CarConnect, a Car Rental Platform

Balakumaran P

Instructions:

• Submitting assignments should be a single file or through git hub link shared with trainer and hexavarsity.

• Each assignment builds upon the previous one, and by the end, you will have a comprehensive application implemented in Java/C#/Python with a strong focus on SQL schema design, control flow statements, loops, arrays, collections, and database interaction.

• Follow object-oriented principles throughout the Java programming assignments. Use classes and objects to model real-world entities, encapsulate data and behavior, and ensure code reusability.

• Throw user defined exception from method and handle in the main method.

• The following Directory structure is to be followed in the application.

o entity/model

▪ Create entity classes in this package. All entity class should not have any business logic.

o dao

▪ Create Service Provider interface/abstract class to showcase functionalities.

▪ Create the implementation class for the above interface/abstract class with db interaction.

o exception

▪ Create user defined exceptions in this package and handle exceptions whenever needed.

o util

▪ Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.

▪ Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object.

o main

▪ Create a class MainModule and demonstrate the functionalities in a menu driven application.

Key Functionalities:

User Authentication:

• Secure user authentication and authorization mechanisms.

Vehicle Management:

• CRUD operations for vehicles, including details such as model, make, availability, and pricing.Reservation System:

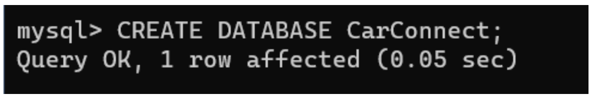
• Real-time reservation handling with conflict resolution.

• Email/SMS notifications for reservation confirmation and reminders.Reporting:

• Generation of reports for administrators, including reservation history, vehicle utilization, and revenue.

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

Created a database named CarConnect:



```
mysql> CREATE DATABASE CarConnect;
Query OK, 1 row affected (0.05 sec)
```

SQL Tables:

1. Customer Table:

• CustomerID (Primary Key): Unique identifier for each customer.

• FirstName: First name of the customer.

• LastName: Last name of the customer.

• Email: Email address of the customer for communication.

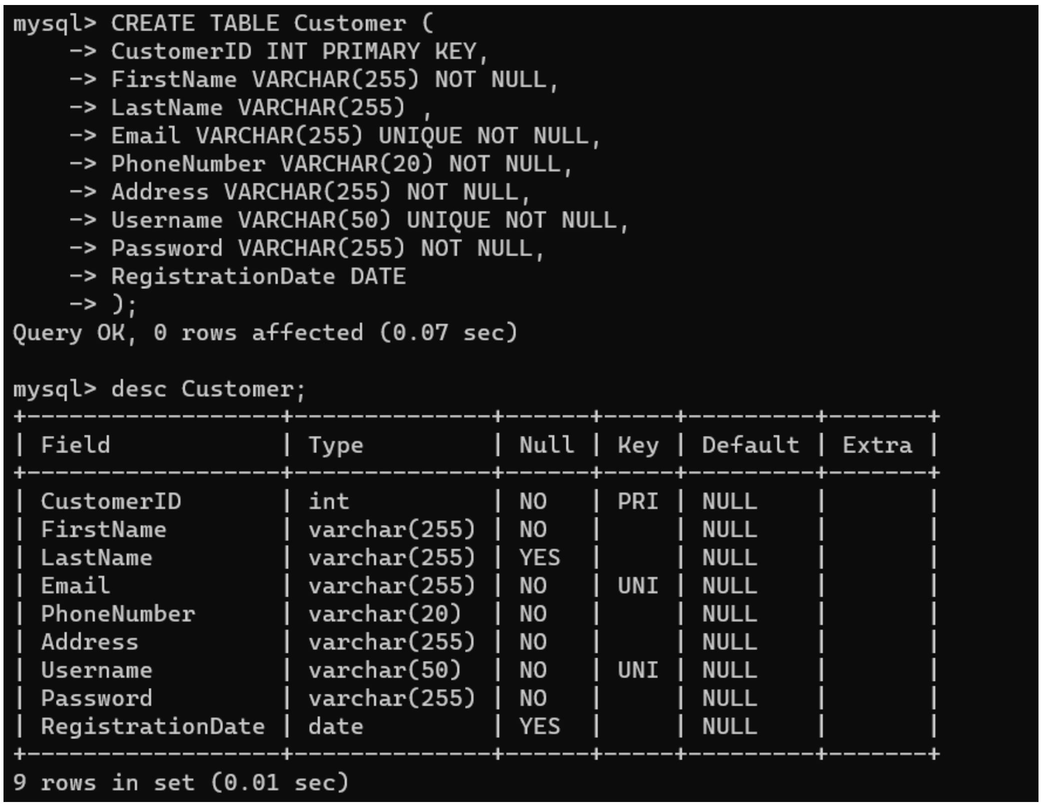
• PhoneNumber: Contact number of the customer.

• Address: Customer's residential address.

• Username: Unique username for customer login.

• Password: Securely hashed password for customer authentication.

• RegistrationDate: Date when the customer registered.



```
mysql> CREATE TABLE Customer (
    -> CustomerID INT PRIMARY KEY,
    -> FirstName VARCHAR(255) NOT NULL,
    -> LastName VARCHAR(255) ,
    -> Email VARCHAR(255) UNIQUE NOT NULL,
    -> PhoneNumber VARCHAR(20) NOT NULL,
    -> Address VARCHAR(255) NOT NULL,
    -> Username VARCHAR(50) UNIQUE NOT NULL,
    -> Password VARCHAR(255) NOT NULL,
    -> RegistrationDate DATE
    -> );
Query OK, 0 rows affected (0.07 sec)

mysql> desc Customer;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| CustomerID       | int          | NO   | PRI | NULL    |       |
| FirstName        | varchar(255) | NO   |     | NULL    |       |
| LastName         | varchar(255) | YES  |     | NULL    |       |
| Email            | varchar(255) | NO   | UNI | NULL    |       |
| PhoneNumber      | varchar(20)  | NO   |     | NULL    |       |
| Address          | varchar(255) | NO   |     | NULL    |       |
| Username         | varchar(50)  | NO   | UNI | NULL    |       |
| Password         | varchar(255) | NO   |     | NULL    |       |
| RegistrationDate | date         | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
9 rows in set (0.01 sec)
```

2. Vehicle Table:

• VehicleID (Primary Key): Unique identifier for each vehicle.

• Model: Model of the vehicle.

• Make: Manufacturer or brand of the vehicle.

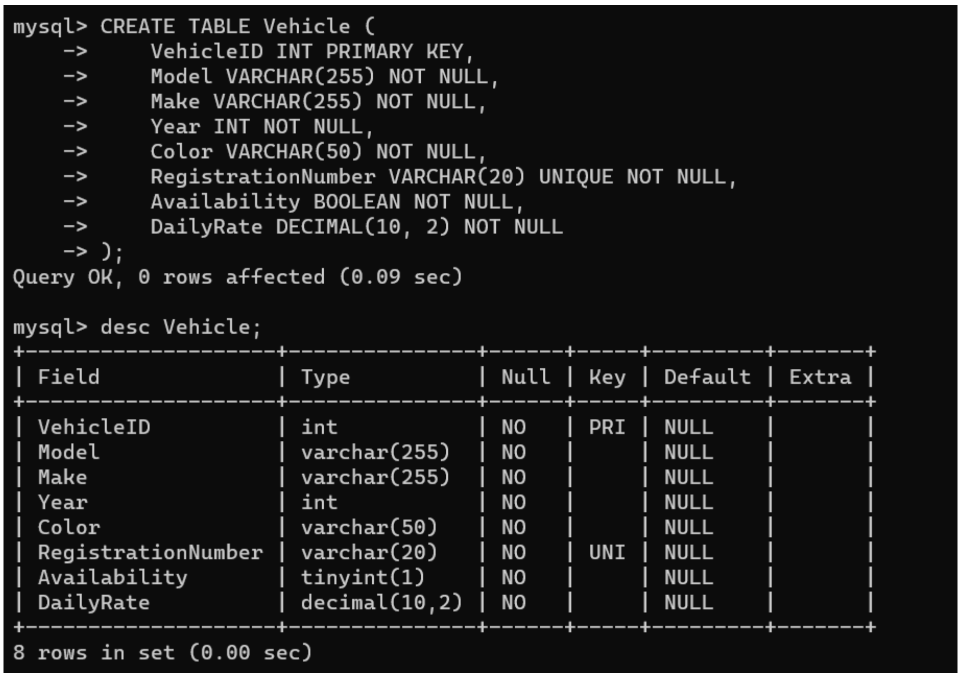
• Year: Manufacturing year of the vehicle.

