
77. The UsedService class return an UsedService object to actionPerformed() method.

78. The actionPerformed() method calls the CarReceived class
79. The CarReceived class calls getListUsedService().add() method
80. The getList UsedService().add() method returns to the actionPerformed() method
81. The class SearchServiceView displays the Services which has been used to the Technician Manager.
82. The Technician Manager click next button after adding services.
83. The method actionPerformed() of class SearchServiceFrm is called.
84. The class actionPerformed() calls the class ChooseFixSlotFrm.
85. The constructor ChooseFixSlotFrm() is called.
86. The interface ChooseFixSlotFrm is shown to the Technician.
87. The Technician Manager choose the car's numberPlate.
88. The method actionPerformed() of the class ChooseFixSlotFrm is called.
89. The method actionPerformed() calls the method searchFreeFixSlot() of class FixSlotDAO.
90. The method searchFreeFixSlot() executes.
91. The method searchFreeFixSlot() calls the class FixSlot to pack the results.
92. The class FixSlot pack each result into a FixSlot object.
93. The class FixSlot returns the object to the method searchFreeFixSlot ().
94. The method searchFreeFixSlot() returns the results to the method actionPerformed()
95. The method actionPerformed() displays the results on the interface ChooseFixSlotFrm to the TechnicianManager.
96. The Technician Manager chooses the fixslot.
97. The method actionPerformed() of class ChooseFixSlotFrm is called.
98. The actionPerformed() method calls the CarReceived class
99. The CarReceived class calls setFixSlot() method
100. The setFixSlot() method returns to the actionPerformed() method
101. The class ChooseFixSlotFrm displays the Services which has been selected to the Technician Manager.
102. The Technician Manager click next button after adding fix slot.
103. The method actionPerformed() of class ChooseFixSlotFrm is called.
104. The class actionPerformed() calls the class ChooseTechnicianFrm.
105. The constructor ChooseTechnicianFrm() is called.
106. The interface ChooseTechnicianFrm is shown to the Technician.
107. The Technician Manager choose the car's numberPlate.

