COIS 2240 Final project Report
April 6th,2018
Nikolas Gordon, Batstone Christyanton, Taran Dorland, Kevin Dsane-Selby

Overview

The overall usage of our program will be for users who belong to a dealership such as a manager, retailer, or even a service provider. Initial, this program will be used to manage and oversee not only the stock of cars, but servicing, and payment. This can allow the user to have seamless flow through the program whether they are inputting, pulling, checking, editing data all at once. This is achieved with the use of databases which keeps all necessary info such as car owner name and car type. The usefulness of this program would be, instead of keeping names and car information in books and external websites, you now can do it in your own program.

Industry

The intended target audience should be those in the position of management and employees at car dealerships as well, as those in that work at the automotive assemblers. They will be dealing with the validation process behind inventory being sold and that are in stock. Customers as well be a target audience as they will have access to the knowledge of cars on sale and that are in stock. This program is initially targeted to the industry of automation as it can be possible many car dealerships can be in contact with multiple automotive companies at once. There are many audiences and groups of people that can be involved with this program and can be useful for the automation industry. The initial main users of this program will be the employers at the dealership as the deal with the processing of everything coming in and out of the dealership.

Class diagram

The class diagram goes over the different class and how they relate towards each other. These relations can stem upon various descriptions such as association, aggression, and dependency just to name a few. First, we have the dealership class, which introduces the customer to the program. Its essentially the main. The dealership class has three main connection, the first one being car class which has 2 inherencies, car sales and car service. The next dealership connection is booking type and finally, the last connection is the customer which inherits the payment class

Sequence diagram

The sequence diagram models the user interaction with the various departments of the dealership. At first the user will contact the dealership and then proceed to either sales, services, and parts. There are many options for the user to choose from which is why the chart is somewhat complicated. In short if he user picks sales he has the option add, edit, delete, and search for cars. The other department is service which also has its own database and customer classes, as some people may just want to deal with one department, not all.

Activity diagram

The activity diagram deals with how the program flows through out. In other words what happens as certain times and in all what the program is doing. Very similar to the sequence diagram. We utilized the sequence diagram to resolve to the different departments and how they handle user data and how certain choices pan out. At first, we have the user class, which is connected to five different choices, four of them being add and manage both car and customer, with the five un-forked node being user managed. The four nodes combine to one to indicate each different process proceeds to the booking class and then finishes at the payment class.

Activity diagram (swim lane)

Now with the use of swim lanes, we see the use of departments and how they use information as well as how they flow, it is very similar to the regular activity diagram it now just shows how method such as add and manage customer are together as well as add car and manage car. Lastly the booking management as its own lane with payment classes within each department.

Prospective

The easiest thing to handle was the use of scene builder. Instead of having to write the position of certain text fields, labels, pictures, etc. we could simply place a them into the builder. It saved a lot of time compared to hard coding those libraries.

The most difficult would be the implementation of the databases. More so the application such as having stored values in the database show up into a drop-box. The database application made us abandon some important features such as edit and search the database. Our combinations box and choice box did not work, we had to instead put it into a text field and manually remove or add info. In hindsight, it would have been better to see how much vigor is needed to create such a database or instead build around how much we know about databases instead of diving head first into a lot of database application.

One thing we could change would be to get a head start on the assignment way before the actual due date. We still had some time left but doing it before would have caused less of a headache. Another thing, as said before was to understand the intensity of an application instead of speculating. This came to hurt us in the end not because we did not know how to implement certain processes but simply a combination of both timing and intensity of the application. In other words, building an intensive application with a lot of time in advance.