• Color: Color of the vehicle.

• RegistrationNumber: Unique registration number for each vehicle.

• Availability: Boolean indicating whether the vehicle is available for rent.

• DailyRate: Daily rental rate for the vehicle.



```
mysql> CREATE TABLE Vehicle (
    ->     VehicleID INT PRIMARY KEY,
    ->     Model VARCHAR(255) NOT NULL,
    ->     Make VARCHAR(255) NOT NULL,
    ->     Year INT NOT NULL,
    ->     Color VARCHAR(50) NOT NULL,
    ->     RegistrationNumber VARCHAR(20) UNIQUE NOT NULL,
    ->     Availability BOOLEAN NOT NULL,
    ->     DailyRate DECIMAL(10, 2) NOT NULL
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> desc Vehicle;
+--------------------+---------------+------+-----+---------+-------+
| Field              | Type          | Null | Key | Default | Extra |
+--------------------+---------------+------+-----+---------+-------+
| VehicleID          | int           | NO   | PRI | NULL    |       |
| Model              | varchar(255)  | NO   |     | NULL    |       |
| Make               | varchar(255)  | NO   |     | NULL    |       |
| Year               | int           | NO   |     | NULL    |       |
| Color              | varchar(50)   | NO   |     | NULL    |       |
| RegistrationNumber | varchar(20)   | NO   | UNI | NULL    |       |
| Availability       | tinyint(1)    | NO   |     | NULL    |       |
| DailyRate          | decimal(10,2) | NO   |     | NULL    |       |
+--------------------+---------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

3. Reservation Table:

• ReservationID (Primary Key): Unique identifier for each reservation.

• CustomerID (Foreign Key): Foreign key referencing the Customer table.

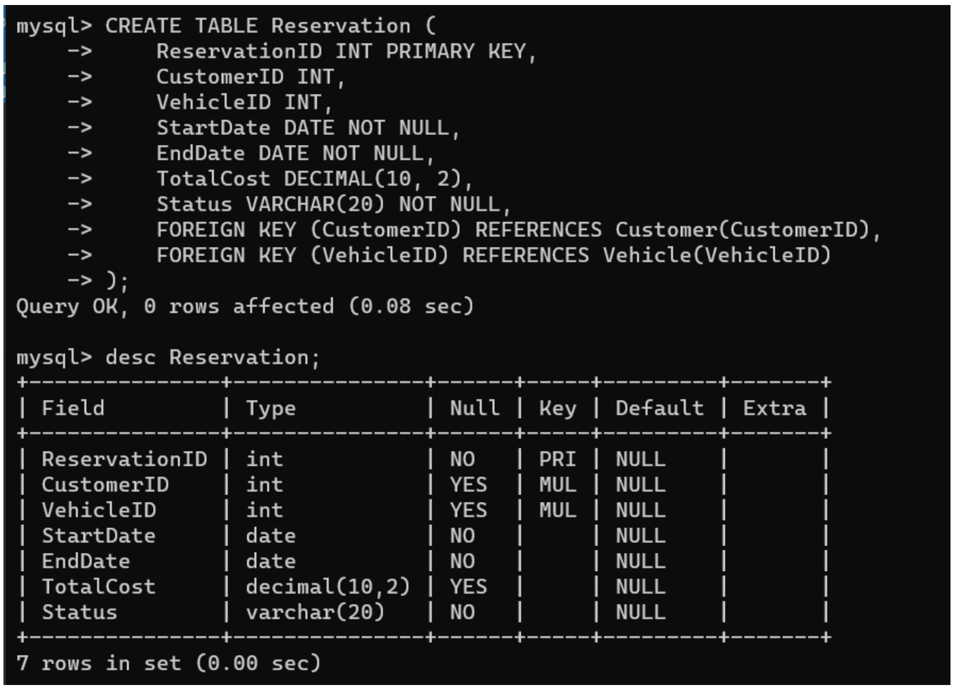
• VehicleID (Foreign Key): Foreign key referencing the Vehicle table.

• StartDate: Date and time of the reservation start.

• EndDate: Date and time of the reservation end.

• TotalCost: Total cost of the reservation.

• Status: Current status of the reservation (e.g., pending, confirmed, completed).



```
mysql> CREATE TABLE Reservation (
    ->     ReservationID INT PRIMARY KEY,
    ->     CustomerID INT,
    ->     VehicleID INT,
    ->     StartDate DATE NOT NULL,
    ->     EndDate DATE NOT NULL,
    ->     TotalCost DECIMAL(10, 2),
    ->     Status VARCHAR(20) NOT NULL,
    ->     FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    ->     FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID)
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> desc Reservation;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| ReservationID | int           | NO   | PRI | NULL    |       |
| CustomerID    | int           | YES  | MUL | NULL    |       |
| VehicleID     | int           | YES  | MUL | NULL    |       |
| StartDate     | date          | NO   |     | NULL    |       |
| EndDate       | date          | NO   |     | NULL    |       |
| TotalCost     | decimal(10,2) | YES  |     | NULL    |       |
| Status        | varchar(20)   | NO   |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

4. Admin Table:

• AdminID (Primary Key): Unique identifier for each admin.

• FirstName: First name of the admin.

• LastName: Last name of the admin.

• Email: Email address of the admin for communication.

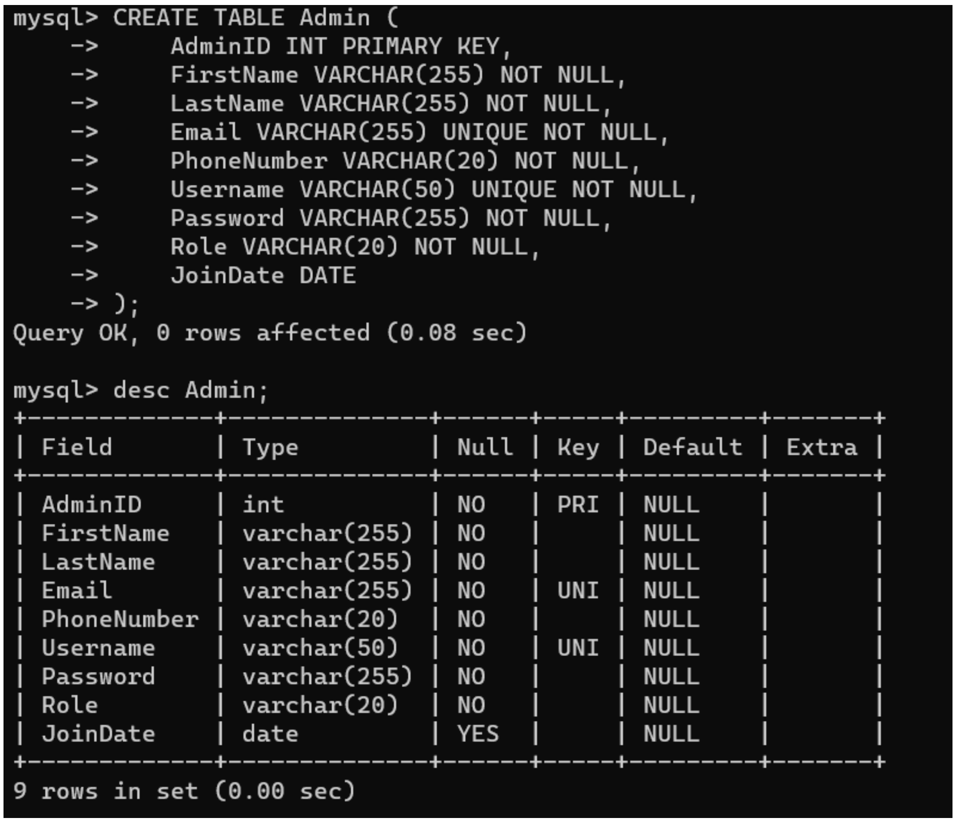
• PhoneNumber: Contact number of the admin.

• Username: Unique username for admin login.

• Password: Securely hashed password for admin authentication.

• Role: Role of the admin within the system (e.g., super admin, fleet manager).

