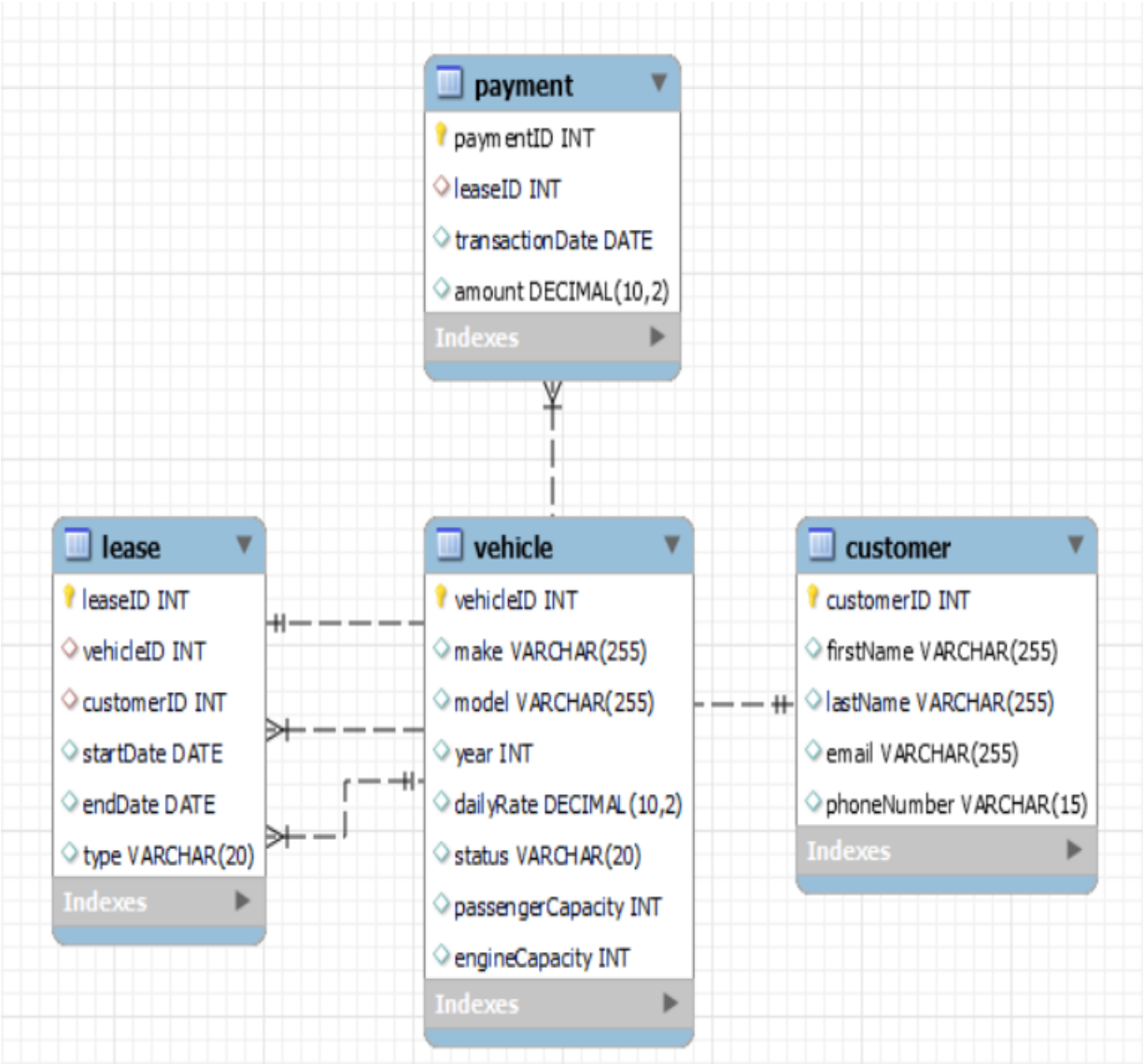
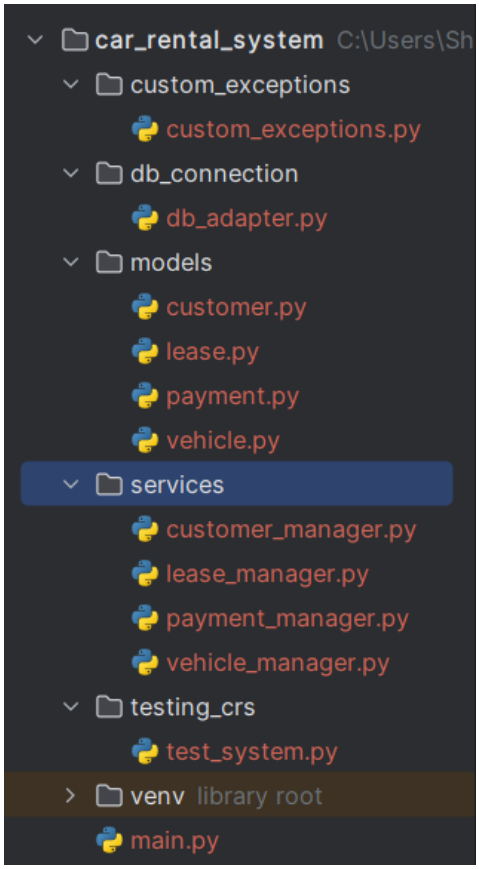
Case Study - Car Rental System

