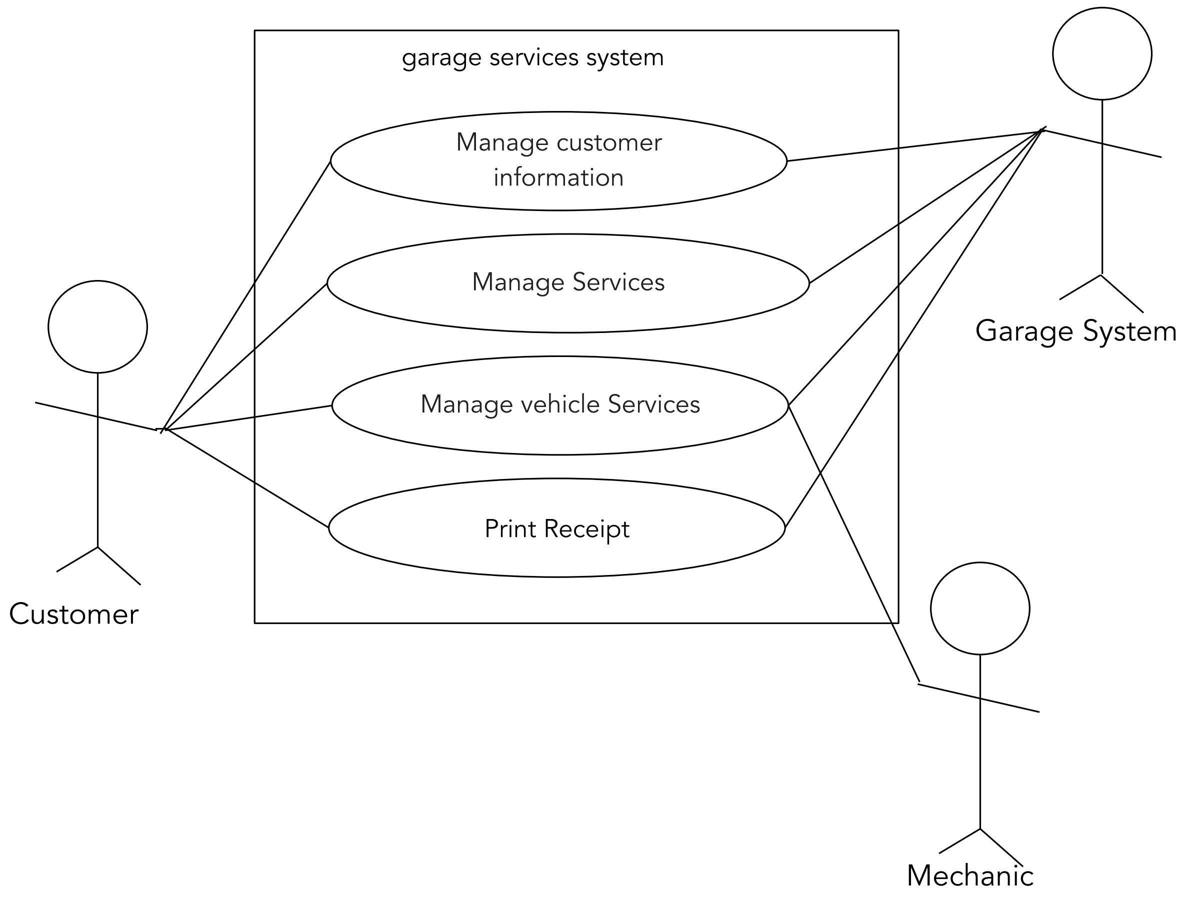
1. Identify the use cases for the software. Draw the **UML use-case diagram **and include supporting use-case descriptions. At-least 3 scenarios must be identified.



From the diagram, we can see three actors.

1) The customer is the vehicle owner who has brought their vehicle in for service. They use the garage's system to make appointments, bring and retrieve their vehicles, and pay for work.

2) The garage system is the group or system that runs the garage and does things like schedule appointments, assign mechanics, order parts, bill customers, and more.