• JoinDate: Date when the admin joined the system.



```
mysql> CREATE TABLE Admin (
    ->     AdminID INT PRIMARY KEY,
    ->     FirstName VARCHAR(255) NOT NULL,
    ->     LastName VARCHAR(255) NOT NULL,
    ->     Email VARCHAR(255) UNIQUE NOT NULL,
    ->     PhoneNumber VARCHAR(20) NOT NULL,
    ->     Username VARCHAR(50) UNIQUE NOT NULL,
    ->     Password VARCHAR(255) NOT NULL,
    ->     Role VARCHAR(20) NOT NULL,
    ->     JoinDate DATE
    -> );
Query OK, 0 rows affected (0.08 sec)

mysql> desc Admin;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| AdminID     | int          | NO   | PRI | NULL    |       |
| FirstName   | varchar(255) | NO   |     | NULL    |       |
| LastName    | varchar(255) | NO   |     | NULL    |       |
| Email       | varchar(255) | NO   | UNI | NULL    |       |
| PhoneNumber | varchar(20)  | NO   |     | NULL    |       |
| Username    | varchar(50)  | NO   | UNI | NULL    |       |
| Password    | varchar(255) | NO   |     | NULL    |       |
| Role        | varchar(20)  | NO   |     | NULL    |       |
| JoinDate    | date         | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
9 rows in set (0.00 sec)
```

Create the model/entity classes corresponding to the schema within package entity with variables

declared private, constructors (default and parametrized) and getters,setters)

Classes:

- **Customer:**
- **Properties**: CustomerID, FirstName, LastName, Email, PhoneNumber, Address, Username, Password, RegistrationDate
- **Methods**: Authenticate(password)

```
class Customer:
    def __init__(self, customerID, firstName, lastName, email, phoneNumber,
address, username, password,
                 registrationDate):
        self._customerID = customerID
        self._firstName = firstName
        self._lastName = lastName
        self._email = email
        self._phoneNumber = phoneNumber
        self._address = address
        self._username = username
        self._password = password
        self._registrationDate = registrationDate

    @property
    def customerID(self):
        return self._customerID

    @customerID.setter
    def customerID(self, value):
        self._customerID = value

    @property
    def firstName(self):
        return self._firstName

    @firstName.setter
    def firstName(self, value):
        self._firstName = value

    @property
    def lastName(self):
        return self._lastName
```

```python
    @lastName.setter
    def lastName(self, value):
        self._lastName = value

    @property
    def email(self):
        return self._email

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

    @property
    def phoneNumber(self):
        return self._phoneNumber

    @phoneNumber.setter
    def phoneNumber(self, value):
        self._phoneNumber = value

    @property
    def address(self):
        return self._address

    @address.setter
    def address(self, value):
        self._address = value

    @property
    def username(self):
        return self._username

    @username.setter
    def username(self, value):
        self._username = value

    @property
    def password(self):
        return self._password
    @password.setter
    def password(self, value):
        self._password = value

    @property
    def registrationDate(self):
        return self._registrationDate

    @registrationDate.setter
    def registrationDate(self, value):
        self._registrationDate = value

    def authenticate(self, enteredPassword):
        return self._password == enteredPassword
```

• **Vehicle:**

• **Properties**: VehicleID, Model, Make, Year, Color, RegistrationNumber, Availability, DailyRate

```python
class Vehicle:
    def __init__(self, vehicleID, model, make, year, color, registrationNumber,
availability, dailyRate):
        self._vehicleID = vehicleID
        self._model = model
        self._make = make
        self._year = year
        self._color = color
        self._registrationNumber = registrationNumber
        self._availability = availability
        self._dailyRate = dailyRate

    @property
    def vehicleID(self):
        return self._vehicleID

    @vehicleID.setter
    def vehicleID(self, value):
        self._vehicleID = value

    @property
    def model(self):
        return self._model
    @model.setter
    def model(self, value):
        self._model = value

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

    @make.setter
    def make(self, value):
        self._make = value

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

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

    @property
    def color(self):
        return self._color

    @color.setter
    def color(self, value):
        self._color = value

    @property
    def registrationNumber(self):
        return self._registrationNumber

    @registrationNumber.setter
    def registrationNumber(self, value):
        self._registrationNumber = value

    @property
    def availability(self):
        return self._availability
    @availability.setter
    def availability(self, value):
        self._availability = value

    @property
    def dailyRate(self):
        return self._dailyRate

    @dailyRate.setter
    def dailyRate(self, value):
        self._dailyRate = value
```

• **Reservation**:

• **Properties**: ReservationID, CustomerID, VehicleID, StartDate, EndDate, TotalCost, Status

• **Methods**: CalculateTotalCost()

```
class Reservation:
    def __init__(self, reservationID, customerID, vehicleID, startDate, 
endDate, totalCost, status):
        self._reservationID = reservationID
        self._customerID = customerID
        self._vehicleID = vehicleID
        self._startDate = startDate
        self._endDate = endDate
        self._totalCost = totalCost
        self._status = status

    @property
    def reservationID(self):
        return self._reservationID

    @reservationID.setter
    def reservationID(self, value):
        self._reservationID = value

    @property
    def customerID(self):
        return self._customerID

    @customerID.setter
    def customerID(self, value):
        self._customerID = value

    @property
    def vehicleID(self):
        return self._vehicleID

    @vehicleID.setter
    def vehicleID(self, value):
        self._vehicleID = value

    @property
    def startDate(self):
        return self._startDate

    @startDate.setter
    def startDate(self, value):
        self._startDate = value

    @property
    def endDate(self):
        return self._endDate

    @endDate.setter
    def endDate(self, value):
        self._endDate = value

    @property
    def totalCost(self):
        return self._totalCost

    @totalCost.setter
    def totalCost(self, value):
        self._totalCost = value

    @property
    def status(self):
        return self._status

    @status.setter
    def status(self, value):
        self._status = value
    def calculateTotalCost(self):
        pass
```

• **Admin**:

• **Properties**: AdminID, FirstName, LastName, Email, PhoneNumber, Username, Password, Role, JoinDate

• **Methods**: Authenticate(password)

```
class Admin:
    def __init__(self, adminID, firstName, lastName, email, phoneNumber,
username, password, role, joinDate):
        self._adminID = adminID
        self._firstName = firstName
        self._lastName = lastName
        self._email = email
        self._phoneNumber = phoneNumber
        self._username = username
        self._password = password
        self._role = role
        self._joinDate = joinDate

    @property
    def adminID(self):
        return self._adminID

    @adminID.setter
    def adminID(self, value):
        self._adminID = value

    @property
    def firstName(self):
        return self._firstName

    @firstName.setter
    def firstName(self, value):
        self._firstName = value

    @property
    def lastName(self):
        return self._lastName
```

```python
    @lastName.setter
    def lastName(self, value):
        self._lastName = value

    @property
    def email(self):
        return self._email

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

    @property
    def phoneNumber(self):
        return self._phoneNumber

    @phoneNumber.setter
    def phoneNumber(self, value):
        self._phoneNumber = value

    @property
    def username(self):
        return self._username

    @username.setter
    def username(self, value):
        self._username = value

    @property
    def password(self):
        return self._password

    @password.setter
    def password(self, value):
        self._password = value

    @property
    def role(self):
        return self._role

    @role.setter
    def role(self, value):
        self._role = value

    @property
    def joinDate(self):
        return self._joinDate

    @joinDate.setter
    def joinDate(self, value):
        self._joinDate = value

    def authenticate(self, enteredPassword):
        return self._password == enteredPassword
```

• **CustomerService** (implements **ICustomerService**):