ERD Diagram



Folder Structure



Custom_exceptions.py

custom_exceptions.py

```
4 usages
class VehicleNotFoundException(Exception):
    pass


4 usages
class LeaseNotFoundException(Exception):
    pass


6 usages
class CustomerNotFoundException(Exception):
    pass

```

Db_adapter.py

```
import mysql.connector
def get_db_connection():
    # Replace the following values with your MySQL server credentials
    config = {
        'user': 'root',
        'password': 'prakhar123',
        'host': 'localhost',
        'database': 'crsdb'
    }

    try:
        connection = mysql.connector.connect(**config)
        # print("Connected to the database")
        # print('hello World')
        return connection
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None


def get_ids(table_name, id_column_name):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT ' + id_column_name + ' FROM ' + table_name + ' ORDER BY ' +
id_column_name + ' DESC LIMIT 1'
    # print(sql)
    my_cursor.execute(sql)
```

```
    x = list(my_cursor.fetchone())[0]
    return int(x) + 1


def get_cnts(table_name, id_column_name, column_id):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name + '=' + column_id
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x)

def get_counts(table_name, id_column_name1, id_column_name2, column_id1,
column_id2):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT count(*) as count FROM ' + table_name + ' WHERE ' +
id_column_name1 + '=' + column_id1 + ' AND ' + id_column_name2 + '=' + '"' +
str(column_id2) + '"'
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    # print(x)
    return int(x)

def get_data(sql, para):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    my_cursor.execute(sql, para)
    x = [list(i) for i in my_cursor.fetchall()]
    return x
```

Customer.py

```
from db_connection.db_adapter import *

class Customer:

    def __init__(self, customer_id, first_name, last_name, email,
phone_number):
        self.connection = get_db_connection()
        self.customer_id = customer_id
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.phone_number = phone_number
```

```
    def __str__(self):
        return f"Customer ID: {self.customer_id}\n" \
               f"First Name: {self.first_name}\n" \
               f"Last Name: {self.last_name}\n" \
               f"Email: {self.email}\n" \
               f"Phone Number: {self.phone_number}"
```

Payment.py

```
from db_connection.db_adapter import *


class Payment:

    def __init(self, payment_id, lease_id, transaction_date, amount):
        self.connection = get_db_connection()
        self.payment_id = payment_id
        self.lease_id = lease_id
        self.transaction_date = transaction_date
        self.amount = amount

    def __str__(self):
        return f"Payment ID: {self.payment_id}\n" \
               f"Lease ID: {self.lease_id}\n" \
               f"Transaction Date: {self.transaction_date}\n" \
               f"Amount: {self.amount}"
```