108. The method actionPerformed() of the class ChooseTechnicianFrm is called.

109. The method actionPerformed() calls the method searchFreeTechnician () of class TechnicianDAO.

110. The method searchFreeTechnician () executes.

111. The method searchFreeTechnician () calls the class Technician to pack the results.

112. The class Technician pack each result into a Technician object.

113. The class Technician returns the object to the method searchFreeTechnician().

114. The method searchFreeTechnician() returns the results to the method actionPerformed()

115. The method actionPerformed() displays the results on the interface ChooseTechnicianFrm to the TechnicianManager.

116. The Technician Manager chooses the technician.

117. The method actionPerformed() of class ChooseTechnicianFrm is called.

118. The actionPerformed() method calls the ServicedTechnician class.

119. The ServicedTechnician class calls constructor ServicedTechnician() and its setters.

120. The ServicedTechnician class return a ServicedTechnician object to actionPerformed() method.

121. The actionPerformed() method calls the CarReceived class.

122. The CarReceived class calls getListServicedTechnician().add() method

123. The getListServicedTechnician().add() method returns to the actionPerformed() method.

124. The class ChooseTechnicianFrm displays the Technician(s) which has been used to the Technician Manager.

125. The Technician Manager click next button after adding technician(s).

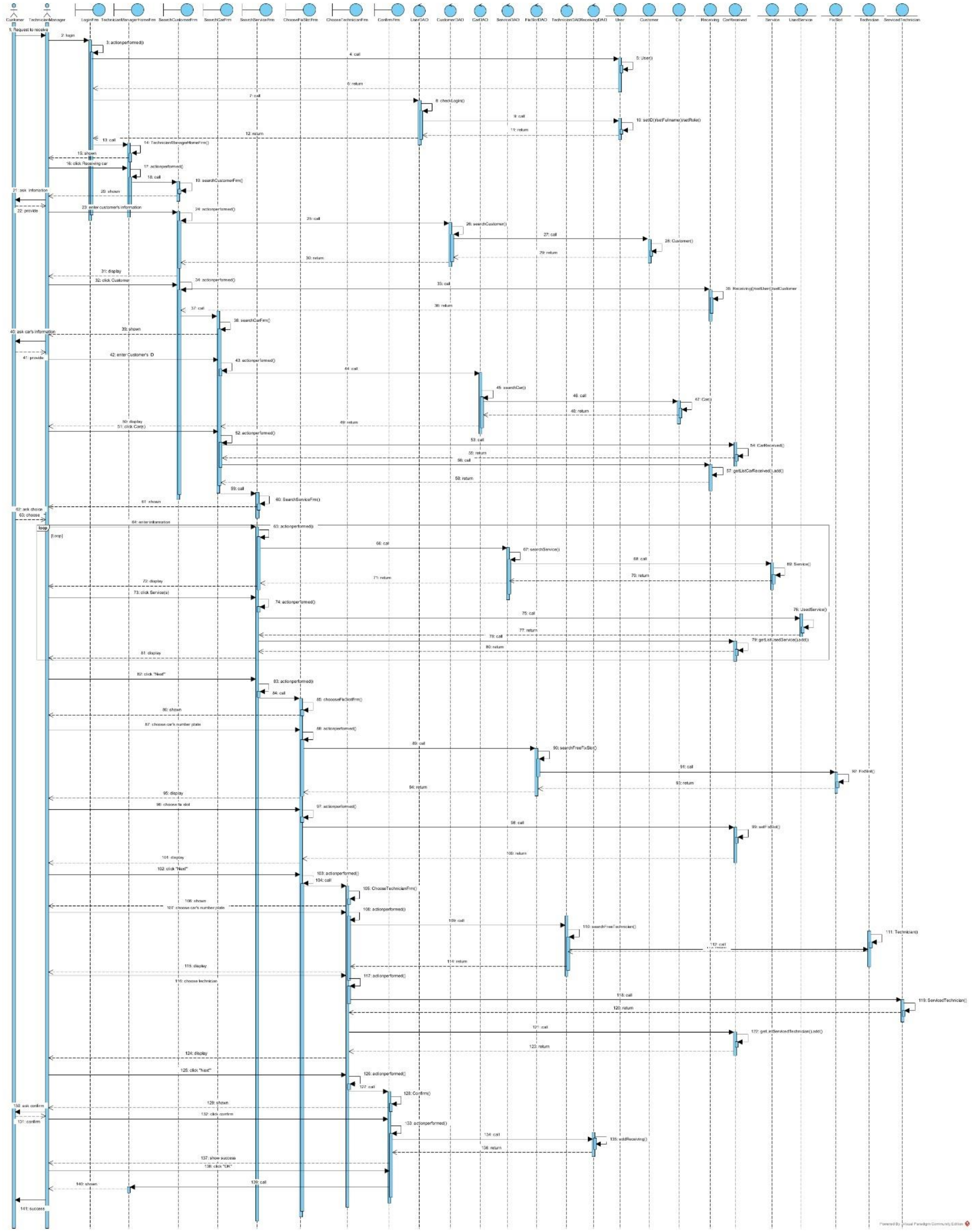
126. The method actionPerformed() of class ChooseTechnicianFrm is called.

127. The class actionPerformed() calls the class ConfirmFrm.

128. The constructor ConfirmFrm() is called

129. The interface ConfirmFrm is shown to the TechnicianManager.

130. The Technician repeats the receiving information to the customer and asks him to confirm.
131. The customer confirms it.
132. The TechnicianManager clicks on the confirm button.
133. The method actionPerformed() of the class ConfirmFrm is called.
134. The method actionPerformed() calls the method addReceiving() of the class ReceivingDAO.
135. The method addReceiving() executes.
136. The method addReceiving() returns the turn to the method actionPerformed()
137. The method actionPerformed() displays a success message
138. The TechnicianManager clicks on the OK button of the message.
139. The method actionPerformed() recalls the interface TechnicianManagerHomeFrm
140. The interface TechnicianManagerHomeFrm is shown to the TechnicianManager
141. The TechnicianManager confirms the successful receivingto the Customer .