• **Methods**: GetCustomerById, GetCustomerByUsername, RegisterCustomer, UpdateCustomer, DeleteCustomer

```
from dao.service.ICustomerService import ICustomerService


class CustomerService(ICustomerService):
    def __init__(self, database_context):
        self.db_context = database_context

    def generate_customer_id(self):
        self.db_context.connect()
        connection = self.db_context.get_connection()

        cur = connection.cursor()
        cur.execute("SELECT MAX(CustomerID) FROM Customer")
        customer_id = cur.fetchone()[0]
        if customer_id is None:
            customer_id = 1
        else:
            customer_id += 1
        return customer_id

    def get_customer_by_id(self, customer_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Customer WHERE CustomerID = %s",
(customer_id,))
        res = cur.fetchone()
        return res

    def get_customer_by_username(self, username):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Customer WHERE Username = %s", (username,))
        res = cur.fetchone()
        return res

    def register_customer(self, customer_data):
        connection = self.db_context.get_connection()
        customer_id = self.generate_customer_id()
        cur = connection.cursor()
        cur.execute(
            "INSERT INTO Customer "
            "(CustomerID, FirstName, LastName, Email, PhoneNumber, Address, 
Username, Password, "
            "RegistrationDate) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)",
            (customer_id, customer_data['FirstName'], 
customer_data['LastName'], customer_data['Email'],
             customer_data['PhoneNumber'], customer_data['Address'], 
customer_data['Username'],
             customer_data['Password'], customer_data['RegistrationDate'])
        )
        connection.commit()
        print("Customer registration successful...")

    def update_customer(self, customer_data):
        connection = self.db_context.get_connection()
        cur = connection.cursor()

        customer_id = customer_data.get("customer_id")
        first_name = customer_data.get("first_name")
        last_name = customer_data.get("last_name")
        email = customer_data.get("email")
        phone_number = customer_data.get("phone_number")
        address = customer_data.get("address")
        username = customer_data.get("username")
        password = customer_data.get("password")
        update_query = """
            UPDATE Customer
            SET
                FirstName = %s,
                LastName = %s,
                Email = %s,
                PhoneNumber = %s,
                Address = %s,
                Username = %s,
                Password = %s
            WHERE
                CustomerID = %s
        """
        cur.execute(update_query,
                    (first_name, last_name, email, phone_number, address,
username, password, customer_id))
        connection.commit()

    def delete_customer(self, customer_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("DELETE FROM Customer WHERE CustomerID = %s",
(customer_id,))
        connection.commit()
```

• **VehicleService** (implements IVehicleService):

• **Methods**: GetVehicleById, GetAvailableVehicles, AddVehicle, UpdateVehicle, RemoveVehicle

```
from dao.service.IVehicleService import IVehicleService
from exception.CustomExceptions import VehicleNotFoundException


class VehicleService(IVehicleService):
    def __init__(self, database_context):
        self.db_context = database_context

    def generate_vehicle_id(self):
        self.db_context.connect()
        connection = self.db_context.get_connection()

        cur = connection.cursor()
        cur.execute("SELECT MAX(VehicleID) FROM Vehicle")
        max_vehicle_id = cur.fetchone()[0]
        if max_vehicle_id is None:
            return 1
        else:
            return max_vehicle_id + 1

    def get_vehicle_by_id(self, vehicle_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Vehicle WHERE VehicleID = %s",
(vehicle_id,))
        vehicle = cur.fetchone()
        connection.commit()
        if not vehicle:
            raise VehicleNotFoundException
        return vehicle

    def get_available_vehicles(self):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Vehicle WHERE Availability = 1")
        res = cur.fetchall()
        return res

    def add_vehicle(self, vehicle_data):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        vehicle_id = self.generate_vehicle_id()
        cur.execute("INSERT INTO Vehicle VALUES (%s,%s,%s,%s,%s,%s,%s,%s)"
                    , (vehicle_id, vehicle_data['Model'], vehicle_data['Make'],
                       vehicle_data['Year'], vehicle_data['Color'],
vehicle_data['RegistrationNumber'],
                       vehicle_data['Availability'],
vehicle_data['DailyRate']))
        connection.commit()
        return vehicle_id

    def update_vehicle(self, vehicle_data):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        vehicle_id=vehicle_data.get('vehicle_id')
        model = vehicle_data.get('Model')
        make = vehicle_data.get('Make')
        year = vehicle_data.get('Year')
        color = vehicle_data.get('Color')
        reg_no = vehicle_data.get('RegistrationNumber')
        avl = vehicle_data.get('Availability')
        daily_rate = vehicle_data.get('DailyRate')
        cur.execute('''
            UPDATE Vehicle
            SET Model = %s,
                Make = %s,
                Year = %s,
                Color = %s,
                RegistrationNumber = %s,
                Availability = %s,
                DailyRate = %s
            WHERE VehicleID = %s
            ''',
                    (model, make, year, color, reg_no, avl, daily_rate, 
vehicle_id))
        connection.commit()

    def remove_vehicle(self, vehicle_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("DELETE FROM Vehicle WHERE VehicleID = %s", (vehicle_id,))
        connection.commit()
```

• **ReservationService** (implements **IReservationService**):

• **Methods**: GetReservationById, GetReservationsByCustomerId, CreateReservation, UpdateReservation, CancelReservation

```
from dao.service.IReservationService import IReservationService
from exception.CustomExceptions import ReservationException



class ReservationService(IReservationService):
    def __init__(self, database_context):
        self.db_context = database_context
    def generate_reservation_id(self):
        self.db_context.connect()
        connection = self.db_context.get_connection()

        cur = connection.cursor()
        cur.execute("SELECT MAX(ReservationID) FROM Reservation")
        reservation_id = cur.fetchone()[0]
        if reservation_id is None:
            reservation_id = 1
        else:
            reservation_id += 1
        return reservation_id

    def get_reservation_by_id(self, reservation_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Reservation WHERE ReservationID = %s",
(reservation_id,))
        res = cur.fetchone()
        return res

    def get_reservations_by_customer_id(self, customer_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT * FROM Reservation WHERE CustomerID = %s",
(customer_id,))
        res = cur.fetchall()
        return res

    def create_reservation(self, reservation_data):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT COUNT(*) FROM Reservation WHERE VehicleID = %s",
(reservation_data['VehicleID'],))
        count = cur.fetchone()[0]

        if count > 0:
            raise ReservationException("Attempting to make a reservation for a
vehicle that is already reserved.")
        else:
```

```python
            res_id = self.generate_reservation_id()
            cur.execute("INSERT INTO Reservation VALUES
(%s,%s,%s,%s,%s,%s,%s)",
                        (res_id, reservation_data['CustomerID'],
reservation_data['VehicleID'],
                         reservation_data['StartDate'],
reservation_data['EndDate'], reservation_data['TotalCost'],
                         reservation_data['Status']))
            connection.commit()

    def update_reservation(self, reservation_data):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        res_id = reservation_data.get('ReservationID')
        CustomerID = reservation_data.get('CustomerID')
        VehicleID = reservation_data.get('VehicleID')
        StartDate = reservation_data.get('StartDate')
        EndDate = reservation_data.get('EndDate')
        TotalCost = reservation_data.get('TotalCost')
        Status = reservation_data.get('Status')
        cur.execute('''
                    UPDATE Reservation
                    SET CustomerID = %s,
                        VehicleID = %s,
                        StartDate = %s,
                        EndDate = %s,
                        TotalCost = %s,
                        Status = %s
                    WHERE ReservationID = %s
                    ''',
                    (CustomerID, VehicleID, StartDate, EndDate, TotalCost,
Status, res_id))
        connection.commit()
        print("Updated successfully....")

    def cancel_reservation(self, reservation_id):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("DELETE FROM Reservation WHERE ReservationID = %s",
(reservation_id,))
        connection.commit()
```

• **AdminService** (implements IAdminService):

• **Methods**: GetAdminById, GetAdminByUsername, RegisterAdmin, UpdateAdmin, DeleteAdmin

```
from dao.service.IAdminService import IAdminService
from exception.CustomExceptions import AdminNotFoundException


class AdminService(IAdminService):
    def __init__(self,database_context):
        self.database_context=database_context
    def generate_admin_id(self):
        connection=self.database_context.get_connection()
        cur=connection.cursor()
        cur.execute("SELECT MAX(AdminID) FROM Admin")
        admin_id=cur.fetchone()[0]
        if admin_id is None:
            admin_id=1
        else:
            admin_id+=1
        return admin_id

    def get_admin_by_username(self, username):
        con=self.database_context.get_connection()
        cur=con.cursor()
        cur.execute("SELECT * FROM Admin WHERE username = %s",(username,))
        res=cur.fetchone()
        if res is None:
            raise AdminNotFoundException
        return res
    def get_admin_by_id(self, admin_id):
        con = self.database_context.get_connection()
        cur = con.cursor()
        cur.execute("SELECT * FROM Admin WHERE username = %s", (admin_id,))
        res = cur.fetchone()

        return res
    def register_admin(self, admin_data):
        connection = self.database_context.get_connection()
        cur = connection.cursor()
        admin_id = self.generate_admin_id()
        cur.execute("INSERT INTO Admin VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"
                    , (admin_id, admin_data['FirstName'],
admin_data['LastName'],
                       admin_data['Email'], admin_data['PhoneNumber'],
admin_data['Username'],
admin_data['Password'],admin_data['Role'],admin_data['JoinDate']))
        connection.commit()

    def update_admin(self, admin_data):
        connection = self.database_context.get_connection()
        cur = connection.cursor()
        AdminID=admin_data.get('AdminID')
        FirstName = admin_data.get('FirstName')
        LastName = admin_data.get('LastName')
        Email = admin_data.get('Email')
        PhoneNumber = admin_data.get('PhoneNumber')
        Username = admin_data.get('Username')
        Password = admin_data.get('Password')
        Role = admin_data.get('Role')
        JoinDate = admin_data.get('JoinDate')
        cur.execute('''
            UPDATE Admin
            SET FirstName = %s,
                LastName = %s,
                Email = %s,
                PhoneNumber = %s,
                Username = %s,
                Password = %s,
                Role = %s,
                JoinDate = %s
            WHERE
                AdminID = %s
        ''', (FirstName, LastName, Email, PhoneNumber, Username, Password,
Role, JoinDate, AdminID))

        connection.commit()

    def delete_admin(self, admin_id):
        con=self.database_context.get_connection()
        cur=con.cursor()
        cur.execute("DELETE FROM Admin WHERE AdminID = %s", (admin_id,))
        con.commit()
```