Lease.py

```
from db_connection.db_adapter import *


class Lease:

    def __init__(self, lease_id, vehicle_id, customer_id, start_date, end_date,
type):
        self.connection = get_db_connection()
        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.type = type
```

```
    def __str__(self):
        return f"Lease ID: {self.lease_id}\n" \
               f"Vehicle ID: {self.vehicle_id}\n" \
               f"Customer ID: {self.customer_id}\n" \
               f"Start Date: {self.start_date}\n" \
               f"End Date: {self.end_date}\n" \
               f"Type: {self.type}"
```

Vehicle.py

```
from db_connection.db_adapter import *


class Vehicle:

    def __init__(self, vehicle_id, make, model, year, daily_rate, status,
passenger_capacity, engine_capacity):
        self.connection = get_db_connection()
        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

    def __str__(self):
        return f"Vehicle ID: {self.vehicle_id}\n" \
               f"Make: {self.make}\n" \
               f"Model: {self.model}\n" \
               f"Year: {self.year}\n" \
               f"Daily Rate: {self.daily_rate}\n" \
               f"Status: {self.status}\n" \
               f"Passenger Capacity: {self.passenger_capacity}\n" \
               f"Engine Capacity: {self.engine_capacity}"
```

services/payment_manager.py

```
from db_connection.db_adapter import *
from datetime import date

class PaymentManager:
    def __init__(self):
        self.connection = get_db_connection()

    def record_payment(self, lease_id, amount):
```

```
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            INSERT INTO Payment(paymentID, leaseID, transactionDate, amount)
            VALUES (%s, %s, %s, %s)
            '''
            para = (get_ids('payment', 'paymentID') ,lease_id, date.today(),
amount)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Payment recorded successfully.')

        except Exception as e:
            print('An error occurred: ', e)
```

services/customer_manager.py

```
from db_connection.db_adapter import *
from models.customer import Customer
from custom_exceptions.custom_exceptions import *


class CustomerManager:

    def __init__(self):
        self.connection = get_db_connection()

    def add_new_customer(self, first_name, last_name, email, phone_number):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            INSERT INTO Customer(customerID, firstName, lastName, email,
phoneNumber)
            VALUES (%s, %s, %s, %s, %s)
            '''
            para = (get_ids('customer', 'customerID'), first_name, last_name,
email, phone_number)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Customer Added Successfully')
        except Exception as e:
            print('An error occurred: ', e)

    def update_customer_info(self, customer_id, first_name=None,
last_name=None, email=None, phone_number=None):
        try:
            my_cursor = self.connection.cursor()
```

```
            if first_name:
                sql = '''
                UPDATE Customer SET firstName = %s WHERE customerID = %s
                '''
                para = (first_name, customer_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('First Name updated successfully')

            if last_name:
                sql = '''
                UPDATE Customer SET lastName = %s WHERE customerID = %s
                '''
                para = (last_name, customer_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('Last Name updated successfully')

            if email:
                sql = '''
                UPDATE Customer SET email = %s WHERE customerID = %s
                '''
                para = (email, customer_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('Email updated successfully')

            if phone_number:
                sql = '''
                UPDATE Customer SET phoneNumber = %s WHERE customerID = %s
                '''
                para = (phone_number, customer_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('Phone Number updated successfully')

            print('Customer Details updated successfully')

        except Exception as e:
            print('An error occurred: ', e)

    def get_customer_details(self, customer_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * FROM Customer WHERE customerID = %s
            '''
            para = (customer_id,)
            my_cursor.execute(sql, para)
```

```
            x = list(my_cursor.fetchone())
            print(*x, sep=',')
        except Exception as e:
            print('An error occurred: ', {e})

    def get_customer_by_id(self, customer_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * FROM Customer WHERE customerID = %s
            '''
            para = (customer_id,)
            my_cursor.execute(sql, para)
            temp = my_cursor.fetchone()

            if temp is None:
                raise CustomerNotFoundException('Invalid Customer ID')
            x = Customer(*temp)
            return x
        except CustomerNotFoundException as cnfe:
            raise cnfe
        except Exception as e:
            print('An error occurred: ', {e})

    def list_customers(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * FROM Customer
            '''
            my_cursor.execute(sql)
            x = [list(i) for i in my_cursor.fetchall()]
            print(*x, sep='\n')
        except Exception as e:
            print('An error occurred: ', {e})

    def remove_customer(self, customer_id):
        try:
            my_cursor = self.connection.cursor()
            x = get_cnts('customer', 'customerID', customer_id)

            if x == 0:
                raise CustomerNotFoundException('Enter a valid Customer ID')

            sql = '''
                SELECT * FROM Lease WHERE customerID = %s AND CURDATE() BETWEEN
startDate AND endDate
            '''
            para = (customer_id,)
```

```
            my_cursor.execute(sql, para)
            x = my_cursor.fetchone()

            if x is not None:
                print('Lease already exists cannot delete the customer')
                return
            sql_b = '''DELETE FROM Lease WHERE customerID = %s'''
            sql = '''DELETE FROM Customer WHERE customerID = %s'''
            para = (customer_id,)
            my_cursor.execute(sql_b, para)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Customer Removed Successfully')
        except CustomerNotFoundException as e:
            print('An error occurred: ', e)
        except Exception as e:
            print('An error occurred: ', e)
```