3) The mechanic is the person who performs the actual repair or maintenance work on the customer's vehicle.

At the same time, we can see the whole system " the rectangle which is the garage services system " in the garage service system there are 4 use cases garage system used when a customer comes to the garage.

1) Manage customer information: In this scenario, the garage system communicates with the client to collect the data it needs to carry out the service. The consumer

plays a supporting role in the garage system in this scenario. The garage management system must gather information such as the customer's name, phone number, and vehicle specifics.

2) Manage services: The customer and the garage system are the main actors in this scenario. Clients make service requests, like oil changes and brake installations, through the garage's automated system. The garage system suggests a list for the customer containing the price and the service name. The consumer can add or remove the services and it is up to them to decide whether or not to proceed with the service. after choosing the services the garage system verifies the date for the customer to bring the vehicle.

3) Manage vehicle service: In this scenario, the customer, the garage management system, and the mechanic all play roles in the management of car servicing. When a customer brings their car in for service, the garage management system determines who will work on the car and sends them to them. A mechanic's job is to check out a car, figure out what's wrong with it, and fix or keep it in good shape. Once the service is finished, the garage's technology will send the customer a notification and a comprehensive report.

4) In this scenario, the customer and garage management system are the main actors, and the use case is to print a receipt. The consumer receives a receipt from the garage's system that itemizes the services rendered and the associated costs after the work is finished. The customer makes a payment for the work done, and the garage management system issues them a receipt for their records.

In summary, the relationships between the actors are as follows:

- In all scenarios, the garage system is the main player because it is in charge of the service and has to communicate with the other actors.
- In each scenario, the consumer must engage with the garage system in some way to either supply data, make a service request, or make a transactional payment.
- In the "Manage car services" use case, the mechanic communicates with the garage system to do the necessary servicing and repairs for the customer's vehicle.

Use case 1: Manage customer information

Use Case:	Manage customer information
Trigger:	The system manages customer information and registers him/her in the system.
Main scenarios:	1. The customer specifies his/her personal

	information first name, last name, EID, email, and phone number.
	2. The garage system verifies the customer’s information
	3. The garage system asks the customer to provide the vehicle information, make, year, model, type, color, and ID.
	4. The garage system verifies the vehicle information.
2a	- The customer information is incorrect. - Ex. (The customer ID is invalid, The customer number is invalid…) - The use case ends, and an error message is communicated.
3a	- The vehicle information is incorrect. - The use case ends, and an error message is communicated

Use Case:	Manage Services
Trigger:	The customer wants to add a garage service.
Prediction:	-
Main scenario:	1. Clients make service requests.
	2. The garage system displays a list of existing services.
	3. The customer selects the wanted services from the list.
	4. the customer can choose and remove unwanted services from the list.
	5. The garage system updates the list.
	6. The garage system verifies the list
	8. The garage system gives the customer date to bring the car.
	9. The garage system verifies the date.
6a	- The user wants to add one extra service. - The user wants to remove the service. - The use case ends, and an error

	message is communicated.
9a	- If the date is not suitable for the customer. - If the mechanic is not available on that date. - The use case ends, and an error message is communicated.

Use Case:	Manage vehicle Services
Trigger:	- A customer brings a vehicle for service
Prediction:	- The customer is registered - A list of wanted services is generated.
Main scenario:	1. The garage system initiates a new service request for the customer's car.
	2. The garage system assigns to the vehicle ID.
	3. A mechanic is sent to the service request by the garage system .
	4. The mechanic checks the service request.
	5. The mechanic performs the requested service on the vehicle.
	6. The mechanic logs the accomplished service with his name in the system.
	7. The mechanic confirms the service and tells the garage system that the service has been completed and writes a brief description of the service and how long it takes with the price.
	8. The garage system reviews the completed service request.
	9. The garage system tells the customer that the vehicle is ready for pickup.
8a	- if there is a service that has not been accomplished the use case ends, and an error message is communicated.