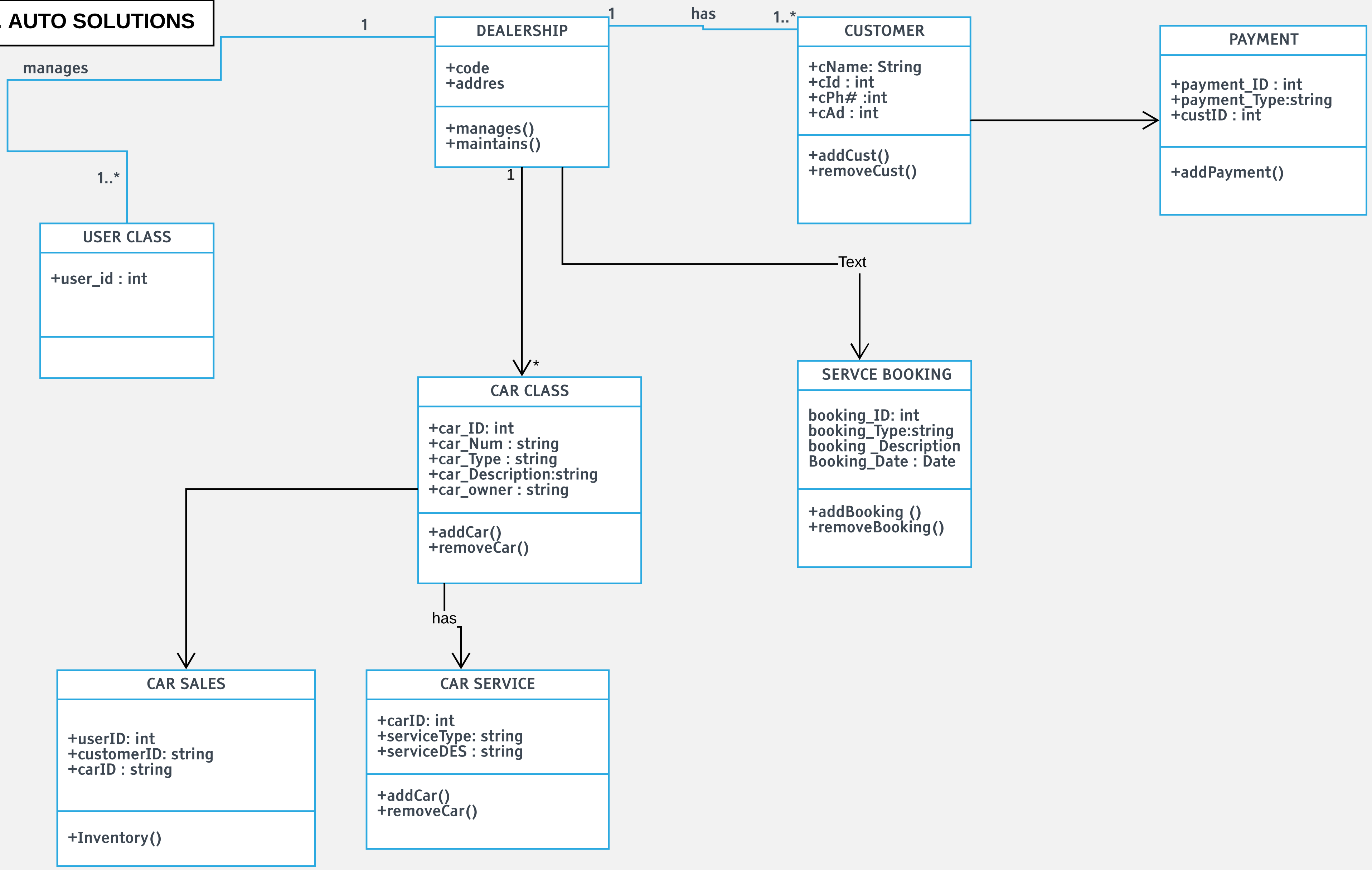
In all, we first learned how to use just java-fx for our application. Later we used scene builder with databases and applications relying on databases. We overall learned a glimpse about what developers must write in, but the importance of planning ahead as well as thinking about how the user should flow throughout the program.