services/lease_manager.py

```
from db_connection.db_adapter import *
from models.lease import Lease
from custom_exceptions.custom_exceptions import *

class LeaseManager:
    def __init__(self):
        self.connection = get_db_connection()

    def create_lease(self, vehicle_id, customer_id, start_date, end_date,
lease_type):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            INSERT INTO Lease(leaseID ,vehicleID, customerID, startDate,
endDate, type)
            VALUES (%s ,%s, %s, %s, %s, %s)
            '''
            para = (get_ids('lease', 'leaseID'), vehicle_id, customer_id,
start_date, end_date, lease_type)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('Lease created successfully')
            return 1
        except Exception as e:
            print('An error occurred: ', e)

    def return_vehicle(self, lease_id):
        try:
```

```
            my_cursor = self.connection.cursor()
            sql1 = '''
            SELECT vehicleID FROM Lease WHERE leaseID = %s AND CURDATE()
BETWEEN startDate AND endDate
            '''
            para1 = (lease_id,)
            my_cursor.execute(sql1, para1)
            x = my_cursor.fetchone()[0]

            if x is None:
                print('Invalid Lease ID')
                return

            sql2 = '''
                UPDATE Vehicle SET status = 'available' WHERE vehicleID = %s
            '''
            para2 = (x,)
            my_cursor.execute(sql2, para2)
            self.connection.commit()
            print('Car returned successfully')
        except Exception as e:
            print('An error occurred: ', e)

    def list_active_leases(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * FROM Lease WHERE CURDATE() BETWEEN startDate AND endDate
            '''
            my_cursor.execute(sql)
            leases = [Lease(*row) for row in my_cursor.fetchall()]
            return leases

        except Exception as e:
            print('An error occurred: ', e)

    def list_lease_history(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * FROM Lease
            '''
            my_cursor.execute(sql)
            leases = [Lease(*row) for row in my_cursor.fetchall()]
            return leases

        except Exception as e:
            print('An error occurred: ', e)
```

```
    def get_lease_by_id(self, lease_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''SELECT * FROM Lease WHERE leaseID = %s'''
            para = (lease_id,)
            my_cursor.execute(sql, para)
            row = my_cursor.fetchone()
            if row:
                return Lease(*row)
            else:
                raise LeaseNotFoundException('Invalid Lease ID entered')
        except LeaseNotFoundException as lnfe:
            raise lnfe
        except Exception as e:
            print('An error occurred: ', e)
```