Customer
TechnicianManager
LoginFrm
TechnicianManagerHomeFrm
SearchCustomerFrm
SearchCarFrm
SearchServiceFrm
ChooseFixSlotFrm
ChooseTechnicianFrm
ConfirmFrm
UserDAO
CustomerDAO
CarDAO
ServiceDAO
FixSlotDAO
TechnicianDAO
ReceivingDAO
User
Customer
Car
Receiving
CarReceived
Service
UsedService
FixSlot
Technician
ServicedTechnician
1: Request to receive
2: login
3: actionperformed()
4: call
5: User()
6: return
7: call
8: checkLogin()
9: call
10: setID()/setFullname()/setRole()
11: return
12: return
13: call
14: TechnicianManagerHomeFrm()
15: shown
16: click Receiving car
17: actionperformed()
18: call
19: searchCustomerFrm()
20: shown
21: ask information
22: provide
23: enter customer's information
24: actionperformed()
25: call
26: searchCustomer()
27: call
28: Customer()
29: return
30: return
31: display
32: click Customer
33: call
34: actionperformed()
35: Receiving()/setUser()/setCustomer
36: return
37: call
38: searchCarFrm()
39: shown
40: ask car's information
41: provide
42: enter Customer's ID
43: actionperformed()
44: call
45: searchCar()
46: call
47: Car()
48: return
49: return
50: display
51: click Car(s)
52: actionperformed()
53: call
54: CarReceived()
55: return
56: call
57: getListCarReceived().add()
58: return
59: call
60: SearchServiceFrm()
61: shown
62: ask choice
63: choose
64: enter information
loop
[Loop]
65: actionperformed()
66: call
67: searchService()
68: call
69: Service()
70: return
71: return
72: display
73: click Service(s)
74: actionperformed()
75: call
76: UsedService()
77: return
78: call
79: getListUsedService().add()
80: return
81: display
82: click "Next"
83: actionperformed()
84: call
85: chooseFixSlotFrm()
86: shown
87: choose car's number plate
88: actionperformed()
89: call
90: searchFreeFixSlot()
91: call
92: FixSlot()
93: return
94: return
95: display
96: choose fix slot
97: actionperformed()
98: call
99: setFixSlot()
100: return
101: display
102: click "Next"
103: actionperformed()
104: call
105: ChooseTechnicianFrm()
106: shown
107: choose car's number plate
108: actionperformed()
109: call
110: searchFreeTechnician()
111: Technician()
112: call
114: return
115: display
116: choose technician
117: actionperformed()
118: call
119: ServicedTechnician()
120: return
121: call
122: getListServicedTechnician().add()
123: return
124: display
125: click "Next"
126: actionperformed()
127: call
128: Confirm()
129: shown
130: ask confirm
131: confirm
132: click confirm
133: actionperformed()
134: call
135: addReceiving()
136: return
137: show success
138: click "OK"
139: call
140: shown
141: success
Powered By Visual Paradigm Community Edition

10. Test plan and standard test case for black box testing of the module

Black-box test case list:

No.	Module	Test case
1	Receiving a car	There is customer exists, car exists, available Service ,available FixSlot, available Technician
2		There is customer exists, car exists, available Service,available FixSlot, no available Technician
3		There is customer exists, car exists, available Service,no available FixSlot
4		There is customer exists, car exists, no available Service
5		There is customer exists, car . does not exist, available Service, available FixSlot, available Technician
6		There is customer exists, car does not exist, available Service, available FixSlot, no available Technician
7		There is customer exists, car does not exist, available Service, no available FixSlot.
8		There is customer does not exists , car does not exists ,available Service, available FixSlot, no available Technician
9		There is customer does not exists , car does not exists ,available Service, available FixSlot, available Technician
10		There is customer does not exists , car does not exists ,available Service, no available FixSlot
11		There is customer does not exists , car does not exists ,no available Service
12		Add 2 receiving with a Customer ,Car ,Service ,FixSlot,Technician

Test case No1(Standard test case)

The database before testing:

tblUser :

