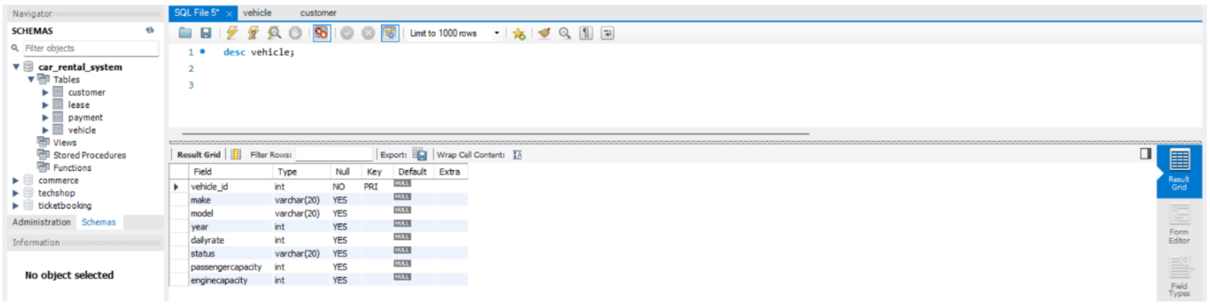
Name: R. Surya prakash

Case Study –Car Rental System

1.Create following tables in SQL Schema with appropriate class and write the unit test case for the Car Rental application.

Schema Design:

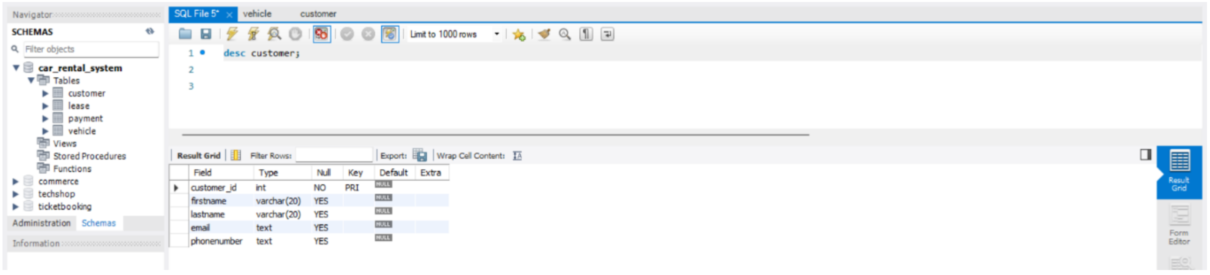
1. Vehicle Table: • vehicleID (Primary Key) • make • model • year • daily Rate • status (available, not Available) • passenger Capacity • engine Capacity



```
desc vehicle;
```

Field	Type	Null	Key	Default	Extra
vehicle_id	int	NO	PRI	NULL	
make	varchar(20)	YES		NULL	
model	varchar(20)	YES		NULL	
year	int	YES		NULL	
dailyrate	int	YES		NULL	
status	varchar(20)	YES		NULL	
passengercapacity	int	YES		NULL	
enginecapacity	int	YES		NULL	

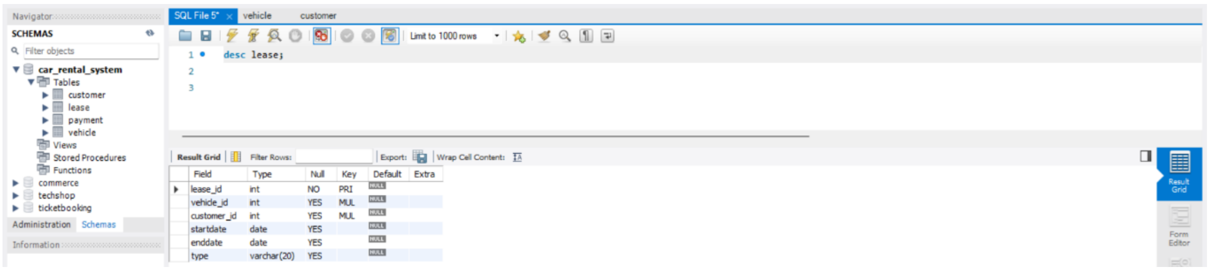
2. Customer Table: • customerID (Primary Key) • first Name • last Name • email • phone Number



```
desc customer;
```

Field	Type	Null	Key	Default	Extra
customer_id	int	NO	PRI	NULL	
firstname	varchar(20)	YES		NULL	
lastname	varchar(20)	YES		NULL	
email	text	YES		NULL	
phonenumber	text	YES		NULL	

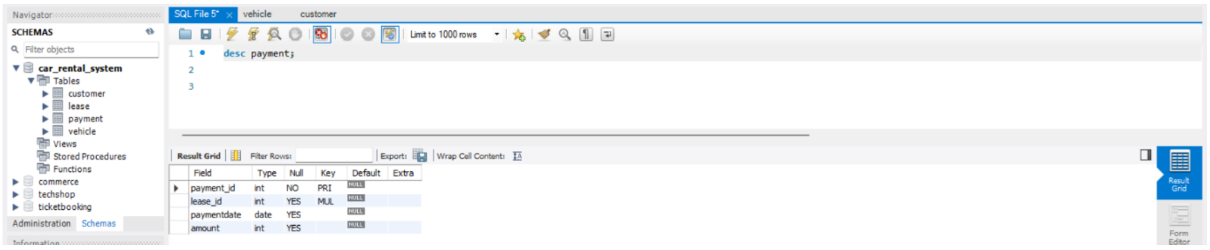
3. Lease Table: • leaseID (Primary Key) • vehicleID (Foreign Key referencing Vehicle Table) • customerID (Foreign Key referencing Customer Table) • start Date • end Date • type (to distinguish between Daily Lease and Monthly Lease)



```
desc lease;
```

Field	Type	Null	Key	Default	Extra
lease_id	int	NO	PRI	NULL	
vehicle_id	int	YES	MUL	NULL	
customer_id	int	YES	MUL	NULL	
startdate	date	YES		NULL	
enddate	date	YES		NULL	
type	varchar(20)	YES		NULL	