Use Case:	Print Receipt

Trigger:	- The customer wants to pay for the services that have been done.
Prediction:	- A list of the done services is generated with the mechanic's name.
Main scenario:	1. The system tells the user to enter any discount code that is available.
	2. User enters discount code, if available.
	3. If the code is valid, the system takes the discount off the total.
	4. System adds taxes based on the total cost.
	5. The system shows the user the total cost with taxes and any discounts.
	6. the customer chose the payment method
	7. the garage system verify the payment method.
	8.The system prints the receipt: - The customer information - The mechanic name - The vehicle information - The The date - The services with the price - The taxes - The discount - And the total cost . - mechanic's name - list of the accomplished services with each service price .
3a	If the code is invalid the use case ends, and an error message is communicated.
7a	If the payment method is invalid the use case ends, and an error message is communicated.

The classes:

1. Customer: the customer class

   Attributes :

   - firstName: string
   - lastName: staring
   - phoneNumber: string

- customerEID: string
- Email: string

2. vehicle :

   Attributes:
   - make: string
   - model: string
   - Year: string
   - color: Enum
   - ID: string

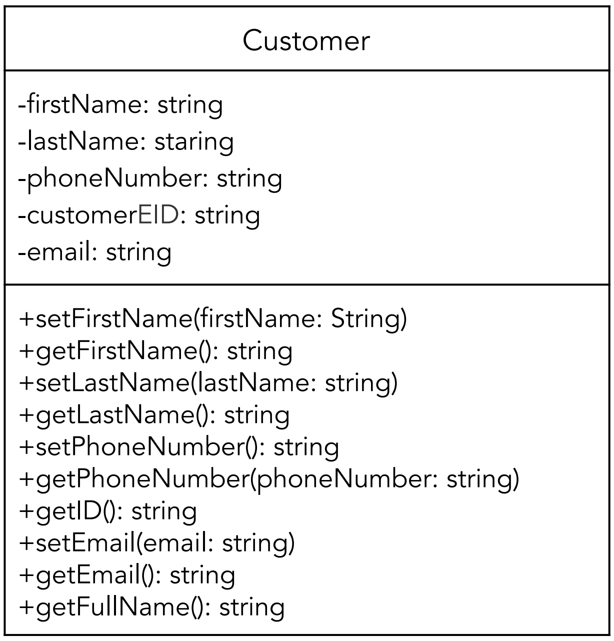
3. Service:

   Attributes:
   - name: string
   - description: string
   - duration: string
   - price: float

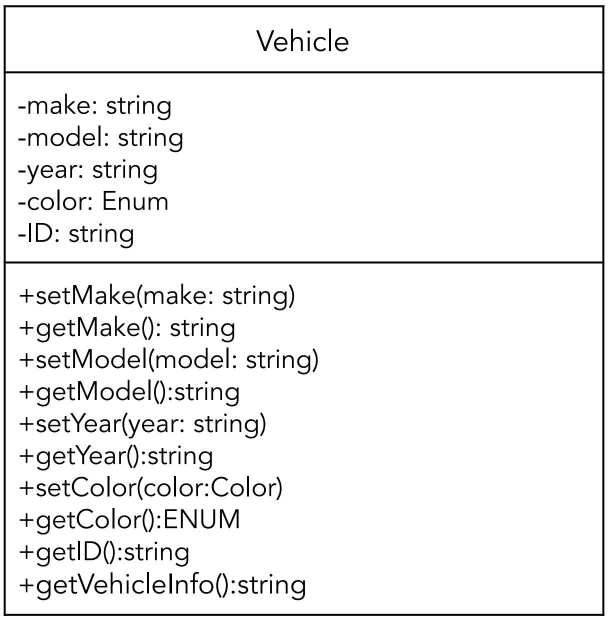
4. Receipt :

   Attributes:
   - customer : Customer
   - mechanic: string
   - vechicle: Vehicle
   - services: List[Services]
   - date: date
   - taxes: float
   - discount: float