• **DatabaseContext**:

• A class responsible for handling database connections and interactions.

```
from exception.CustomExceptions import DatabaseConnectionException
from util.DBConnUtil import DBConnUtil
from util.DBPropertyUtil import DBPropertyUtil

class DatabaseContext:
    def __init__(self):
        self.connection_string = DBPropertyUtil.get_connection_string()
        self.connection = None

    def connect(self):
        try:
            self.connection = DBConnUtil.get_connection(self.connection_string)
        except DatabaseConnectionException as e:
            print(e)

    def get_connection(self):
        return self.connection
```

• **AuthenticationService**:

• A class responsible for handling user authentication.

```
from exception.CustomExceptions import AuthenticationException


class AuthenticationService:
    def __init__(self, database_context):
        self.db_context = database_context

    def authenticate_user(self, username, password):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT Username,Password FROM Customer WHERE Username = %s
and Password = %s",
                    (username, password))
        res = cur.fetchone()
        if res is None:
            raise AuthenticationException
        return res is not None
    def authenticate_admin(self, username, password):
        connection = self.db_context.get_connection()
        cur = connection.cursor()
        cur.execute("SELECT UserName,Password FROM Admin WHERE Username = %s 
AND Password = %s",
                    (username, password))
        res = cur.fetchone()
        if res is None:
            raise AuthenticationException
        return res is not None
```

• **ReportGenerator**:

• A class for generating reports based on reservation and vehicle data.

```
class ReportGenerator:
    def __init__(self, database_context):
        self.db_context = database_context

    def generate_report(self,res_id):
        connection = self.db_context.get_connection()
        cursor = connection.cursor()

        query = "SELECT r.ReservationID, r.CustomerID, r.VehicleID, 
r.StartDate, r.EndDate, r.TotalCost, r.Status, " \
                "v.Model, v.Make, v.Year, v.Color, v.RegistrationNumber, 
v.Availability, v.DailyRate " \
                "FROM Reservation r JOIN Vehicle v ON r.VehicleID = v.VehicleID 
WHERE r.ReservationID = %s"

        cursor.execute(query,(res_id,))
        combined_data = cursor.fetchall()

        print("Report:")
        for entry in combined_data:
            print(f"Reservation ID: {entry[0]}")
            print(f"Customer ID: {entry[1]}")
            print(f"Vehicle ID: {entry[2]}")
            print(f"Start Date: {entry[3]}")
            print(f"End Date: {entry[4]}")
            print(f"Total Cost: {entry[5]}")
            print(f"Status: {entry[6]}")
            print(f"Model: {entry[7]}")
            print(f"Make: {entry[8]}")
            print(f"Year: {entry[9]}")
            print(f"Color: {entry[10]}")
            print(f"Registration Number: {entry[11]}")
            print(f"Availability: {entry[12]}")
            print(f"Daily Rate: {entry[13]}")
            print("\n")
```

Interfaces:

• **ICustomerService:**

• GetCustomerById(customerId)

• GetCustomerByUsername(username)

• RegisterCustomer(customerData)

• UpdateCustomer(customerData)

• DeleteCustomer(customerId)

```
from abc import ABC, abstractmethod
class ICustomerService(ABC):
    @abstractmethod
    def get_customer_by_id(self, customer_id):
        pass

    @abstractmethod
    def get_customer_by_username(self, username):
        pass

    @abstractmethod
    def register_customer(self, customer_data):
        pass

    @abstractmethod
    def update_customer(self, customer_data):
        pass

    @abstractmethod
    def delete_customer(self, customer_id):
        pass
```

- **IVehicleService:**
- GetVehicleById(vehicleId)
- GetAvailableVehicles()
- AddVehicle(vehicleData)
- UpdateVehicle(vehicleData)
- RemoveVehicle(vehicleId)

```
from abc import ABC, abstractmethod
class IVehicleService(ABC):
    @abstractmethod
    def get_vehicle_by_id(self, vehicle_id):
        pass

    @abstractmethod
    def generate_vehicle_id(self):
        pass

    @abstractmethod
    def get_available_vehicles(self):
        pass

    @abstractmethod
    def add_vehicle(self, vehicle_data):
        pass

    @abstractmethod
    def update_vehicle(self, vehicle_data):
        pass

    @abstractmethod
    def remove_vehicle(self, vehicle_id):
        pass
```

- **IReservationService:**
- GetReservationById(reservationId)
- GetReservationsByCustomerId(customerId)
- CreateReservation(reservationData)

- UpdateReservation(reservationData)
- CancelReservation(reservationId)

```
from abc import ABC, abstractmethod
class IReservationService(ABC):
    @abstractmethod
    def get_reservation_by_id(self, reservation_id):
        pass

    @abstractmethod
    def generate_reservation_id(self):
        pass

    @abstractmethod
    def get_reservations_by_customer_id(self, customer_id):
        pass

    @abstractmethod
    def create_reservation(self, reservation_data):
        pass

    @abstractmethod
    def update_reservation(self, reservation_data):
        pass

    @abstractmethod
    def cancel_reservation(self, reservation_id):
        pass
```

- **IAdminService:**
- GetAdminById(adminId)
- GetAdminByUsername(username)
- RegisterAdmin(adminData)
- UpdateAdmin(adminData)
- DeleteAdmin(adminId)

```
from abc import ABC, abstractmethod
class IAdminService(ABC):
    @abstractmethod
    def generate_admin_id(self):
        pass
    @abstractmethod
    def get_admin_by_id(self, admin_id):
        pass

    @abstractmethod
    def get_admin_by_username(self, username):
        pass

    @abstractmethod
    def register_admin(self, admin_data):
        pass

    @abstractmethod
    def update_admin(self, admin_data):
        pass

    @abstractmethod
    def delete_admin(self, admin_id):
        pass
```

Connect your application to the SQL database:

• Create a connection string that includes the necessary information to connect to your SQL Server database. This includes the server name, database name, authentication credentials, and any other relevant settings.

• Use the SqlConnection class to establish a connection to the SQL Server database.

• Once the connection is open, you can use the SqlCommand class to execute SQL queries.

```
import mysql.connector

from exception.CustomExceptions import DatabaseConnectionException


class DBConnUtil:
    @staticmethod
    def get_connection(connection_string):
        try:
            connection = mysql.connector.connect(**connection_string)
        except DatabaseConnectionException as e:
            print(e)
        return connection
```

```
class DBPropertyUtil:
    @staticmethod
    def get_connection_string():
        host = "localhost"
        database = "CarConnect"
        user = "root"
        password = "root"

        connection_string = {
            "host": host,
            "database": database,
            "user": user,
            "password": password,
        }
        return connection_string
```

Custom Exceptions:

AuthenticationException:

• Thrown when there is an issue with user authentication.

• Example Usage: Incorrect username or password during customer or admin login.

ReservationException:

• Thrown when there is an issue with reservations.

• Example Usage: Attempting to make a reservation for a vehicle that is already reserved.

VehicleNotFoundException:

• Thrown when a requested vehicle is not found.

• Example Usage: Trying to get details of a vehicle that does not exist.

AdminNotFoundException:

• Thrown when an admin user is not found.

• Example Usage: Attempting to access details of an admin that does not exist.

InvalidInputException:

• Thrown when there is invalid input data.

• Example Usage: When a required field is missing or has an incorrect format.