4. Payment Table: • payment (Primary Key) • leaseID (Foreign Key referencing Lease Table) • payment Date • amount

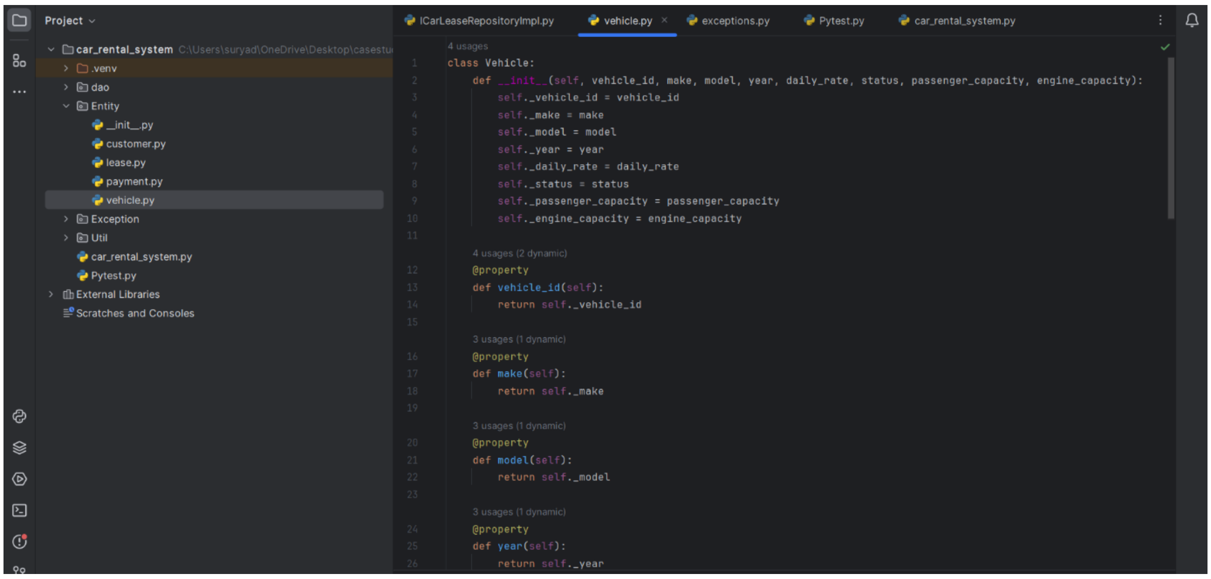


```
desc payment;
```

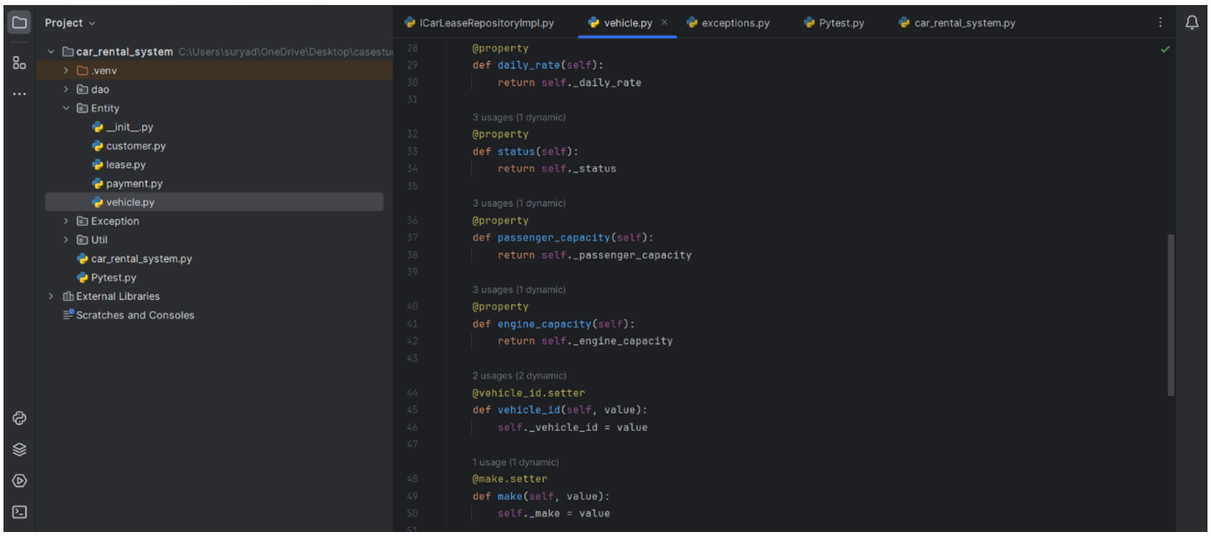
Field	Type	Null	Key	Default	Extra
payment_id	int	NO	PRI	NULL	
lease_id	int	YES	MUL	NULL	
paymentdate	date	YES		NULL	
amount	int	YES		NULL	

5. Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors (default and parametrized) and getters, setters)

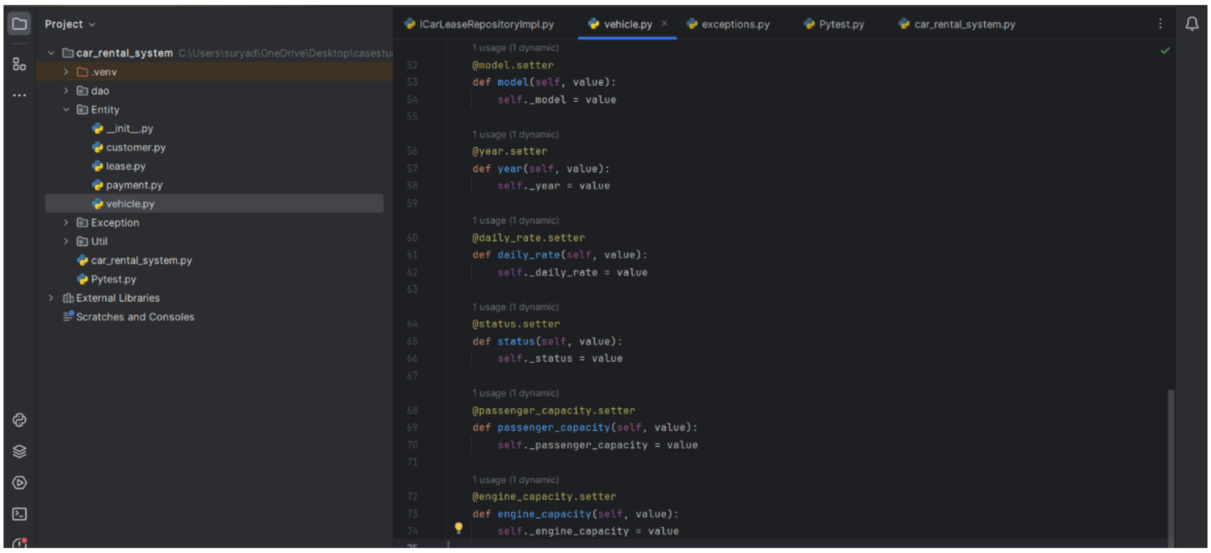
1.Vehicle:



```python
class Vehicle:
    def __init__(self, vehicle_id, make, model, year, daily_rate, status, passenger_capacity, engine_capacity):
        self._vehicle_id = vehicle_id
        self._make = make
        self._model = model
        self._year = year
        self._daily_rate = daily_rate
        self._status = status
        self._passenger_capacity = passenger_capacity
        self._engine_capacity = engine_capacity

    @property
    def vehicle_id(self):
        return self._vehicle_id

    @property
    def make(self):
        return self._make

    @property
    def model(self):
        return self._model

    @property
    def year(self):
        return self._year
```

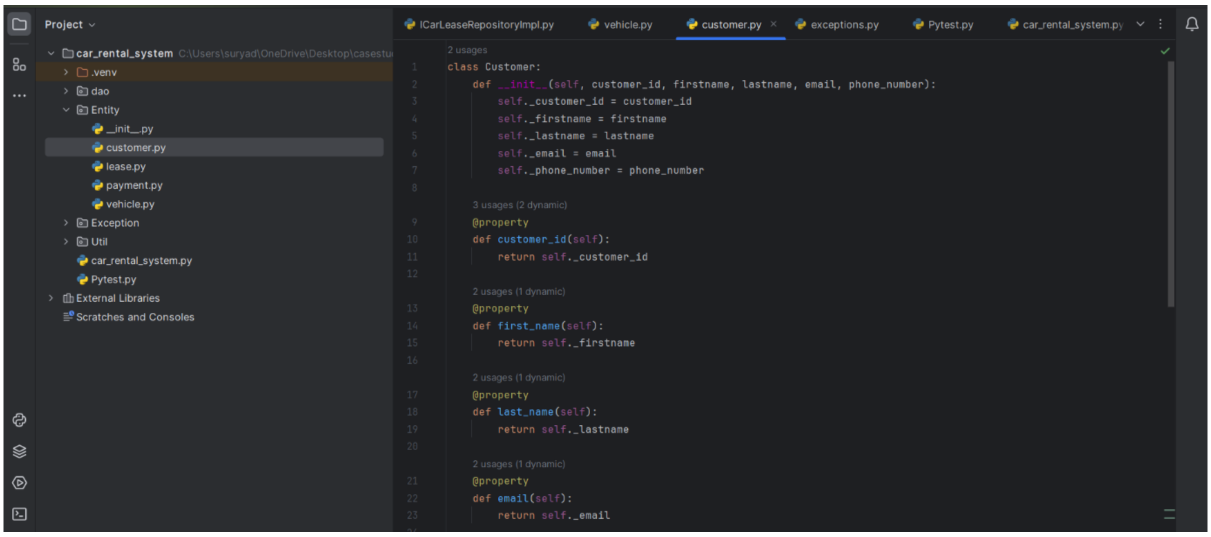


```
    @property
    def daily_rate(self):
        return self._daily_rate

    3 usages (1 dynamic)
    @property
    def status(self):
        return self._status

    3 usages (1 dynamic)
    @property
    def passenger_capacity(self):
        return self._passenger_capacity

    3 usages (1 dynamic)
    @property
    def engine_capacity(self):
        return self._engine_capacity

    2 usages (2 dynamic)
    @vehicle_id.setter
    def vehicle_id(self, value):
        self._vehicle_id = value

    1 usage (1 dynamic)
    @make.setter
    def make(self, value):
        self._make = value
```