ID	Username	Password	Fullname	Role
1	user1	1	Pham Van An	Technician Manager
2	user2	2	Nguyen Thi Binh	Cashier
3	user3	3	Tran Van Cuong	Manager

tblCustomer :

ID	Name	Address	Tel	Note
1	Pham The Du	Nguyễn Trãi, Hà Đông	0123456	
2	Vu Nam Duong	Hải Phòng	0111222	
3	Nguyen Quang Vu	Giải Phóng ,Hai Bà Trưng	0654321	
4	Tran Tuan Cuong	Long Biên	0111111	

tblCar:

ID	CustomerID	Number plate	Model	Make	Description
1	1	30K-99999	Vios	Toyota	Black
2	3	14K-11111	CX9	Mazda	Blue
3	2	18K-88888	Accent	Huyndai	White

tblReceiving:

ID	CustomerID	UserID	ReceivingDate	Note
1	1	1	2025-04-30 08:00:00	
2	2	1	2025-05-01 09:15:00	
3	3	1	2025-05-05 14:30:00	

tblFixSlot:

ID	Note
1	Slot 1
2	Slot 2
3	Slot 3
4	Slot 4
5	Slot 5

tblCarReceived:

ID	ReceivingID	CarID	FixSlotID	Note
1	1	1	1	
2	2	3	1	
3	3	2	2	

tblTechnician

ID	Name	Note
1	Pham The A	
2	Nguyen Quang B	
3	Tran Van C	
4	Pham Quang D	
5	Pham Ngoc E	

tblServicedTechnician:

ID	TechnicianID	tblCarReceivedID
1	1	1
2	2	2
3	3	3

tblService (include Service and Part):

ID	Name	Unit Price	ServiceTime
1	Battery Replacement	5000000.00	01:00:00
2	Change glasses	1000000.00	02:00:00
3	Change the oil	500000.00	00:45:00
4	Paint the car	2000000.00	01:00:00
5	Engine test	1000000.00	01:30:00
6	Replace tire	100000.00	00:10:00
7	Maintenance	1000000.00	02:00:00
8	Washing	20000.00	00:30:00

tblUsedService:

ID	ServicedID	CarReceivedID	Quantity	Price
1	1	1	1	5000000.00
2	6	1	4	100000.00
3	4	2	1	2000000.00
4	2	3	1	1000000.00

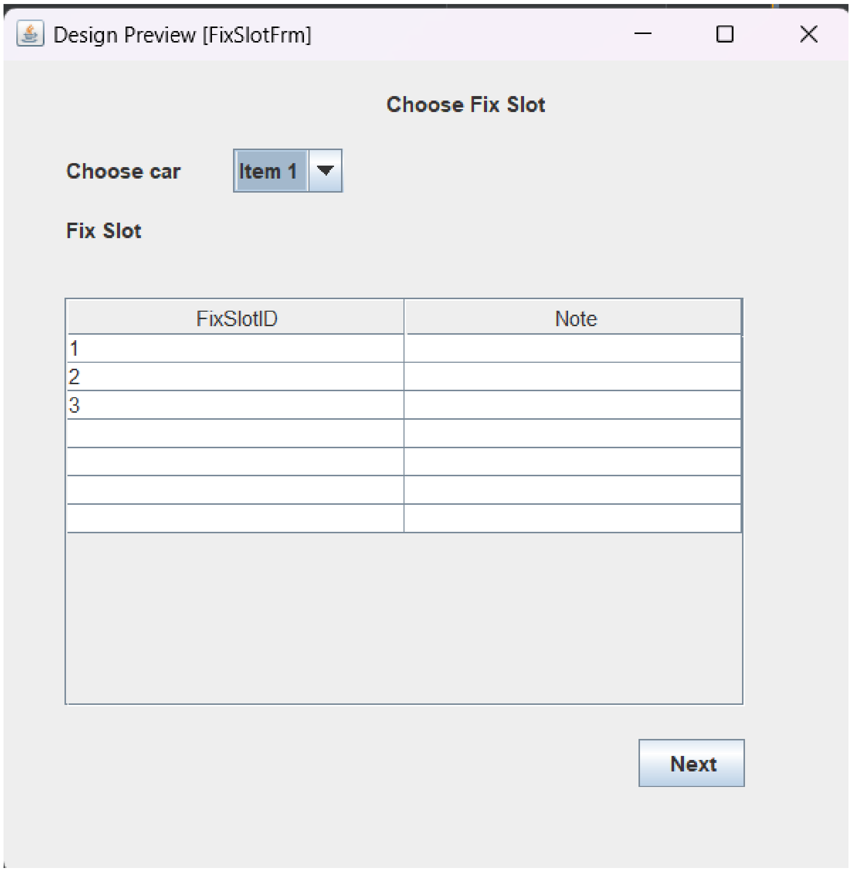
The testing scenario and expected results

