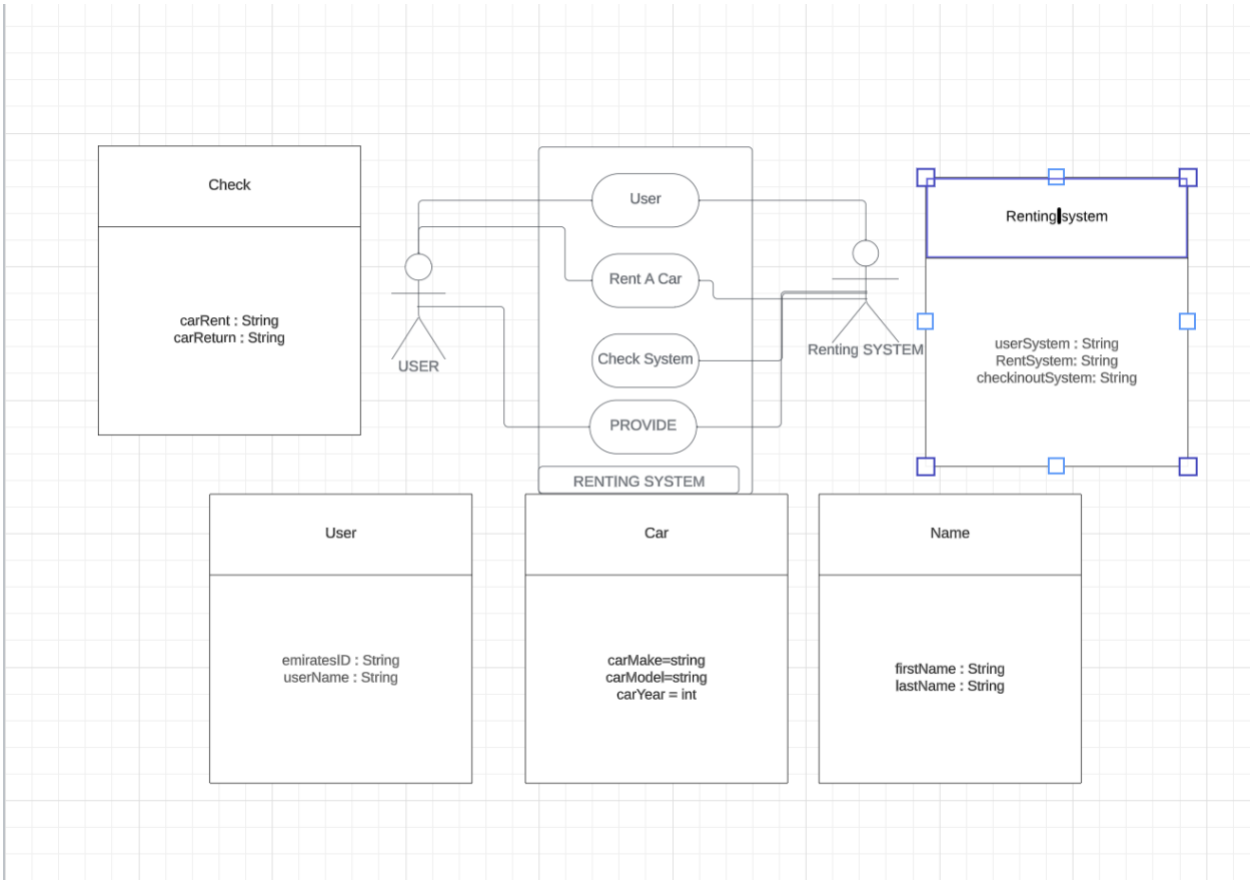
UML CASE DIAGRAM



USE CASE DESCRIPTION

Use Case:	Rent a Car
Trigger	The user wants to rent a car
Precondition	The user has a license
Main Scenario	
1	The user specifies the details of the car to which they wish they want to rent and number of days
2	The system provides the price and day of return
3	The user accepts the bargain
4	The system provides the car
Exceptions	
	The user has many accidents

	The car had an accident , user pay if he is resposible
	The amount is not enough

USER
-firstname -lastname -ID
+ setFirstname(firstName: String) +getFirstname():string + setLasttname(lastName: String) +getLastname():string + setID(ID: String) +getID():string

Car
-make -model -year
+ setCarmake(make: String) +getCarmake():string + setCarmodel(model: String) +getCarmodel():string + setCaryear(year: String) +getCaryear():string

Check
-rent -return
+ setRent(rent:Boolean) +getRent():Boolean + setReturn(Return: Boolean) +getReturn():Boolean

```
class CarRentalSystem:
    ''' Class for car rental system'''

    # Constructor
    def __init__(self):
        self._customer = CustomerInfo()
        self._car = Car()

    # Method for setting customer information
    def setCustomerInfo(self, customerID='', firstName='', lastName=''):
        self._customer.setCustomerID(customerID)
        self._customer.setName(firstName, lastName)

    # Method for setting car information
    def setCarInfo(self, make='', model='', year=0):
        self._car.setMake(make)
        self._car.setModel(model)
        self._car.setYear(year)
        self._car.setRented(False)  # Initialize car as not rented

    # Method to rent a car
    def rentCar(self):
        if not self._car.isRented():
            self._car.setRented(True)
            return "Car rented successfully."
        else:
```

```
            return "Car is already rented."

    # Method to return a rented car
    def returnCar(self):
        if self._car.isRented():
            self._car.setRented(False)
            return "Car returned successfully."
        else:
            return "Car is not currently rented."


    # Getter methods
    def getCustomerInfo(self):
        return self._customer.getCustomerID(), self._customer.getName()

    def getCarInfo(self):
        return self._car.getMake(), self._car.getModel(), self._car.getYear()


class CustomerInfo:
    def __init__(self):
        self._customerID = None
        self._firstName = None
        self._lastName = None

    def setCustomerID(self, customerID):
        self._customerID = customerID

    def getCustomerID(self):
        return self._customerID

    def setName(self, firstName, lastName):
        self._firstName = firstName
        self._lastName = lastName

    def getName(self):
        return self._firstName, self._lastName


class Car:
    def __init__(self):
        self._make = None
        self._model = None
        self._year = None
        self._rented = False
```

```
    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 setRented(self, rented):
        self._rented = rented

    def isRented(self):
        return self._rented


# Create an instance of the CarRentalSystem class
customer1 = CarRentalSystem()

# Set customer and car information
customer1.setCustomerInfo('34GHJY89', 'Hassan', 'AlAmeri')
customer1.setCarInfo('Toyota', 'Camry', 2020)

# Rent the car
print(customer1.rentCar())

# Try to rent the car again (should not be allowed)
print(customer1.rentCar())

# Return the car
print(customer1.returnCar())

# Try to return the car again (should not be allowed)
print(customer1.returnCar())

# Get and print customer and car information
customer_info = customer1.getCustomerInfo()
```

```
car_info = customer1.getCarInfo()

print("Customer Info:", customer_info)
print("Car Info:", car_info)
```