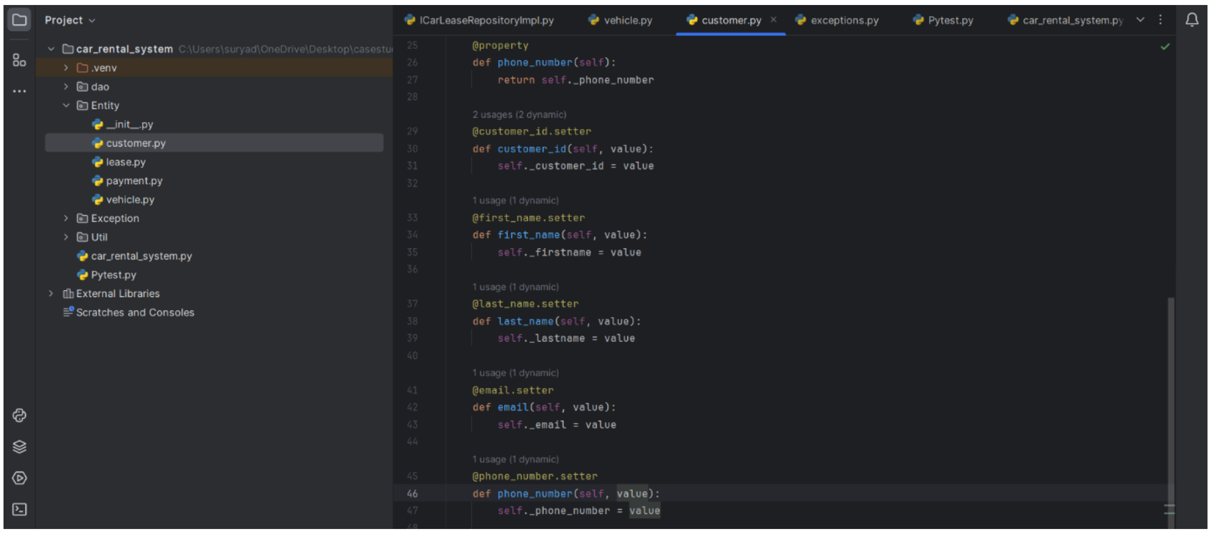


```
    1 usage (1 dynamic)
    @model.setter
    def model(self, value):
        self._model = value

    1 usage (1 dynamic)
    @year.setter
    def year(self, value):
        self._year = value

    1 usage (1 dynamic)
    @daily_rate.setter
    def daily_rate(self, value):
        self._daily_rate = value

    1 usage (1 dynamic)
    @status.setter
    def status(self, value):
        self._status = value

    1 usage (1 dynamic)
    @passenger_capacity.setter
    def passenger_capacity(self, value):
        self._passenger_capacity = value

    1 usage (1 dynamic)
    @engine_capacity.setter
    def engine_capacity(self, value):
        self._engine_capacity = value
```

2.Customer

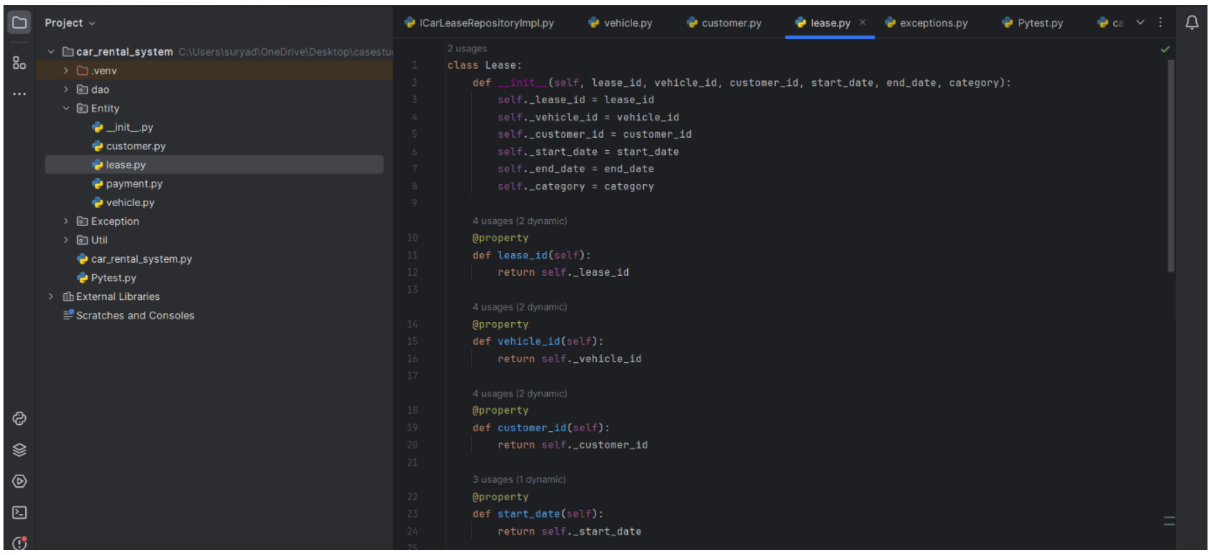


```
2 usages
class Customer:
    def __init__(self, customer_id, firstname, lastname, email, phone_number):
        self._customer_id = customer_id
        self._firstname = firstname
        self._lastname = lastname
        self._email = email
        self._phone_number = phone_number

    3 usages (2 dynamic)
    @property
    def customer_id(self):
        return self._customer_id

    2 usages (1 dynamic)
    @property
    def first_name(self):
        return self._firstname

    2 usages (1 dynamic)
    @property
    def last_name(self):
        return self._lastname

    2 usages (1 dynamic)
    @property
    def email(self):
        return self._email
```

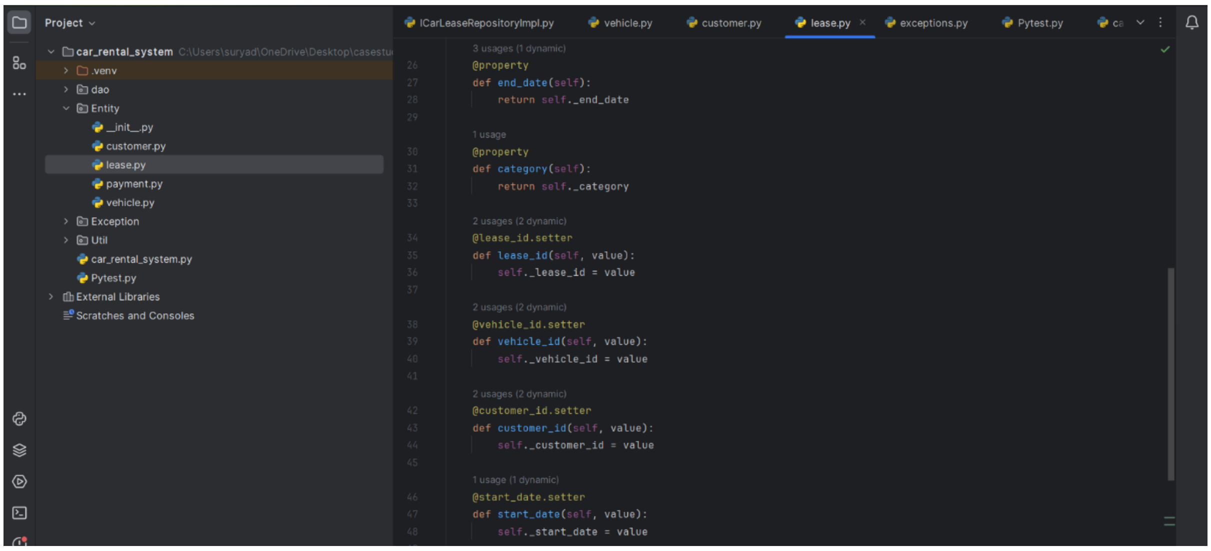


```
    @property
    def phone_number(self):
        return self._phone_number

    2 usages (2 dynamic)
    @customer_id.setter
    def customer_id(self, value):
        self._customer_id = value

    1 usage (1 dynamic)
    @first_name.setter
    def first_name(self, value):
        self._firstname = value

    1 usage (1 dynamic)
    @last_name.setter
    def last_name(self, value):
        self._lastname = value

    1 usage (1 dynamic)
    @email.setter
    def email(self, value):
        self._email = value

    1 usage (1 dynamic)
    @phone_number.setter
    def phone_number(self, value):
        self._phone_number = value
```