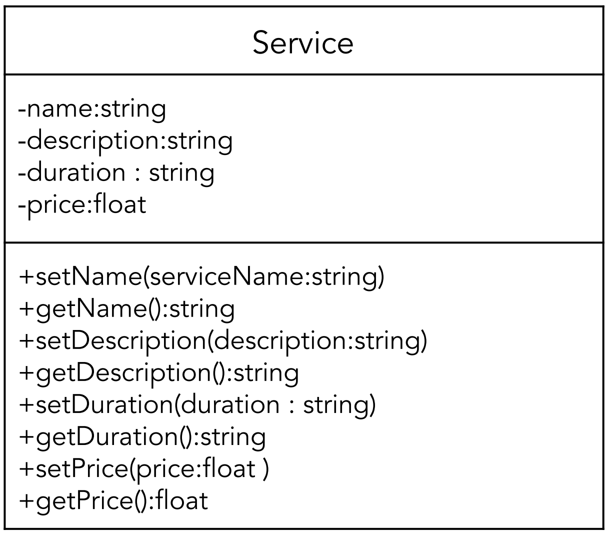
**UML Class **Diagram and Description:



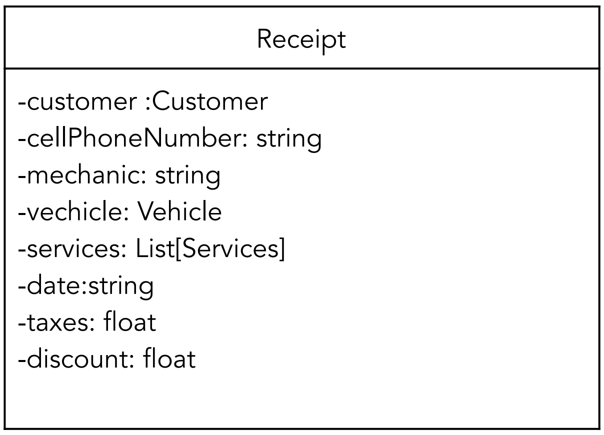
Customer
-firstName: string -lastName: staring -phoneNumber: string -customerEID: string -email: string
+setFirstName(firstName: String) +getFirstName(): string +setLastName(lastName: string) +getLastName(): string +setPhoneNumber(): string +getPhoneNumber(phoneNumber: string) +getID(): string +setEmail(email: string) +getEmail(): string +getFullName(): string



```
Vehicle
-make: string
-model: string
-year: string
-color: Enum
-ID: string
+setMake(make: string)
+getMake(): string
+setModel(model: string)
+getModel():string
+setYear(year: string)
+getYear():string
+setColor(color:Color)
+getColor():ENUM
+getID():string
+getVehicleInfo():string
```



```
Service
-name:string
-description:string
-duration : string
-price:float
+setName(serviceName:string)
+getName():string
+setDescription(description:string)
+getDescription():string
+setDuration(duration : string)
+getDuration():string
+setPrice(price:float )
+getPrice():float
```



```
Receipt
-customer :Customer
-cellPhoneNumber: string
-mechanic: string
-vechicle: Vehicle
-services: List[Services]
-date:string
-taxes: float
-discount: float
```

+getCustomer(): Customer
+getPhoneNumber(): string
+setMechanic(mechanic:string)
+getMechanic():string
+setDate(date:string)
+getDate():date
+setTaxes(taxes:float)
+getTaxes():float
+setDiscount(discount:float)
+getDiscount(): float
+Total(): float (sum(Service Price)
+Toatl+=taxes
+Total-=discount