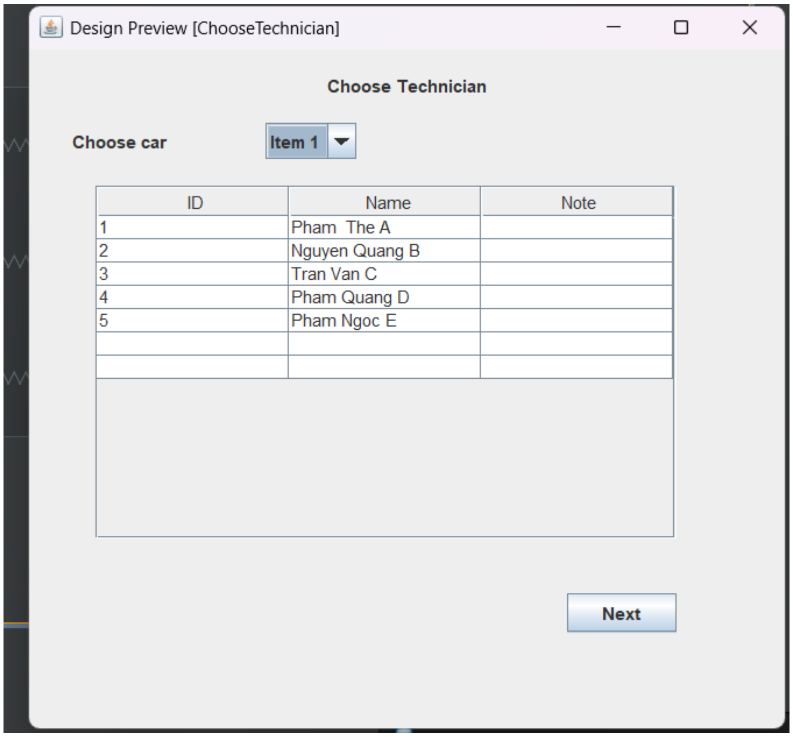
<table>
<tr><td>Scenario</td><td>Expected results</td></tr>
<tr><td>1. Start the application</td><td>The login interface appeared with: text fields for entering username, password, a login button</td></tr>
<tr><td>2. Enter username = user1
password = 1
and click on login button</td><td>The home interface of the Technician Manager appeared with a button: Receiving</td></tr>
<tr><td>3. Click on the receving button</td><td>The search customer interface appeared with 4 text field(Name,Adress,Tel,Note) a search button and a button add new,button next</td></tr>
<tr><td>4. Enter:
– name = Du
click on the Search button</td><td>List customer appeared and a button next below:
<table>
<tr><td>ID</td><td>Name</td><td>Address</td><td>Tel</td><td>Note</td><td></td></tr>
<tr><td>1</td><td>Pham The Du</td><td>Nguyễn Trãi, Hà Đông</td><td>0123456</td><td></td><td>□</td></tr>
<tr><td>2</td><td>Vu Nam Duong</td><td>Hải Phòng</td><td>0111222</td><td></td><td>□</td></tr>
</table>
Next</td></tr>
<tr><td>5. Click on the next button</td><td>List customer’s car appeared and a button next below:
<table>
<tr><td>ID</td><td>CustomerID</td><td>Number plate</td><td>Model</td><td>Make</td><td>Description</td><td></td></tr>
</table></td></tr>
</table>