services/vehicle_manager.py

```
from db_connection.db_adapter import *
from models.vehicle import Vehicle
from custom_exceptions.custom_exceptions import *


class VehicleManager:
    def __init__(self):
        self.connection = get_db_connection()

    def add_vehicle(self, make, model, year, daily_rate, status,
passenger_capacity, engine_capacity):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            INSERT INTO Vehicle(vehicleID, make, model, year, dailyRate,
status, passengerCapacity, engineCapacity)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s)
            '''
            para = (
                get_ids('vehicle', 'vehicleID'),
                make,
                model,
                year,
                daily_rate,
                status,
                passenger_capacity,
                engine_capacity
            )
            my_cursor.execute(sql, para)
            self.connection.commit()
```

```
            print('Vehicle Added Successfully')
            return 1
        except Exception as e:
            print('An error occurred: ', e)

    def remove_vehicle(self, vehicle_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
            SELECT * FROM Lease WHERE vehicleID = %s AND CURDATE() BETWEEN
startDate AND endDate
            '''
            para = (vehicle_id,)
            my_cursor.execute(sql, para)
            x = my_cursor.fetchone()

            if x is not None:
                print('Vehicle is currently rented cannot be deleted')
                return

            a = get_cnts('vehicle', 'vehicleID', vehicle_id)

            if a == 0:
                print('Invalid vehicle ID/vehicle does not exist')
                return

            sql2 = '''
                DELETE FROM Vehicle WHERE vehicleID = %s
            '''
            para = (vehicle_id,)
            my_cursor.execute(sql2, para)
            self.connection.commit()
            print('Vehicle removed successfully')
        except Exception as e:
            print('An error occurred: ', e)

    def list_available_vehicles(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''SELECT * FROM Vehicle WHERE status = %s'''
            para = ('available',)
            my_cursor.execute(sql, para)
            vehicles = [Vehicle(*row) for row in my_cursor.fetchall()]
            return vehicles
        except Exception as e:
            print('An error occurred: ', e)

    def list_rented_vehicles(self):
        try:
```

```
            my_cursor = self.connection.cursor()
            sql = '''SELECT * FROM Vehicle WHERE status = %s'''
            para = ('notAvailable',)
            my_cursor.execute(sql, para)
            vehicles = [Vehicle(*row) for row in my_cursor.fetchall()]
            return vehicles
        except Exception as e:
            print('An error occurred: ', e)

    def find_vehicle_by_id(self, vehicle_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''SELECT * FROM Vehicle WHERE vehicleID = %s'''
            para = (vehicle_id,)
            my_cursor.execute(sql, para)
            row = my_cursor.fetchone()
            if row:
                return Vehicle(*row)
            else:
                raise VehicleNotFoundException('Invalid Vehicle ID entered')
        except VehicleNotFoundException as vnfe:
            raise vnfe
        except Exception as e:
            print('An error occurred: ', e)
```