DatabaseConnectionException:

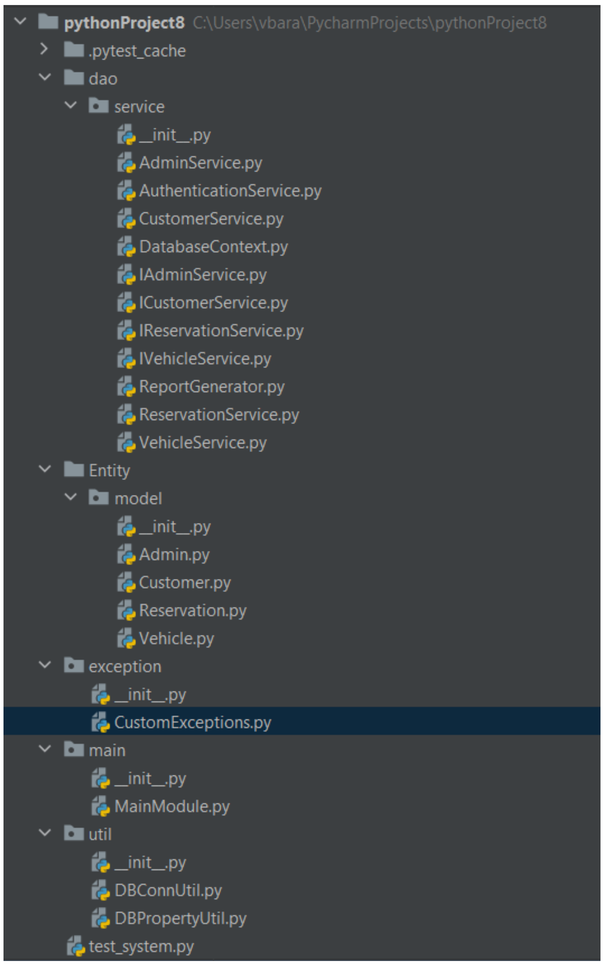
• Thrown when there is an issue with the database connection.

• Example Usage: Unable to establish a connection to the database.

Classes For Handling exceptions:

```python
class AuthenticationException(Exception):
    def __str__(self):
        return "Incorrect username or password during customer or admin login."
class ReservationException(Exception):
    def __str__(self):
        return "Attempting to make a reservation for a vehicle that is already
reserved."
class VehicleNotFoundException(Exception):
    def __str__(self):
        return "Trying to get details of a vehicle that does not exist."
class AdminNotFoundException(Exception):
    def __str__(self):
        return "Attempting to access details of an admin that does not exist"
class InvalidInputException(Exception):
    pass
class DatabaseConnectionException(Exception):
    def __str__(self):
        return 'Unable to establish a connection to the database.'
```

Project Structure:



```
pythonProject8  C:\Users\vbara\PycharmProjects\pythonProject8
  .pytest_cache
  dao
    service
      __init__.py
      AdminService.py
      AuthenticationService.py
      CustomerService.py
      DatabaseContext.py
      IAdminService.py
      ICustomerService.py
      IReservationService.py
      IVehicleService.py
      ReportGenerator.py
      ReservationService.py
      VehicleService.py
  Entity
    model
      __init__.py
      Admin.py
      Customer.py
      Reservation.py
      Vehicle.py
  exception
    __init__.py
    CustomExceptions.py
  main
    __init__.py
    MainModule.py
  util
    __init__.py
    DBConnUtil.py
    DBPropertyUtil.py
  test_system.py
```

Main Module:

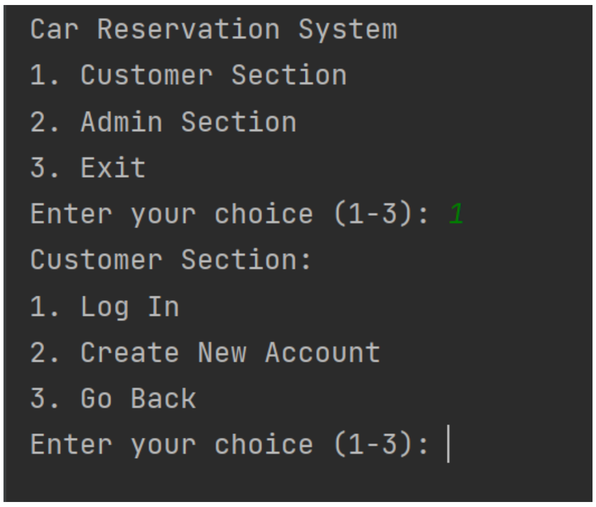
```
from dao.service.AuthenticationService import AuthenticationService
from dao.service.CustomerService import CustomerService
from datetime import date

from dao.service.ReportGenerator import ReportGenerator
from dao.service.VehicleService import VehicleService
from dao.service.ReservationService import ReservationService
from dao.service.AdminService import AdminService
from dao.service.DatabaseContext import DatabaseContext
from exception.CustomExceptions import AuthenticationException, ReservationException,
VehicleNotFoundException, \
    AdminNotFoundException, DatabaseConnectionException


def display_menu():
    print("Car Reservation System")
    print("1. Customer Section")
    print("2. Admin Section")
    print("3. Exit")


def main():
    db_context = DatabaseContext()

    try:
        db_context.connect()
    except DatabaseConnectionException as e:
        print(e)


    customer_service = CustomerService(db_context)
    vehicle_service = VehicleService(db_context)
    reservation_service = ReservationService(db_context)
    admin_service = AdminService(db_context)
    authunticate_service = AuthenticationService(db_context)
    report=ReportGenerator(db_context)
    while True:
        display_menu()
        section_choice = input("Enter your choice (1-3): ")

        if section_choice == '1':
            # Customer Section
            print("Customer Section:")
            print("1. Log In\n2. Create New Account\n3. Go Back")
            customer_auth_choice = input("Enter your choice (1-3): ")

            if customer_auth_choice == '1':
                try:
                    customer_username = input("Enter your username: ")
                    customer_password = input("Enter your password: ")

                    if authunticate_service.authenticate_user(customer_username,
customer_password):
                        print("Login successful!\n")

                        while True:
                            print("Customer Operations:")
                            customer_choice = input(
                                "1. View Your Details\n2. Update Your Details\n3.
Remove Your Account\n"
                                "4. View Vehicle Details\n5. View Available
Vehicles\n6. Reserve a Vehicle\n"
                                "7. Update Reservation \n8. Cancel Reservation 9. Log
Out\nEnter your choice: ")

                            if customer_choice == '1':
                                print(customer_username)
                                customer_details =
customer_service.get_customer_by_username(customer_username)
                                if customer_details:
                                    print("Customer Details:")
                                    print(f"ID: {customer_details[0]}")
                                    print(f"Name: {customer_details[1]}")
                                    print(f"Address: {customer_details[5]}")
                                    print(f"Phone: {customer_details[4]}")
                                    print(f"Email: {customer_details[3]}")
                                else:
                                    print("Customer not found!")



                            elif customer_choice == '2':
                                update_customer = {
                                    "customer_id": input("Enter the customer id to
update: "),
                                    "first_name": input("Enter the first name to
update: "),
                                    "last_name": input("Enter the last name to update:
"),
                                    "email": input("Enter the email to update: "),
                                    "phone_number": input("Enter the phone_number to 
update: "),
                                    "address": input("Enter the address to update: "),
                                    "username": input("Enter the username to update: 
"),
                                    "password": input("Enter the password to update: ")
                                }
                                customer_service.update_customer(update_customer)
                                print("Customer updated successfully!")

                            elif customer_choice == '3':
                                customer_id = int(input("Enter the customer ID: "))
                                print("Enter your username and password to confirm 
deletion...!")
                                username = input("Enter your username:")
                                password = input("Enter your password :")
                                if authunticate_service.authenticate_user(username, 
password):
                                    customer_service.delete_customer(customer_id)
                                    print("Your details have been deleted 
successfully....")
                                else:
                                    print("Invalid username or password!!!")

                            elif customer_choice == '4':
                                vehicle_id = int(input("Enter the vehicle id: "))
                                try:
                                    details = 
vehicle_service.get_vehicle_by_id(vehicle_id)
                                    print("Model: ", details[1])
                                    print("Make: ", details[2])
                                    print("Year: ", details[3])
                                    print("Color: ", details[4])
                                    print("RegistrationNumber: ", details[5])
                                    print("Availability: ", details[6])
                                    print("DailyRate: ", details[7])
                                except VehicleNotFoundException as e:
                                    print(e)
                            elif customer_choice == '5':
                                res = vehicle_service.get_available_vehicles()
                                print("Available Vehicles")
                                print(res)
                            elif customer_choice == '6':
                                reservation_data = {
                                    "CustomerID": input("Enter customer ID: "),
                                    "VehicleID": input("Enter Vehicle ID: "),
                                    "StartDate": date.today(),
                                    "EndDate": input("Enter EndDate: "),
                                    "TotalCost": input("Enter TotalCost: "),
                                    "Status": input("Enter Status: ")
                                }
                                try:

reservation_service.create_reservation(reservation_data)
                                except ReservationException as e:
                                    print(e)

                            elif customer_choice == '7':
                                update_reservation_data = {
                                    "ReservationID": input("Enter the reservation ID to
be updated:"),
                                    "CustomerID": input("Enter customer ID: "),
                                    "VehicleID": input("Enter Vehicle ID: "),
                                    "StartDate": date.today(),
                                    "EndDate": input("Enter EndDate: "),
                                    "TotalCost": input("Enter TotalCost: "),
                                    "Status": input("Enter Status: ")
                                }


reservation_service.update_reservation(update_reservation_data)
                            elif customer_choice == '8':
                                res_id = input("Enter your reservation ID: ")
                                reservation_service.cancel_reservation(res_id)
                                print("Your reservation have been cancelled
sucessfully....")

                            elif customer_choice == '9':
                                print("Logging out...\n")
                                break

                except AuthenticationException as e:
                    print(e)



            elif customer_auth_choice == '2':

                customer_details = {
                    'FirstName': input("Enter your First Name:"),
                    'LastName': input("Enter your Last Name:"),
                    'Email': input("Enter your Email:"),
                    'PhoneNumber': input("Enter your PhoneNumber:"),
                    'Address': input("Enter your Address:"),
                    'Username': input("Enter Username:"),
                    'Password': input("Enter your Password:"),
                    'RegistrationDate': date.today()
                }

                customer_service.register_customer(customer_details)
                print("Account created successfully!\n")

            elif customer_auth_choice == '3':

                print("Going back to main menu...\n")

            else:
                print("Invalid choice. Please enter a number between 1 and 3.\n")

        elif section_choice == '2':
            try:
                admin_username = input("Enter your username: ")
                admin_password = input("Enter your password: ")
                if authunticate_service.authenticate_admin(admin_username,
admin_password):
                    print("Login Successfull...!!")
                    while True:
                        print("Admin Operations:")
                        admin_choice = input(
                            "1. View Your Profile\n2. Register New Admin\n3. Update
Profile\n"
                            "4. Remove Account\n5. View Reservation Report\n"
                            "6. Add Vehicle \n7. Update Vehicle Details\n"
                            "8. Remove Vehicle \n9.Log Out\nEnter your choice(1-7): ")

                        if admin_choice == '1':
                            try:
                                details =
admin_service.get_admin_by_username(admin_username)
                                print("Admin ID: ", details[0])
                                print("Name: ", details[1])
                                print("Email: ", details[3])
                                print("PhoneNumber: ", details[4])
                                print("Username: ", details[5])
                                print("Password: ", details[6])
                                print("Role: ", details[7])
                                print("Join Date: ", details[8])
                            except AdminNotFoundException as e:
                                print(e)




                        elif admin_choice == '2':

                            admin_details = {
                                'FirstName': input("Enter admin's First Name:"),
                                'LastName': input("Enter admin's Last Name:"),
                                'Email': input("Enter admin's Email:"),
                                'PhoneNumber': input("Enter admin's PhoneNumber:"),
                                'Username': input("Enter admin's Username:"),
                                'Password': input("Enter admin's Password:"),
                                'Role': input("Enter admin's Role:"),
                                'JoinDate': date.today()
                            }
                            admin_service.register_admin(admin_details)
                            print("Admin registration successful.")

                        elif admin_choice == '3':
                            update_details = {
                                'AdminID': input("Enter your ID to be updated:"),
                                'FirstName': input("Enter your First Name:"),
                                'LastName': input("Enter your Last Name:"),
                                'Email': input("Enter your Email:"),
                                'PhoneNumber': input("Enter your PhoneNumber:"),
                                'Username': input("Enter your Username:"),
                                'Password': input("Enter your Password:"),
                                'Role': input("Enter your Role:"),
                                'JoinDate': date.today()
                            }
                            admin_service.update_admin(update_details)
                            print("Updated Sucessfully....")

                        elif admin_choice == '4':
                            admin_id = input("Enter your ID : ")
                            admin_service.delete_admin(admin_id)
                            print("Admin removed!!!")

                        elif admin_choice == '5':
                            res_id = input("Enter the reservation ID:")
                            report.generate_report(res_id)

                        elif admin_choice == '6':
                            vehicle_details = {
                                "Model": input("Enter the model: "),
                                "Make": input("Enter the Make: "),
                                "Year": input("Enter the Year: "),
                                "Color": input("Enter the Color: "),
                                "RegistrationNumber": input("Enter the
RegistrationNumber: "),
                                "Availability": input("Enter the Availability: "),
                                "DailyRate": input("Enter the DailyRate: ")

                            }
                            vehicle_service.add_vehicle(vehicle_details)
                            print("Vehicle added sucessfully...")

                        elif admin_choice == '7':
                            update_vehicle = {
                                "vehicle_id": input("Enter the vehicle id whose data to
be updated: "),
                                "Model": input("Enter the model : "),
                                "Make": input("Enter the Make: "),
                                "Year": input("Enter the Year: "),
                                "Color": input("Enter the Color: "),
                                "RegistrationNumber": input("Enter the
RegistrationNumber: "),
                                "Availability": input("Enter the Availability: "),
                                "DailyRate": input("Enter the DailyRate: ")

                            }
                            vehicle_service.update_vehicle(update_vehicle)
                            print("The vehicle datas have been updated
successfully.....")

                        elif admin_choice == '8':
                            vehicle_id = int(input("Enter the vehicle ID: "))
                            vehicle_service.remove_vehicle(vehicle_id)
                            print("This vehicle have been removed sucessfully....")

                        elif admin_choice == '9':
                            print("Logged Out...!!")
                            break
            except AuthenticationException as e:
                print(e)
```

```
        elif section_choice == '3':
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 3.")

    db_context.connection.close()


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

Implementation:

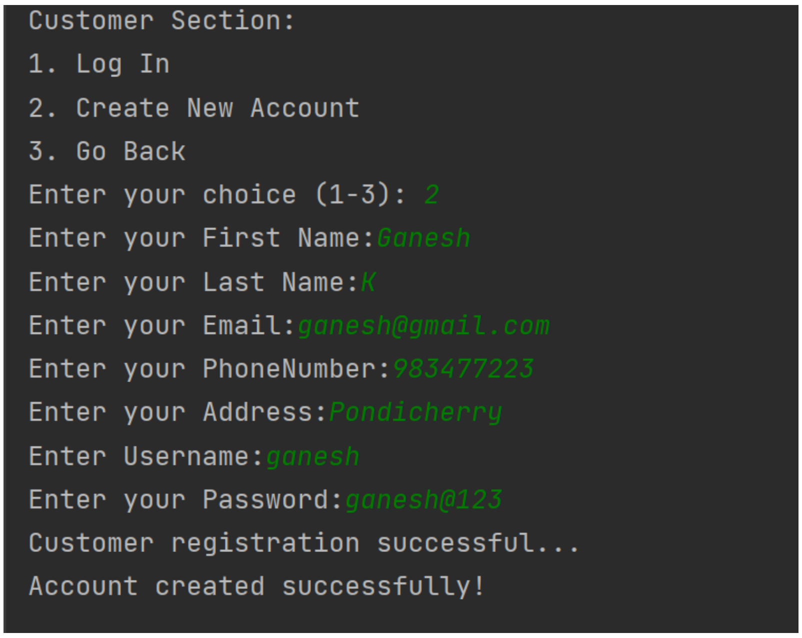
I have divided my application into two parts: the Customer and Admin sections. The Customer section is accessible only to registered customers. To register, customers can enter their details, and thereafter, they can log in using their username and password.

Choosing option 1 allows entry into the customer section. If the user doesn't have an account, the system prompts them to create a new one.



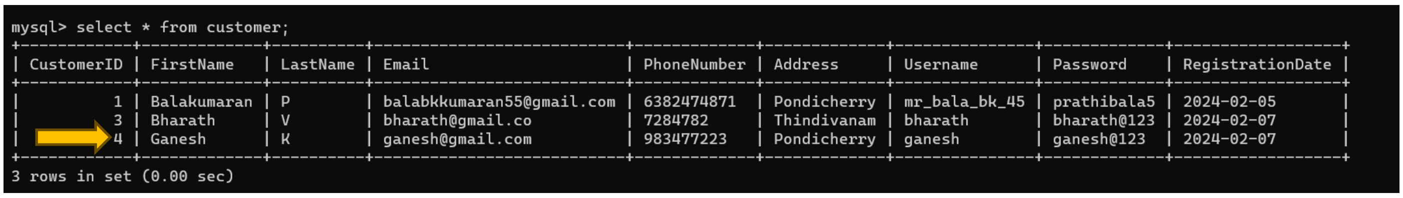
```
Car Reservation System
1. Customer Section
2. Admin Section
3. Exit
Enter your choice (1-3): 1
Customer Section:
1. Log In
2. Create New Account
3. Go Back
Enter your choice (1-3):
```

To create a new account for a customer, they need to provide a unique username and a secure password during the registration process.



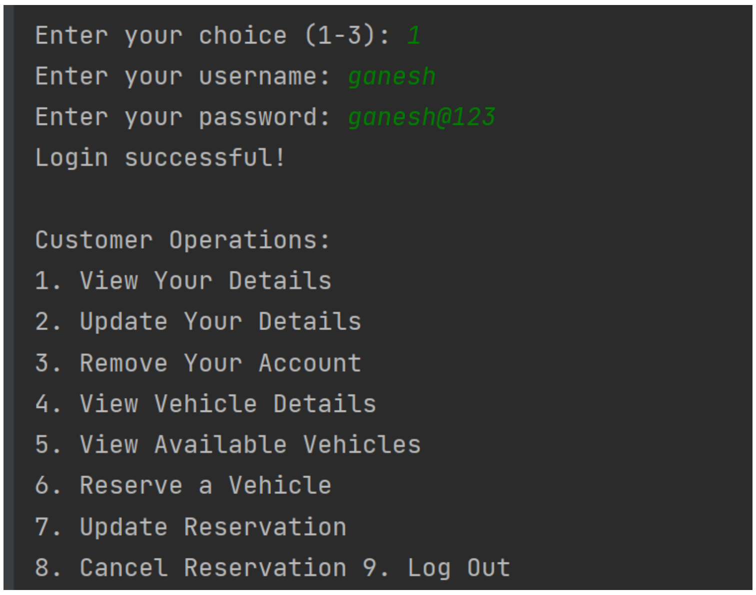
```
Customer Section:
1. Log In
2. Create New Account
3. Go Back
Enter your choice (1-3): 2
Enter your First Name:Ganesh
Enter your Last Name:K
Enter your Email:ganesh@gmail.com
Enter your PhoneNumber:983477223
Enter your Address:Pondicherry
Enter Username:ganesh
Enter your Password:ganesh@123
Customer registration successful...
Account created successfully!
```

A new customer with customer ID: 4 has been created successfully and inserted into the database:



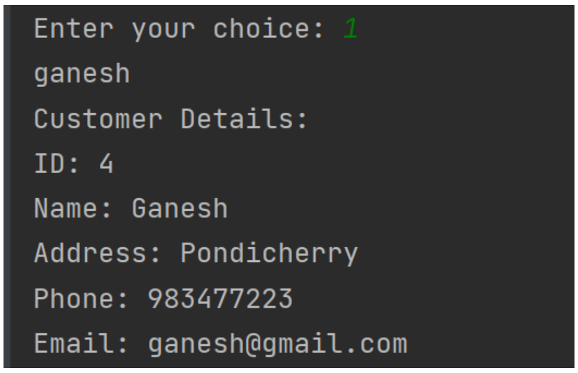
```
mysql> select * from customer;
+------------+-------------+----------+---------------------------+-------------+-------------+---------------+--------------+------------------+
| CustomerID | FirstName   | LastName | Email                     | PhoneNumber | Address     | Username      | Password     | RegistrationDate |
+------------+-------------+----------+---------------------------+-------------+-------------+---------------+--------------+------------------+
|          1 | Balakumaran | P        | balabkkumaran55@gmail.com | 6382474871  | Pondicherry | mr_bala_bk_45 | prathibala5  | 2024-02-05       |
|          3 | Bharath     | V        | bharath@gmail.co          | 7284782     | Thindivanam | bharath       | bharath@123  | 2024-02-07       |
|          4 | Ganesh      | K        | ganesh@gmail.com          | 983477223   | Pondicherry | ganesh        | ganesh@123   | 2024-02-07       |
+------------+-------------+----------+---------------------------+-------------+-------------+---------------+--------------+------------------+
3 rows in set (0.00 sec)
```

Attempting to log in with the provided username and password. Upon successful login, the system displays various options that the customer can choose to perform different operations.



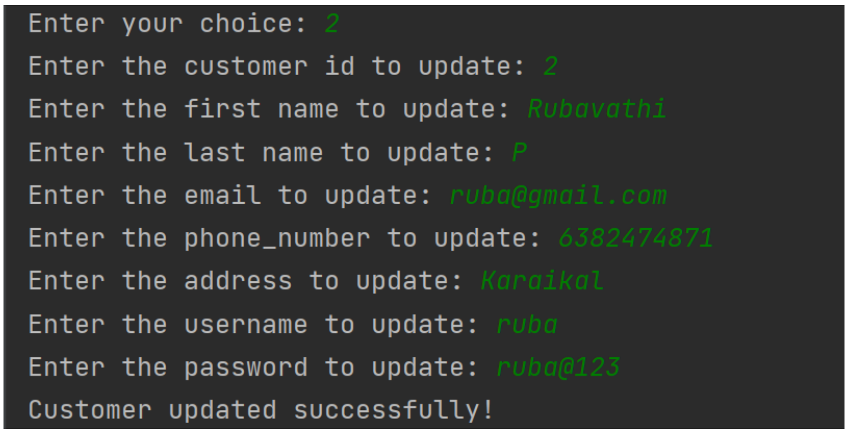
```
Enter your choice (1-3): 1
Enter your username: ganesh
Enter your password: ganesh@123
Login successful!