3.Lease

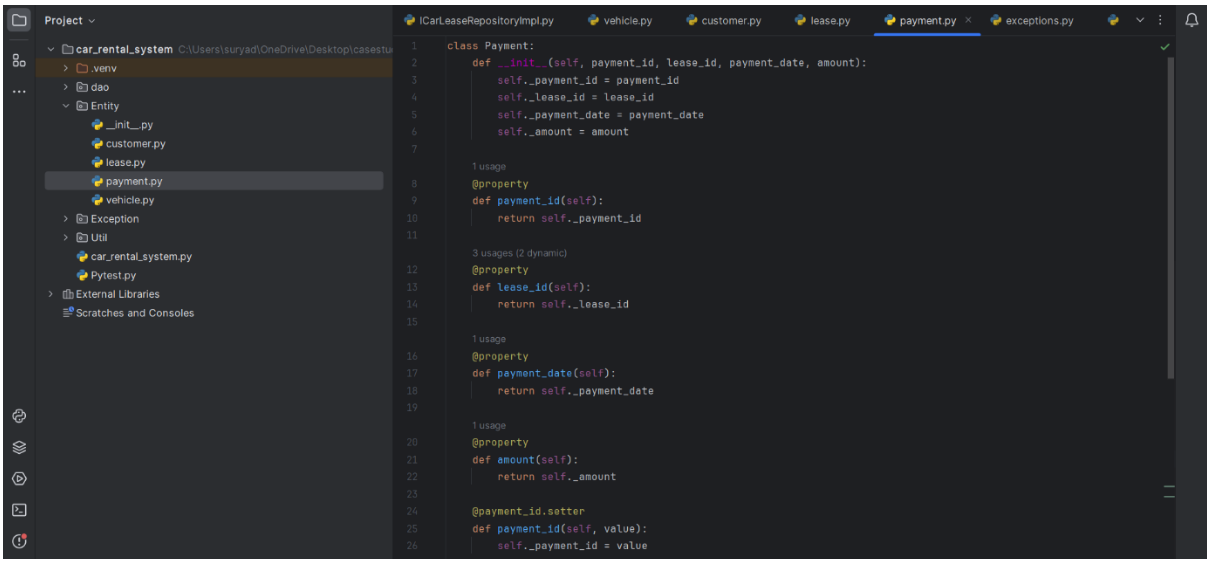


```
2 usages
class Lease:
    def __init__(self, lease_id, vehicle_id, customer_id, start_date, end_date, category):
        self._lease_id = lease_id
        self._vehicle_id = vehicle_id
        self._customer_id = customer_id
        self._start_date = start_date
        self._end_date = end_date
        self._category = category

    4 usages (2 dynamic)
    @property
    def lease_id(self):
        return self._lease_id

    4 usages (2 dynamic)
    @property
    def vehicle_id(self):
        return self._vehicle_id

    4 usages (2 dynamic)
    @property
    def customer_id(self):
        return self._customer_id

    3 usages (1 dynamic)
    @property
    def start_date(self):
        return self._start_date
```

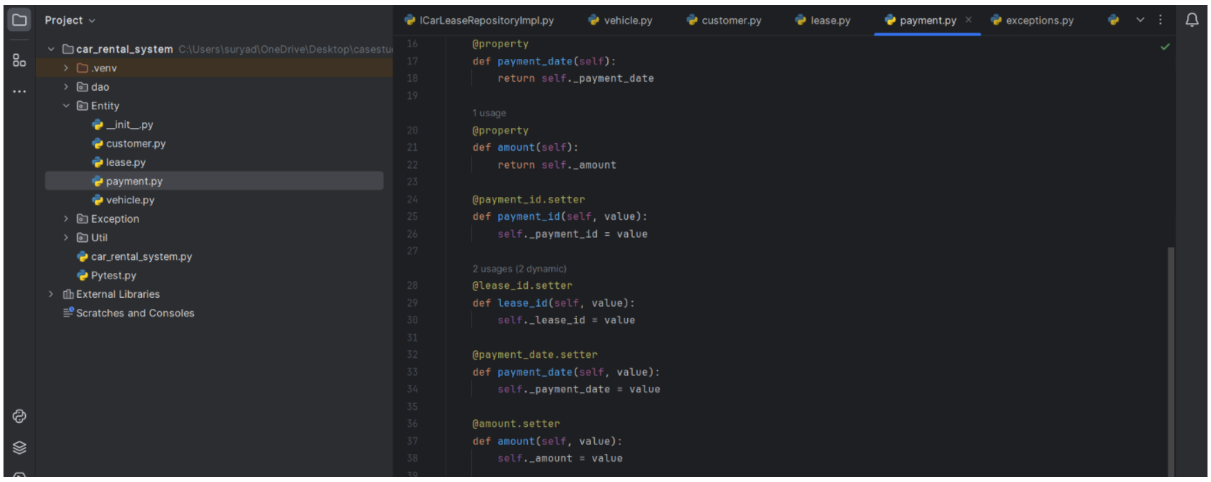


```
    3 usages (1 dynamic)
    @property
    def end_date(self):
        return self._end_date

    1 usage
    @property
    def category(self):
        return self._category

    2 usages (2 dynamic)
    @lease_id.setter
    def lease_id(self, value):
        self._lease_id = value

    2 usages (2 dynamic)
    @vehicle_id.setter
    def vehicle_id(self, value):
        self._vehicle_id = value

    2 usages (2 dynamic)
    @customer_id.setter
    def customer_id(self, value):
        self._customer_id = value

    1 usage (1 dynamic)
    @start_date.setter
    def start_date(self, value):
        self._start_date = value
```

4.Payment

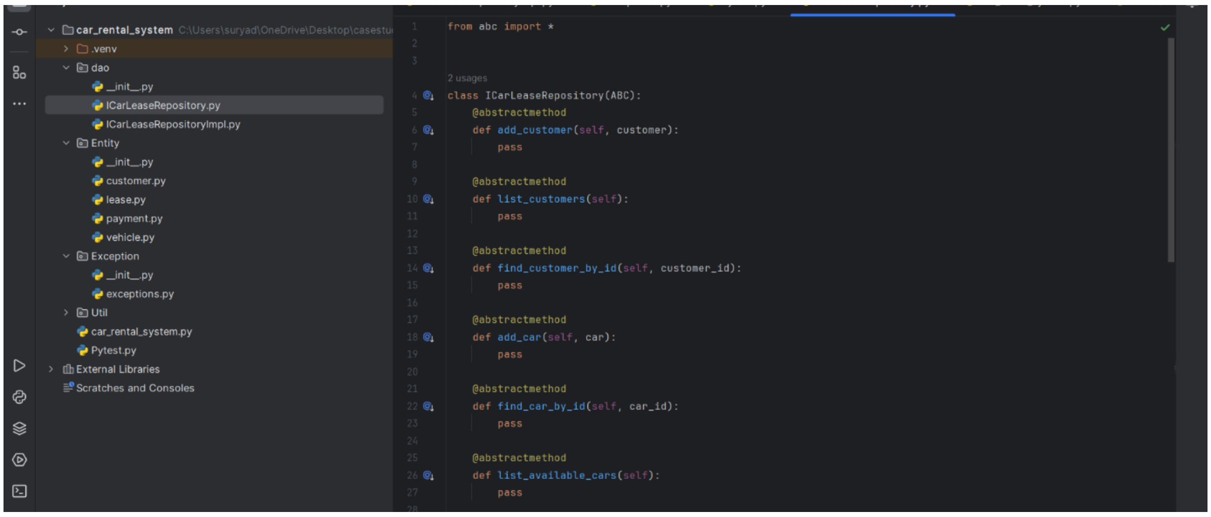


```python
class Payment:
    def __init__(self, payment_id, lease_id, payment_date, amount):
        self._payment_id = payment_id
        self._lease_id = lease_id
        self._payment_date = payment_date
        self._amount = amount

    1 usage
    @property
    def payment_id(self):
        return self._payment_id

    3 usages (2 dynamic)
    @property
    def lease_id(self):
        return self._lease_id

    1 usage
    @property
    def payment_date(self):
        return self._payment_date

    1 usage
    @property
    def amount(self):
        return self._amount

    @payment_id.setter
    def payment_id(self, value):
        self._payment_id = value
```