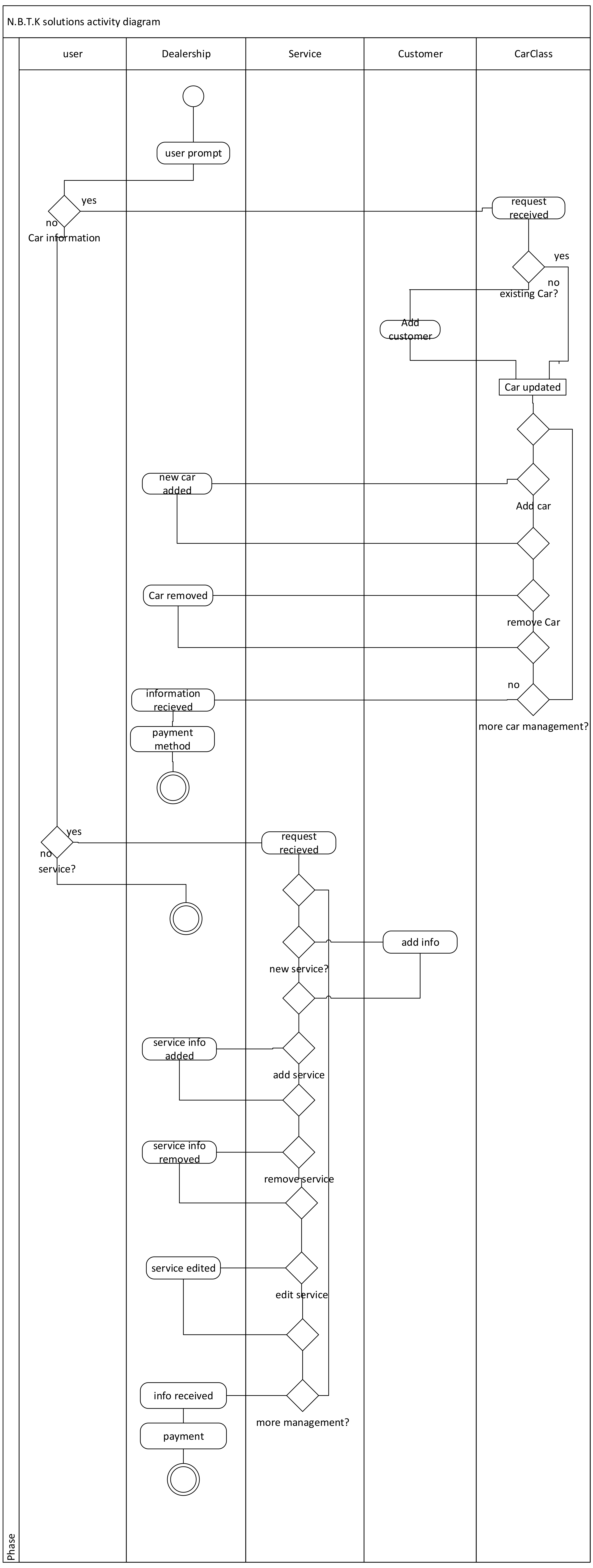
Conclusion

Overall this project demonstrated the use of databases and manipulation towards these databases through user interaction. We believe this to be useful towards users who are managing databases towards there system. We did have some dilemma’s, such as compromising drop-down menus as well as combination boxes. Ultimately, the most important thing was linking all operations together and making sure they worked effectively and efficiently amongst each other.