Main.py

```
from db_connection.db_adapter import get_ids, get_cnts
from models.customer import Customer
from models.vehicle import Vehicle
from models.lease import Lease
from datetime import date
from services.customer_manager import CustomerManager
from services.vehicle_manager import VehicleManager
from services.lease_manager import LeaseManager
from services.payment_manager import PaymentManager

def print_menu():
    print("\nCar Rental System Menu:")
    print("1. Customer Management")
    print("2. Vehicle Management")
    print("3. Lease Management")
    print("4. Payment Management")
    print("0. Exit")

def customer_menu(customer_manager):
    while True:
        print("\nCustomer Management Menu:")
```

```
        print("1. Add New Customer")
        print("2. Update Customer Info")
        print("3. Get Customer Details by ID")
        print("4. List All Customers")
        print("5. Remove Customer")
        print("0. Back to Main Menu")

        choice = input("Enter your choice: ")

        if choice == "1":
            first_name = input("Enter First Name: ")
            last_name = input("Enter Last Name: ")
            email = input("Enter Email: ")
            phone_number = input("Enter Phone Number: ")
            customer_manager.add_new_customer(first_name, last_name, email,
phone_number)

        elif choice == "2":
            customer_id = input("Enter Customer ID: ")
            first_name = input("Enter New First Name (leave empty to skip): ")
            last_name = input("Enter New Last Name (leave empty to skip): ")
            email = input("Enter New Email (leave empty to skip): ")
            phone_number = input("Enter New Phone Number (leave empty to skip):
")
            customer_manager.update_customer_info(customer_id, first_name,
last_name, email, phone_number)

        elif choice == "3":
            customer_id = input("Enter Customer ID: ")
            customer_manager.get_customer_details(customer_id)

        elif choice == "4":
            customer_manager.list_customers()

        elif choice == "5":
            customer_id = input("Enter Customer ID: ")
            customer_manager.remove_customer(customer_id)

        elif choice == "0":
            break

        else:
            print("Invalid choice. Please try again.")

def vehicle_menu(vehicle_manager):
    while True:
        print("\nVehicle Management Menu:")
        print("1. Add New Vehicle")
        print("2. Remove Vehicle")
```

```
        print("3. List Available Vehicles")
        print("4. List Rented Vehicles")
        print("5. Find Vehicle by ID")
        print("0. Back to Main Menu")

        choice = input("Enter your choice: ")

        if choice == "1":
            make = input("Enter Make: ")
            model = input("Enter Model: ")
            year = input("Enter Year: ")
            daily_rate = input("Enter Daily Rate: ")
            status = input('Enter vehicle status available/notAvailable: ')
            passenger_capacity = input("Enter Passenger Capacity: ")
            engine_capacity = input("Enter Engine Capacity: ")
            vehicle_manager.add_vehicle(get_ids('vehicle', 'vehicleID'), make,
model, year, daily_rate, status, passenger_capacity, engine_capacity)

        elif choice == "2":
            vehicle_id = input("Enter Vehicle ID: ")
            vehicle_manager.remove_vehicle(vehicle_id)

        elif choice == "3":
            available_vehicles = vehicle_manager.list_available_vehicles()
            print("Available Vehicles:")
            for vehicle in available_vehicles:
                print(vehicle)

        elif choice == "4":
            rented_vehicles = vehicle_manager.list_rented_vehicles()
            print("Rented Vehicles:")
            for vehicle in rented_vehicles:
                print(vehicle)

        elif choice == "5":
            vehicle_id = input("Enter Vehicle ID: ")
            vehicle = vehicle_manager.find_vehicle_by_id(vehicle_id)
            if vehicle:
                print("Found Vehicle:")
                print(vehicle)
            else:
                print(f"Vehicle with ID {vehicle_id} not found.")

        elif choice == "0":
            break

        else:
            print("Invalid choice. Please try again.")
```

```
def lease_menu(lease_manager):
    while True:
        print("\nLease Management Menu:")
        print("1. Create Lease")
        print("2. Return Vehicle")
        print("3. List Active Leases")
        print("4. List Lease History")
        print("0. Back to Main Menu")

        choice = input("Enter your choice: ")

        if choice == "1":
            customer_id = input("Enter Customer ID: ")
            vehicle_id = input("Enter Vehicle ID: ")
            start_date = input("Enter Start Date (YYYY-MM-DD): ")
            end_date = input("Enter End Date (YYYY-MM-DD): ")
            lease_type = input("Enter Lease Type Daily/Monthly: ")
            lease_manager.create_lease(vehicle_id, customer_id,start_date,
end_date, lease_type)

        elif choice == "2":
            lease_id = input("Enter Lease ID: ")
            lease_manager.return_vehicle(lease_id)

        elif choice == "3":
            active_leases = lease_manager.list_active_leases()
            print("Active Leases:")
            for lease in active_leases:
                print(lease)

        elif choice == "4":
            lease_history = lease_manager.list_lease_history()
            print("Lease History:")
            for lease in lease_history:
                print(lease)

        elif choice == "0":
            break

        else:
            print("Invalid choice. Please try again.")

def payment_menu(payment_manager, lease_manager):
    while True:
        print("\nPayment Management Menu:")
        print("1. Record Payment")
        print("0. Back to Main Menu")

        choice = input("Enter your choice: ")
```