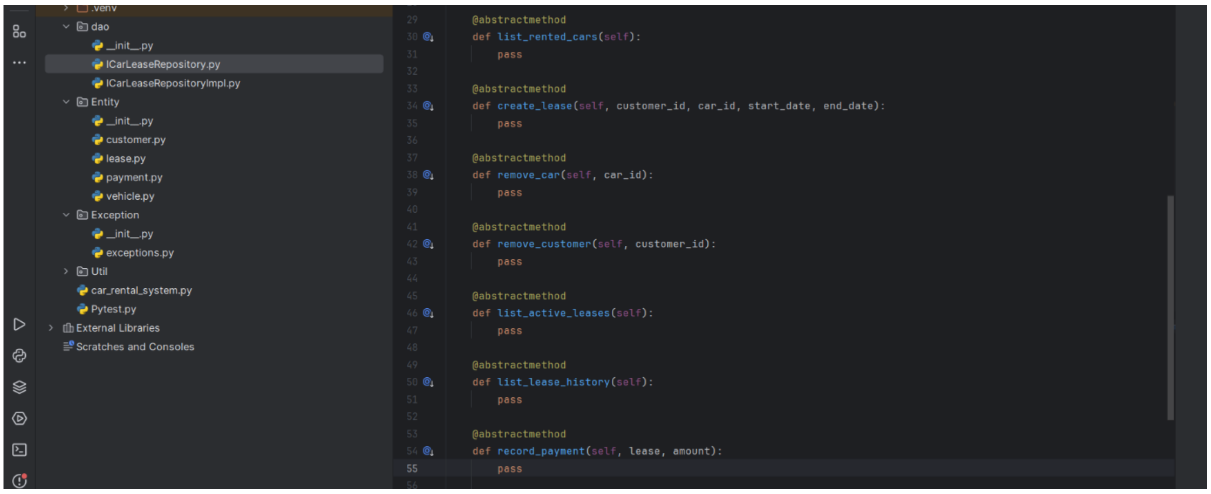


```python
    @property
    def payment_date(self):
        return self._payment_date

    1 usage
    @property
    def amount(self):
        return self._amount

    @payment_id.setter
    def payment_id(self, value):
        self._payment_id = value

    2 usages (2 dynamic)
    @lease_id.setter
    def lease_id(self, value):
        self._lease_id = value

    @payment_date.setter
    def payment_date(self, value):
        self._payment_date = value

    @amount.setter
    def amount(self, value):
        self._amount = value
```

Interface for ICarLeaseRepository:

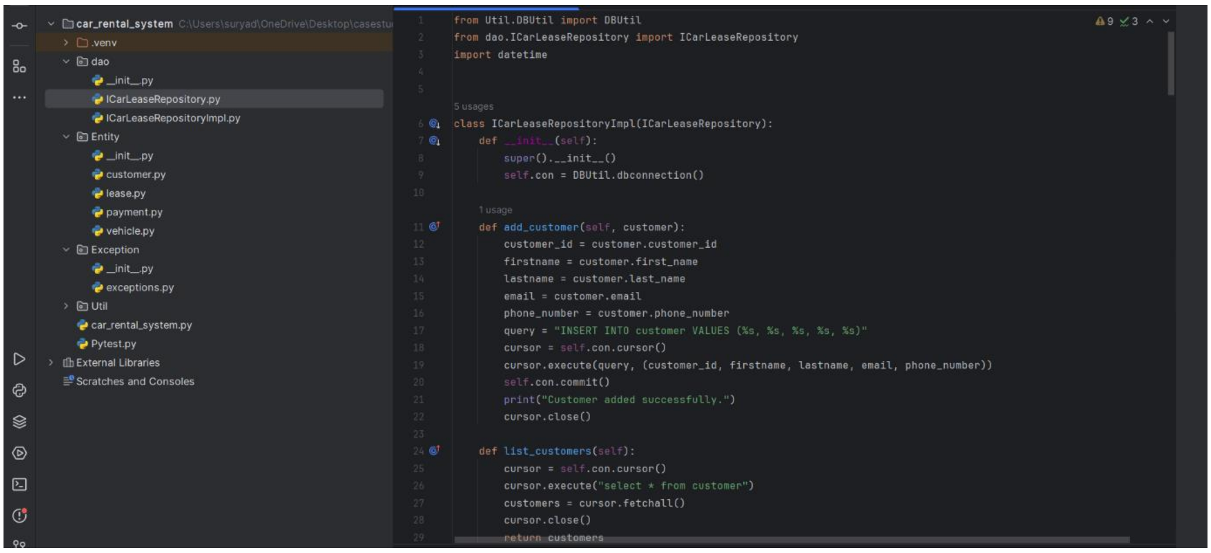


```python
from abc import *


2 usages
class ICarLeaseRepository(ABC):
    @abstractmethod
    def add_customer(self, customer):
        pass

    @abstractmethod
    def list_customers(self):
        pass

    @abstractmethod
    def find_customer_by_id(self, customer_id):
        pass

    @abstractmethod
    def add_car(self, car):
        pass

    @abstractmethod
    def find_car_by_id(self, car_id):
        pass

    @abstractmethod
    def list_available_cars(self):
        pass
```

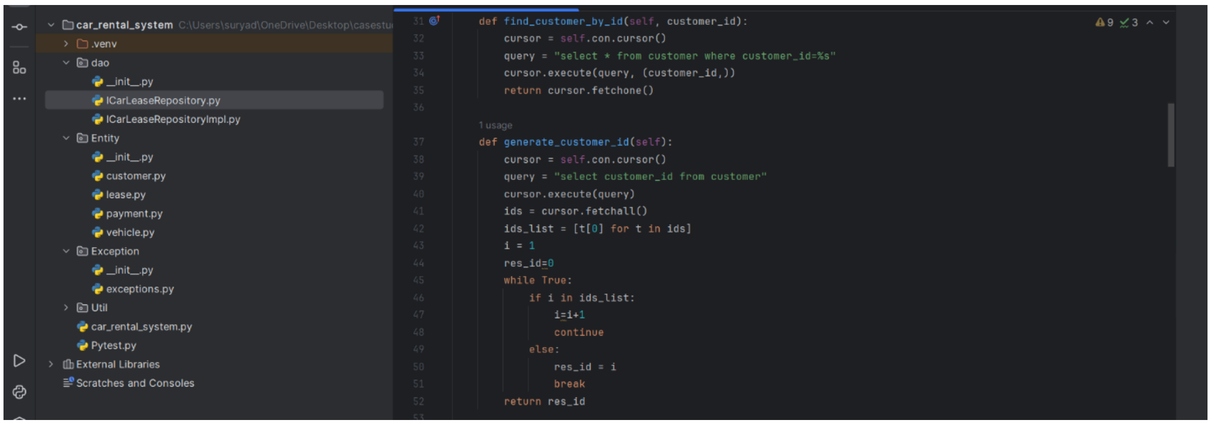


```python
    @abstractmethod
    def list_rented_cars(self):
        pass

    @abstractmethod
    def create_lease(self, customer_id, car_id, start_date, end_date):
        pass

    @abstractmethod
    def remove_car(self, car_id):
        pass

    @abstractmethod
    def remove_customer(self, customer_id):
        pass

    @abstractmethod
    def list_active_leases(self):
        pass

    @abstractmethod
    def list_lease_history(self):
        pass

    @abstractmethod
    def record_payment(self, lease, amount):
        pass
```