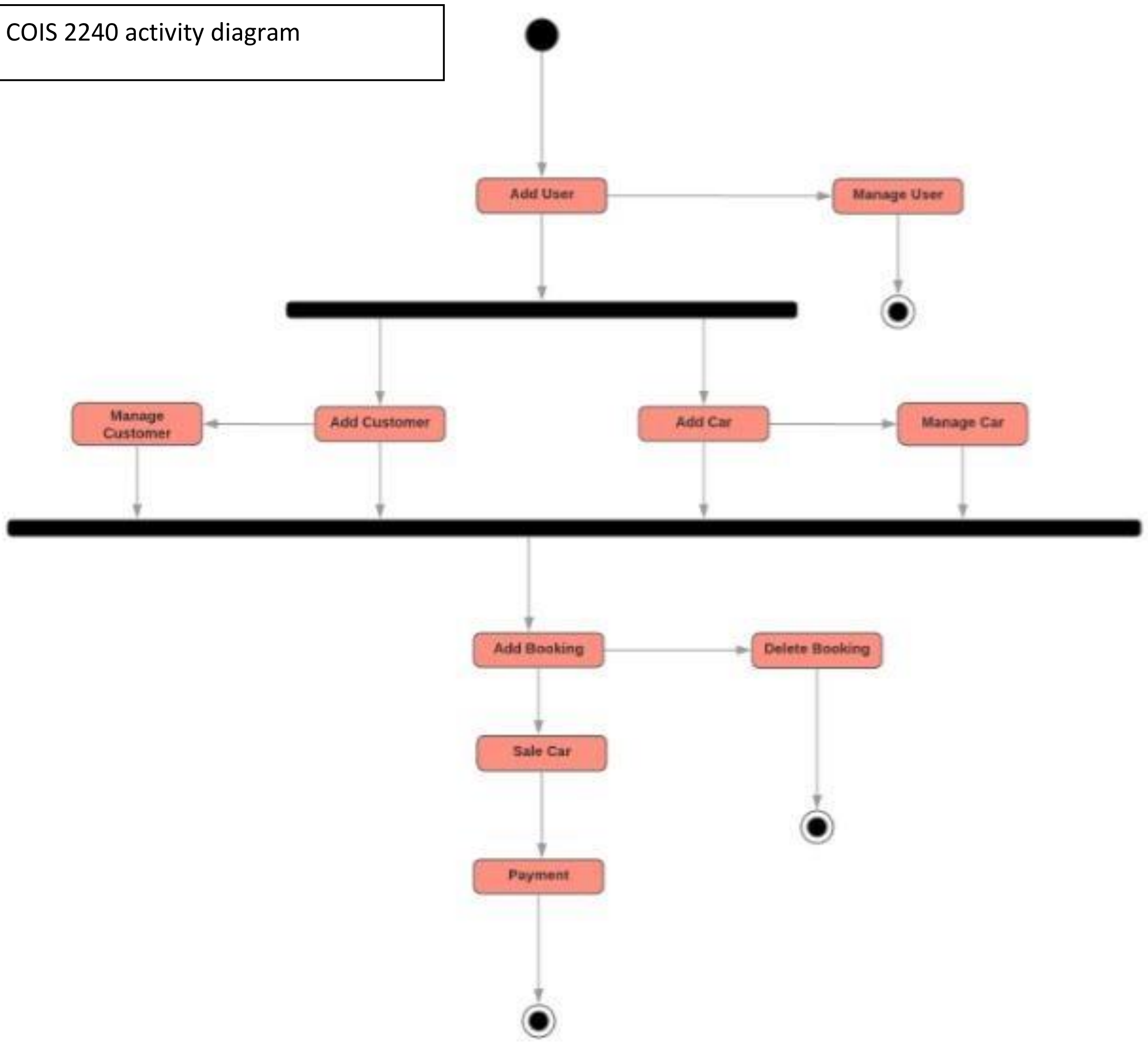
UML diagrams:

N.B.T.K. AUTO SOLUTIONS




N.B.T.K solutions activity diagram
user
Dealership
Service
Customer
CarClass
Phase
user prompt
yes
no
Car information
request received
yes
no
existing Car?
Add customer
Car updated
new car added
Add car
Car removed
remove Car
no
information recieved
more car management?
payment method
yes
no
service?
request recieved
add info
new service?
service info added
add service
service info removed
remove service
service edited
edit service
info received
more management?
payment

COIS 2240 activity diagram



COIS 2240 final project
sequence diagram

April 6th,2018

Nikolas Gordon, Batstone Christyanton, Taran Dorland, Kevin Dsane-Selby

1.User is prompt with catalogue

2. if user chooses sales they..

A).add car to database

B). Edit car from database

C). Search for car from the database

D) Remove car from database

E) return to menu

3) If user chooses Service..