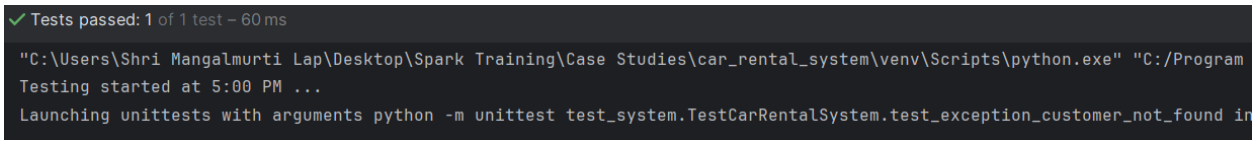
```
        if choice == "1":
            lease_id = input("Enter Lease ID: ")
            amount = input("Enter Payment Amount: ")
            lease = get_cnts('lease', 'leaseID', lease_id)

            if lease > 0:
                payment_manager.record_payment(lease_id, amount)
            else:
                print(f"Lease with ID {lease_id} not found.")

        elif choice == "0":
            break

        else:
            print("Invalid choice. Please try again.")

def main():
    customer_manager = CustomerManager()
    vehicle_manager = VehicleManager()
    lease_manager = LeaseManager()
    payment_manager = PaymentManager()

    while True:
        print_menu()
        choice = input("Enter your choice: ")

        if choice == "1":
            customer_menu(customer_manager)
        elif choice == "2":
            vehicle_menu(vehicle_manager)
        elif choice == "3":
            lease_menu(lease_manager)
        elif choice == "4":
            payment_menu(payment_manager, lease_manager)
        elif choice == "0":
            print("Exiting the Car Rental System. Goodbye!")
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

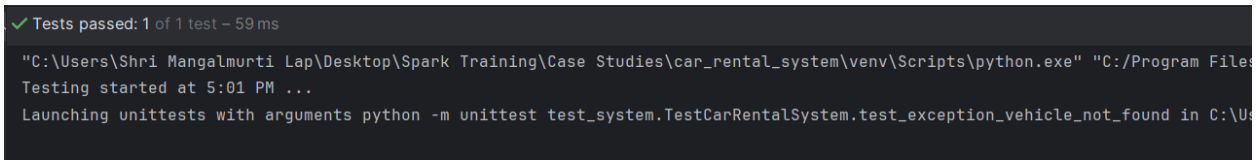
Test_system.py

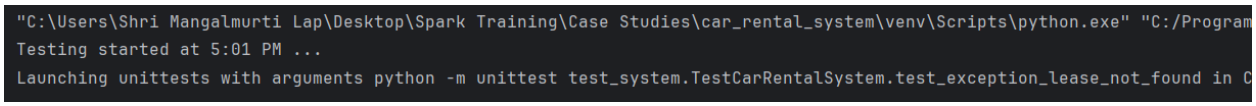
```
import unittest
```

```python
from custom_exceptions.custom_exceptions import CustomerNotFoundException,
VehicleNotFoundException, \
   LeaseNotFoundException
from services.lease_manager import LeaseManager
from services.vehicle_manager import VehicleManager
from services.payment_manager import PaymentManager
from services.customer_manager import CustomerManager


class TestCarRentalSystem(unittest.TestCase):
   def setUp(self):
       self.lease_manager = LeaseManager()
       self.vehicle_manager = VehicleManager()
       self.payment_manager = PaymentManager()
       self.customer_manager = CustomerManager()