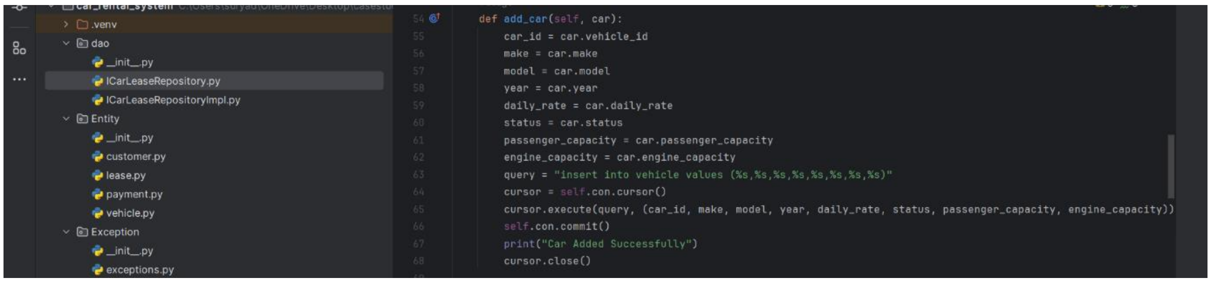
ICarLeaseRepositoryImpl:



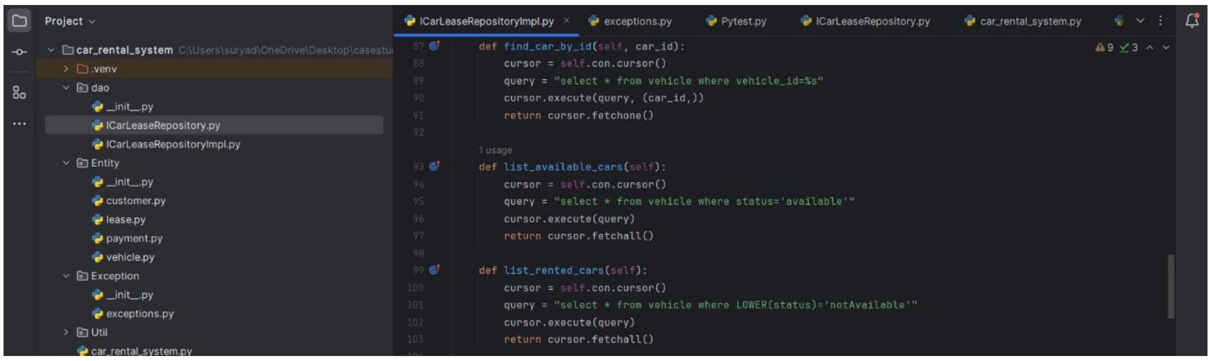
```python
from Util.DBUtil import DBUtil
from dao.ICarLeaseRepository import ICarLeaseRepository
import datetime


5 usages
class ICarLeaseRepositoryImpl(ICarLeaseRepository):
    def __init__(self):
        super().__init__()
        self.con = DBUtil.dbconnection()

    1 usage
    def add_customer(self, customer):
        customer_id = customer.customer_id
        firstname = customer.first_name
        lastname = customer.last_name
        email = customer.email
        phone_number = customer.phone_number
        query = "INSERT INTO customer VALUES (%s, %s, %s, %s, %s)"
        cursor = self.con.cursor()
        cursor.execute(query, (customer_id, firstname, lastname, email, phone_number))
        self.con.commit()
        print("Customer added successfully.")
        cursor.close()

    def list_customers(self):
        cursor = self.con.cursor()
        cursor.execute("select * from customer")
        customers = cursor.fetchall()
        cursor.close()
        return customers
```



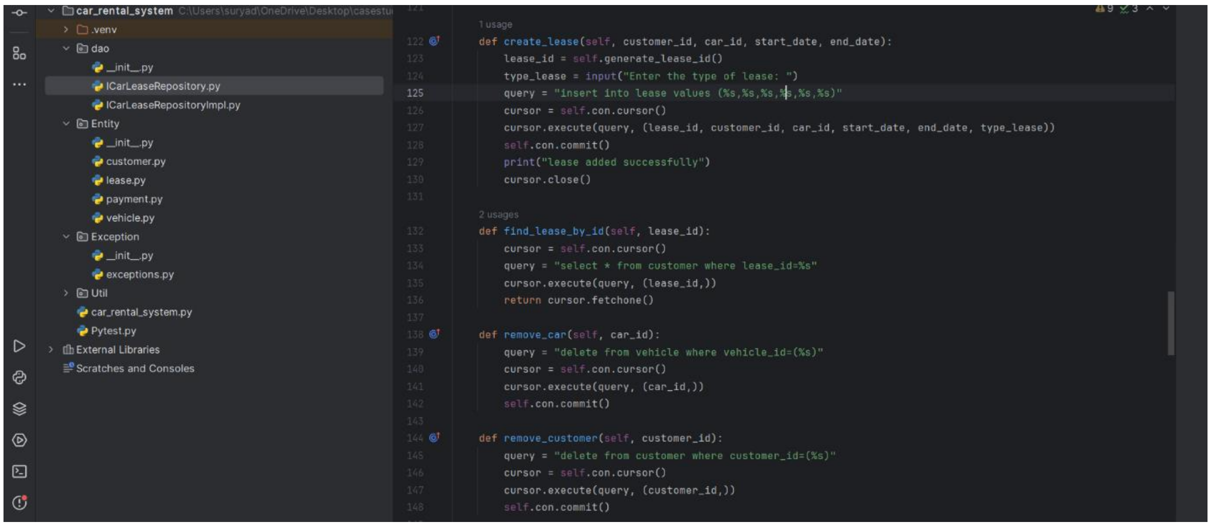
```python
    def find_customer_by_id(self, customer_id):
        cursor = self.con.cursor()
        query = "select * from customer where customer_id=%s"
        cursor.execute(query, (customer_id,))
        return cursor.fetchone()

    def generate_customer_id(self):
        cursor = self.con.cursor()
        query = "select customer_id from customer"
        cursor.execute(query)
        ids = cursor.fetchall()
        ids_list = [t[0] for t in ids]
        i = 1
        res_id=0
        while True:
            if i in ids_list:
                i=i+1
                continue
            else:
                res_id = i
                break
        return res_id
```



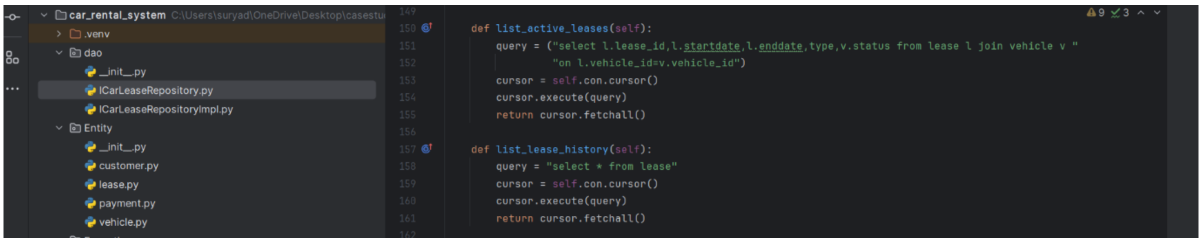
```python
    def add_car(self, car):
        car_id = car.vehicle_id
        make = car.make
        model = car.model
        year = car.year
        daily_rate = car.daily_rate
        status = car.status
        passenger_capacity = car.passenger_capacity
        engine_capacity = car.engine_capacity
        query = "insert into vehicle values (%s,%s,%s,%s,%s,%s,%s,%s)"
        cursor = self.con.cursor()
        cursor.execute(query, (car_id, make, model, year, daily_rate, status, passenger_capacity, engine_capacity))
        self.con.commit()
        print("Car Added Successfully")
        cursor.close()
```



```python
    def find_car_by_id(self, car_id):
        cursor = self.con.cursor()
        query = "select * from vehicle where vehicle_id=%s"
        cursor.execute(query, (car_id,))
        return cursor.fetchone()

    def list_available_cars(self):
        cursor = self.con.cursor()
        query = "select * from vehicle where status='available'"
        cursor.execute(query)
        return cursor.fetchall()

    def list_rented_cars(self):
        cursor = self.con.cursor()
        query = "select * from vehicle where LOWER(status)='notAvailable'"
        cursor.execute(query)
        return cursor.fetchall()
```