<table>
<tr>
<td></td>
<td>
<table>
<tr><td>1</td><td>1</td><td>30K-9999</td><td>Vios</td><td>Toyota</td><td>Black</td><td>□</td></tr>
</table>
<table>
<tr><td>Next</td></tr>
<tr><td>Add new car</td></tr>
</table>
</td>
</tr>
<tr>
<td>5. Click on the next button</td>
<td>The search customer interface appeared with 3 text field : Name and CarID(change the carID if change the ReceivedCar),quantity   a search button , a button add and and a button next</td>
</tr>
<tr>
<td>6. Enter:
– name = Replace tire and
click on the Search button</td>
<td>List service appeared :
<table>
<tr><th>ID</th><th>Name</th><th>Unit Price</th><th>ServiceTime</th></tr>
<tr><td>6</td><td>Replace tire</td><td>100000.00</td><td>00:10:00</td></tr>
</table>
Quantity: ……..
<table><tr><td>Add</td></tr></table>
<table><tr><td>Next</td></tr></table>
</td>
</tr>
</table>

<table>
<tr>
<td>7.Enter the quantity and click add to add the service and after adding service(s) click next.The

Interface

chooseFixSlotFrm appear</td>
<td>
Design Preview [FixSlotFrm]
Choose Fix Slot
Choose car Item 1
Fix Slot
<table>
<tr><th>FixSlotID</th><th>Note</th></tr>
<tr><td>1</td><td></td></tr>
<tr><td>2</td><td></td></tr>
<tr><td>3</td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td>8.Click Slot 1 and click next. The Interface of

chooseTechnianFrm appear</td>
<td>
Design Preview [ChooseTechnician]
Choose Technician
Choose car Item 1
<table>
<tr><th>ID</th><th>Name</th><th>Note</th></tr>
<tr><td>1</td><td>Pham The A</td><td></td></tr>
<tr><td>2</td><td>Nguyen Quang B</td><td></td></tr>
<tr><td>3</td><td>Tran Van C</td><td></td></tr>
<tr><td>4</td><td>Pham Quang D</td><td></td></tr>
<tr><td>5</td><td>Pham Ngoc E</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr>
<td>9.Click Technician 1 and click next.

The confirmFrm apppear</td>
<td>The confirm interface appeared:
+ Customer:
– Name: Pham The Du
– address: Nguyễn Trãi,Hà Đông
– tel: 0123456
+ Receiving:

1.Car
<table>
<tr><td>ID</td><td>CustomerID</td><td>Number plate</td><td>Model</td><td>Make</td><td>Description</td></tr>
<tr><td>1</td><td>1</td><td>30K-99999</td><td>Vios</td><td>Toyota</td><td>Black</td></tr>
</table>
2.Service:

CarID : 1
<table>
<tr><td>ID</td><td>Name</td><td>Unit Price</td><td>Quantity</td></tr>
<tr><td>6</td><td>Replace tire</td><td>100000.00</td><td>4</td></tr>
</table>

Confirm

Cancel</td>
</tr>
<tr>
<td>10.Click confirm button</td>
<td>A message appeared: Receiving successful!</td>
</tr>
</table>

11.Click Ok	Return to the home interface of the Technician Manager

The database after testing:

tblReceiving:

ID	CustomerID	UserID	ReceivingDate	Note
1	1	1	2025-04-30 08:00:00	
2	2	1	2025-05-01 09:15:00	
3	3	1	2025-05-05 14:30:00	
4	1	1	2025-05-22 10:00:00	

tblCarReceived:

ID	ReceivingID	CarID	FixSlotID	Note
1	1	1	1	
2	2	3	1	
3	3	2	2	
4	4	1	1	

tblUsedServiced:

ID	ServicedID	CarReceivedID	Quantity	Price
1	1	1	1	5000000.00
2	6	1	4	100000.00
3	4	2	1	2000000.00
4	2	3	1	1000000.00
5	6	4	6	1000000.00

tblServicedTechnician:

ID	TechnicianID	tblCarReceivedID
1	1	1
2	2	2
3	3	3
4	1	4