A) Service added to database

B) Service removed from database

C)Edit service from the database

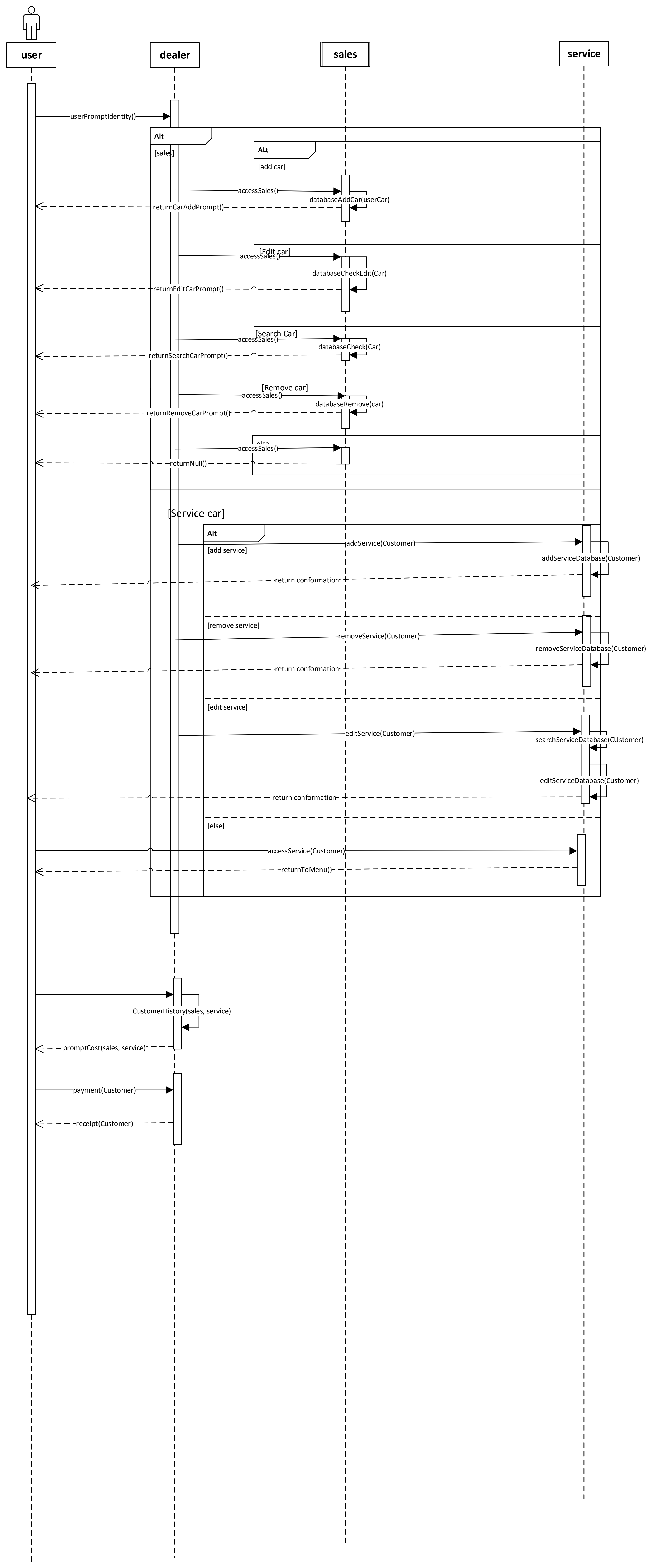
E) Return to menu

4) user is finished and wishes to pay

5). Order is made

6)User chooses payment type and pays

7) User receives receipt


user
dealer
sales
service
userPromptIdentity()
Alt
[sales]
ALt
[add car]
accessSales()
databaseAddCar(userCar)
returnCarAddPrompt()
[Edit car]
accessSales()
databaseCheckEdit(Car)
returnEditCarPrompt()
[Search Car]
accessSales()
databaseCheck(Car)
returnSearchCarPrompt()
[Remove car]
accessSales()
databaseRemove(car)
returnRemoveCarPrompt()
else
accessSales()
returnNull()
[Service car]
Alt
[add service]
addService(Customer)
addServiceDatabase(Customer)
return conformation
[remove service]
removeService(Customer)
removeServiceDatabase(Customer)
return conformation
[edit service]
editService(Customer)
searchServiceDatabase(CUstomer)
editServiceDatabase(Customer)
return conformation
[else]
accessService(Customer)
returnToMenu()
CustomerHistory(sales, service)
promptCost(sales, service)
payment(Customer)
receipt(Customer)