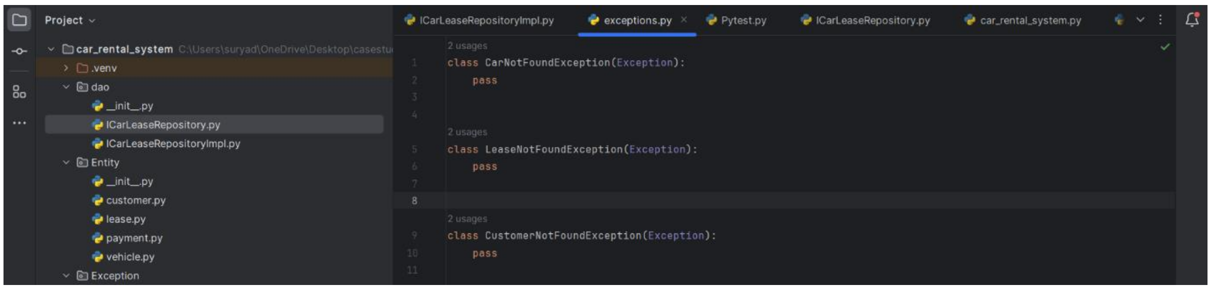


```python
    1 usage
    def create_lease(self, customer_id, car_id, start_date, end_date):
        lease_id = self.generate_lease_id()
        type_lease = input("Enter the type of lease: ")
        query = "insert into lease values (%s,%s,%s,%s,%s,%s)"
        cursor = self.con.cursor()
        cursor.execute(query, (lease_id, customer_id, car_id, start_date, end_date, type_lease))
        self.con.commit()
        print("lease added successfully")
        cursor.close()

    2 usages
    def find_lease_by_id(self, lease_id):
        cursor = self.con.cursor()
        query = "select * from customer where lease_id=%s"
        cursor.execute(query, (lease_id,))
        return cursor.fetchone()

    def remove_car(self, car_id):
        query = "delete from vehicle where vehicle_id=(%s)"
        cursor = self.con.cursor()
        cursor.execute(query, (car_id,))
        self.con.commit()

    def remove_customer(self, customer_id):
        query = "delete from customer where customer_id=(%s)"
        cursor = self.con.cursor()
        cursor.execute(query, (customer_id,))
        self.con.commit()
```



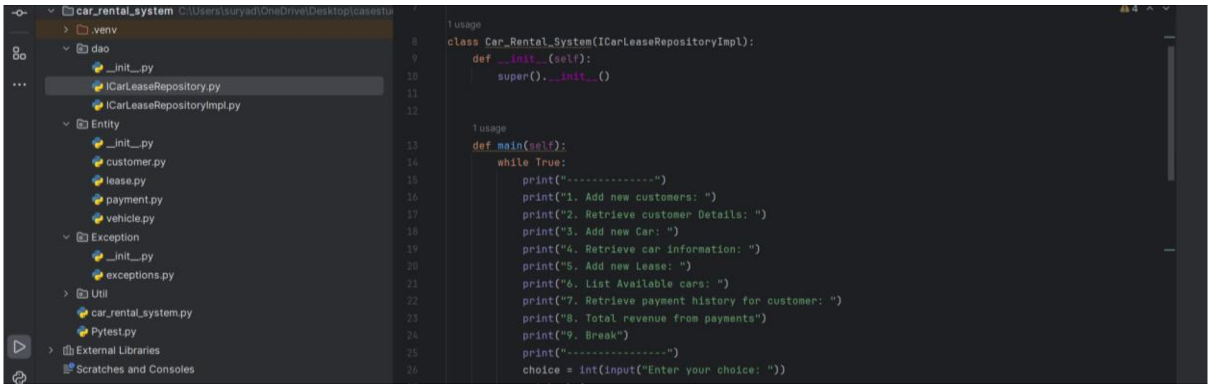
```python
    def list_active_leases(self):
        query = ("select l.lease_id,l.startdate,l.enddate,type,v.status from lease l join vehicle v "
                 "on l.vehicle_id=v.vehicle_id")
        cursor = self.con.cursor()
        cursor.execute(query)
        return cursor.fetchall()

    def list_lease_history(self):
        query = "select * from lease"
        cursor = self.con.cursor()
        cursor.execute(query)
        return cursor.fetchall()
```

Exceptions:

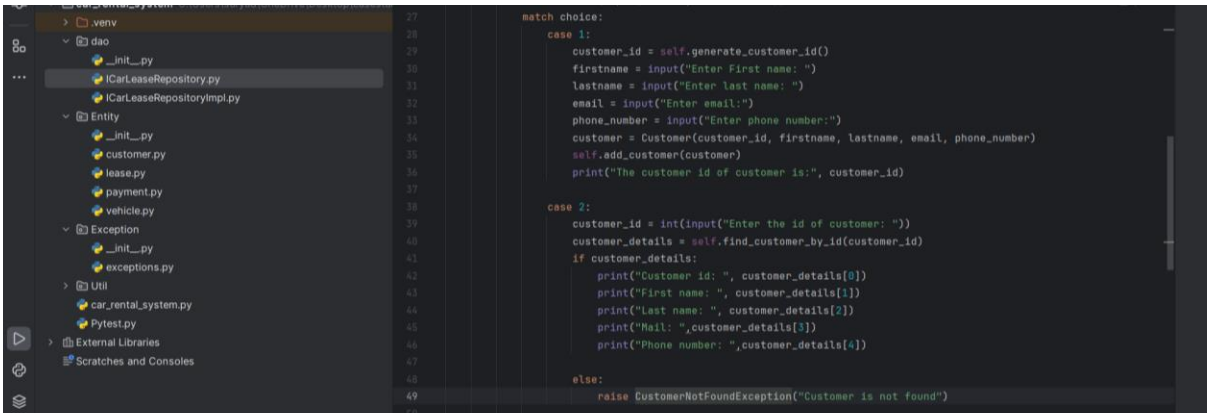


```python
2 usages
class CarNotFoundException(Exception):
    pass


2 usages
class LeaseNotFoundException(Exception):
    pass


2 usages
class CustomerNotFoundException(Exception):
    pass
```

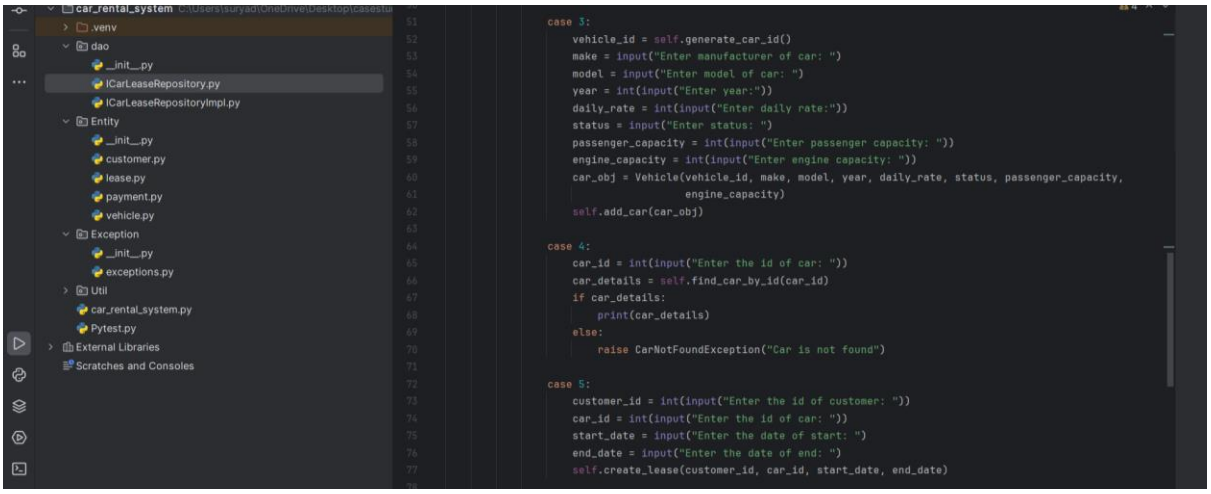
Main Module:



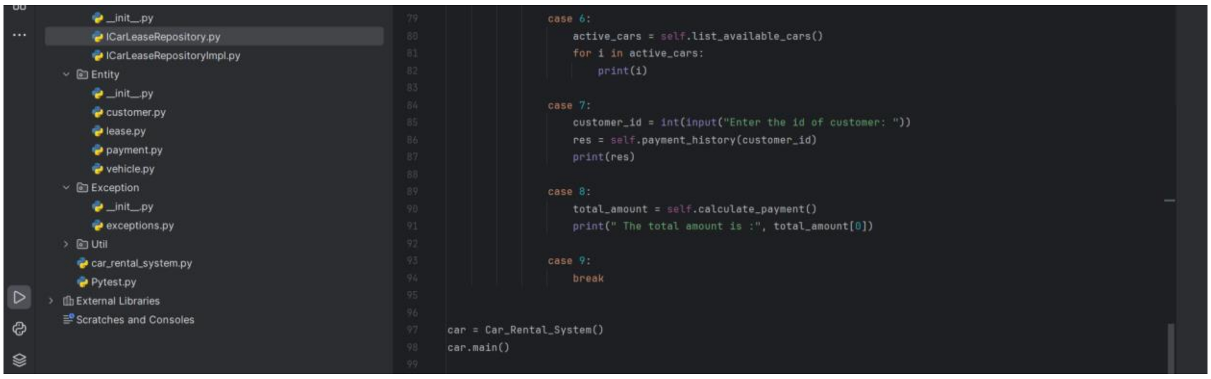
```python
1 usage
class Car_Rental_System(ICarLeaseRepositoryImpl):
    def __init__(self):
        super().__init__()


    1 usage
    def main(self):
        while True:
            print("-------------")
            print("1. Add new customers: ")
            print("2. Retrieve customer Details: ")
            print("3. Add new Car: ")
            print("4. Retrieve car information: ")
            print("5. Add new Lease: ")
            print("6. List Available cars: ")
            print("7. Retrieve payment history for customer: ")
            print("8. Total revenue from payments")
            print("9. Break")
            print("---------------")
            choice = int(input("Enter your choice: "))
```



```python
            match choice:
                case 1:
                    customer_id = self.generate_customer_id()
                    firstname = input("Enter First name: ")
                    lastname = input("Enter last name: ")
                    email = input("Enter email:")
                    phone_number = input("Enter phone number:")
                    customer = Customer(customer_id, firstname, lastname, email, phone_number)
                    self.add_customer(customer)
                    print("The customer id of customer is:", customer_id)

                case 2:
                    customer_id = int(input("Enter the id of customer: "))
                    customer_details = self.find_customer_by_id(customer_id)
                    if customer_details:
                        print("Customer id: ", customer_details[0])
                        print("First name: ", customer_details[1])
                        print("Last name: ", customer_details[2])
                        print("Mail: ",customer_details[3])
                        print("Phone number: ",customer_details[4])

                    else:
                        raise CustomerNotFoundException("Customer is not found")
```

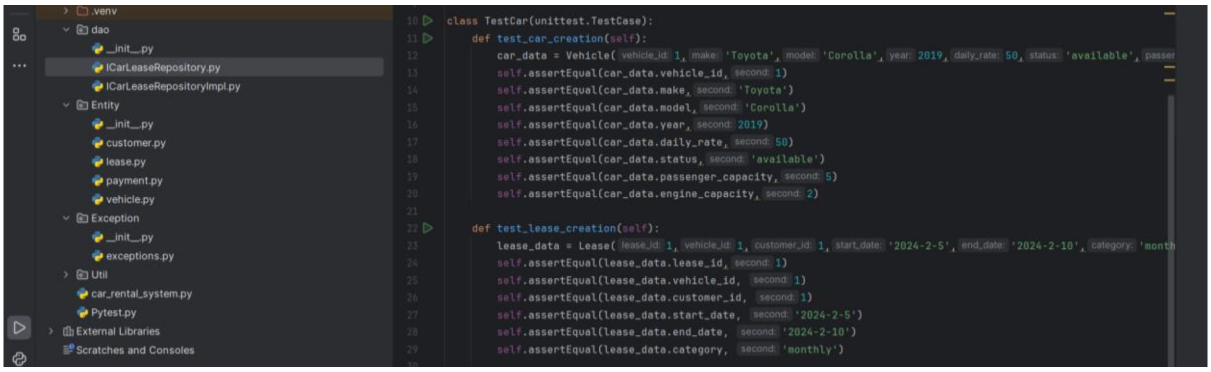


```python
                case 3:
                    vehicle_id = self.generate_car_id()
                    make = input("Enter manufacturer of car: ")
                    model = input("Enter model of car: ")
                    year = int(input("Enter year:"))
                    daily_rate = int(input("Enter daily rate:"))
                    status = input("Enter status: ")
                    passenger_capacity = int(input("Enter passenger capacity: "))
                    engine_capacity = int(input("Enter engine capacity: "))
                    car_obj = Vehicle(vehicle_id, make, model, year, daily_rate, status, passenger_capacity,
                                      engine_capacity)
                    self.add_car(car_obj)

                case 4:
                    car_id = int(input("Enter the id of car: "))
                    car_details = self.find_car_by_id(car_id)
                    if car_details:
                        print(car_details)
                    else:
                        raise CarNotFoundException("Car is not found")

                case 5:
                    customer_id = int(input("Enter the id of customer: "))
                    car_id = int(input("Enter the id of car: "))
                    start_date = input("Enter the date of start: ")
                    end_date = input("Enter the date of end: ")
                    self.create_lease(customer_id, car_id, start_date, end_date)
```



```python
                case 6:
                    active_cars = self.list_available_cars()
                    for i in active_cars:
                        print(i)

                case 7:
                    customer_id = int(input("Enter the id of customer: "))
                    res = self.payment_history(customer_id)
                    print(res)

                case 8:
                    total_amount = self.calculate_payment()
                    print(" The total amount is :", total_amount[0])

                case 9:
                    break


car = Car_Rental_System()
car.main()
```

Pytest:



```python
class TestCar(unittest.TestCase):
    def test_car_creation(self):
        car_data = Vehicle( vehicle_id: 1, make: 'Toyota', model: 'Corolla', year: 2019, daily_rate: 50, status: 'available', passer
        self.assertEqual(car_data.vehicle_id, second: 1)
        self.assertEqual(car_data.make, second: 'Toyota')
        self.assertEqual(car_data.model, second: 'Corolla')
        self.assertEqual(car_data.year, second: 2019)
        self.assertEqual(car_data.daily_rate, second: 50)
        self.assertEqual(car_data.status, second: 'available')
        self.assertEqual(car_data.passenger_capacity, second: 5)
        self.assertEqual(car_data.engine_capacity, second: 2)

    def test_lease_creation(self):
        lease_data = Lease( lease_id: 1, vehicle_id: 1, customer_id: 1, start_date: '2024-2-5', end_date: '2024-2-10', category: 'month
        self.assertEqual(lease_data.lease_id, second: 1)
        self.assertEqual(lease_data.vehicle_id,  second: 1)
        self.assertEqual(lease_data.customer_id,  second: 1)
        self.assertEqual(lease_data.start_date,  second: '2024-2-5')
        self.assertEqual(lease_data.end_date,  second: '2024-2-10')
        self.assertEqual(lease_data.category,  second: 'monthly')
```