Python code :

```
class Vehicle:
   def __init__(self,make,model,year,color,ID):
       self.__make = make
       self.__model = model
       self.__year = year
       self.__color =color
       self.__ID = ID


   def setMake(self,make):
       self.__make = make
   def getMake(self):
       return self.__make
   def setModel(self,model):
       self.__model = model
   def getModel(self):
       return self.__model
   def setYear(self,year):
       self.__year = year
   def getyear(self):
       return self.__year
   def setColor(self,color):
       self.__color=color
   def getColor(self):
       return self.__color
   def getID(self):
       return self.__ID

   def getVehicleInfo(self):
     return self.__make + " " + self.__model + " " + str(self.__year)

class Service:
   def __init__(self,name,description,duration,price):
       self.__name=name
       self.__description=description
       self.__duration=duration
       self.__price=price

   def setName(self, name):
       self.__name=name
   def getName(self):
       return self.__name
   def setDescription(self,description):
       self.__description=description
   def getDescription(self):
       return self.__dscription
   def setDuration(self,duration):
       self.__duration=duration
   def getDuration(self):
       return self.__duration
   def setPrice(self,price):
       self.__price = price
   def getPrice(self):
       return self.__price

class Receipt:
   def __init__(self,Customer,mechanic,vehicle, services, date, taxes,
discount):
       self.__customer=customer
       self.__mechainc=mechanic
       self.__vehicle=vehicle
       self.__services=services
       self.__date =date
       self.__taxes=taxes
       self.__discount=discount
   def setMechanic(self,mechanic):
       self.__mechainc = mechanic
   def getMechanic(self):
       return self.__mechainc
   def setDate(self,date):
       self.__date = date
   def getDate(self):
       return self.__date
   def setTaxes(self,taxes):
       self.__taxes=taxes
   def getTaxes(self):
       return self.__taxes
   def setDiscount(self,discount):
       self.__discount=discount
   def getDiscount(self):
       return self.__discount

   def total(self):
       total=sum(Service.getPrice() for Service in self.__services)
       total+=self.__taxes
       total-=self.__discount
       return total

   def PrintReceipt(self):
       print("Customer Name: ",self.__customer.getFullName())
       print("cell phone Number:",self.__customer.getPhoneNumber())
       print("Email: ",self.__customer.getEmail())
       print("Date:",self.__date)
       print("Mechanic Name:",self.__mechainc)
       print("Vehicle Type:",self.__vehicle.getVehicleInfo())
       print("Vehicle color: ",self.__vehicle.getColor())
       print("Vehicle ID:",self.__vehicle.getID())
       print("Services:",)

       for service in self.__services:
           print(service.getName() , "........................",
service.getPrice(),"AED",".","duration:", service.getDuration())
       print("Taxes:", self.__taxes)
       print("Discount:", self.__discount)
       print("Total:", self.total())


# Create a Customer object
customer = Customer("James", "Jones", "816-897-9862","1234567890",
"jamesjones@example.com")

# Create a Vehicle object
```

```
vehicle = Vehicle("Nissan", "Altima", 2014, Color.Silver.name,
"AD-89034")

# Create a Service object
diagnostics = Service("1.Diagnostics","looking at the care and
dlajd","5 hours", 15.0)
oilReplacement = Service("2.Oil Replacement","changing the oil ","8 h",
120.49)
oilFilterParts = Service("3.Oil Filter Parts","changing oil Filter
Parts","8 h", 35)
tireReplacement =Service("4.Tire Replacement 2 ","replace the tire ","5
h", 100)
tire= Service("5.Tire Replacement (2) ","replace the tire ","5 h", 160)


# Create a Receipt object
receipt = Receipt(customer, mechanic="Hans K", vehicle=vehicle,
services=[diagnostics,oilReplacement,oilFilterParts,tireReplacement,tir
e], date="March 13, 2022", taxes=51.5, discount=11.5)



# Print the receipt
receipt.PrintReceipt()
```

Output :

```
Customer Name:  James Jones
cell phone Number: 816-897-9862
Email:  jamesjones@example.com
Date: March 13, 2022
Mechanic Name: Hans K
Vehicle Type: Nissan Altima 2014
Vehicle color:  Silver
Vehicle ID: AD-89034
Services:
1.Diagnostics ....................... 15.0 AED . duration: 5 hours
2.Oil Replacement ....................... 120.49 AED . duration: 8 h
3.Oil Filter Parts ....................... 35 AED . duration: 8 h
4.Tire Replacement 2  ....................... 100 AED . duration: 5 h
5.Tire Replacement (2)  ....................... 160 AED . duration: 5 h
Taxes: 51.5
```

```
Discount: 11.5
Total: 470.49
```