   @unittest.skip
   def test_create_vehicle_success(self):
       x = self.vehicle_manager.add_vehicle('Toyota', 'Camry', 2022, 50.0,
'available', 5, 2500)
       self.assertIsInstance(x, int)
       self.assertGreater(x, 0)

   @unittest.skip
   def test_create_lease_success(self):
       x = self.lease_manager.create_lease(11, 1, '2023-12-26', '2023-12-31',
'Daily')

       self.assertIsInstance(x, int)
       self.assertGreater(x, 0)

   def test_exception_customer_not_found(self):
       with self.assertRaises(CustomerNotFoundException):
           self.customer_manager.get_customer_by_id(999)

   def test_exception_vehicle_not_found(self):
       with self.assertRaises(VehicleNotFoundException):
           self.vehicle_manager.find_vehicle_by_id(999)

   def test_exception_lease_not_found(self):
       with self.assertRaises(LeaseNotFoundException):
           self.lease_manager.get_lease_by_id(999)


if __name__ == '__main__':
   unittest.main()
```

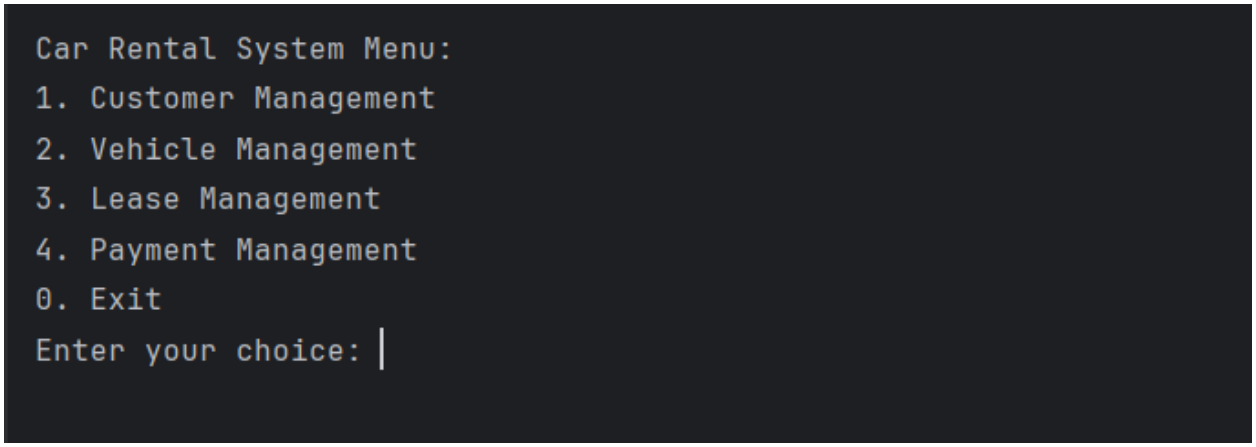
Testing Output : -







System Output : -



```
Car Rental System Menu:
1. Customer Management
2. Vehicle Management
3. Lease Management
4. Payment Management
0. Exit
Enter your choice: 2

Vehicle Management Menu:
1. Add New Vehicle
2. Remove Vehicle
3. List Available Vehicles
4. List Rented Vehicles
5. Find Vehicle by ID
0. Back to Main Menu
Enter your choice: 3
Available Vehicles:
Vehicle ID: 1
Make: Toyota
Model: Camry
Year: 2022
Daily Rate: 50.00
Status: available
Passenger Capacity: 4
Engine Capacity: 1450
Vehicle ID: 2
```

```
Car Rental System Menu:
1. Customer Management
2. Vehicle Management
3. Lease Management
4. Payment Management
0. Exit
Enter your choice: 4

Payment Management Menu:
1. Record Payment
0. Back to Main Menu
Enter your choice:
```