Customer Operations:
1. View Your Details
2. Update Your Details
3. Remove Your Account
4. View Vehicle Details
5. View Available Vehicles
6. Reserve a Vehicle
7. Update Reservation
8. Cancel Reservation 9. Log Out
```

Selecting option 1 from the customer menu to list all details of the current user.



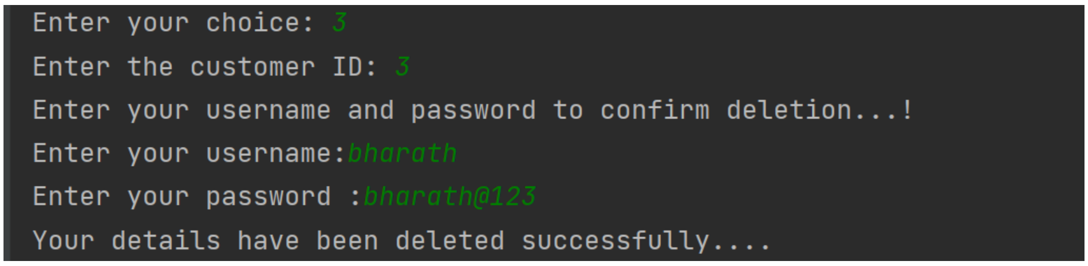
```
Enter your choice: 1
ganesh
Customer Details:
ID: 4
Name: Ganesh
Address: Pondicherry
Phone: 983477223
Email: ganesh@gmail.com
```

Selecting option 2 from the customer menu to update a particular customer details



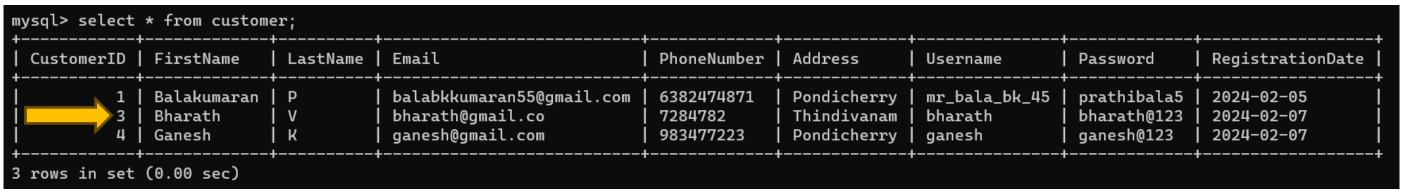
```
Enter your choice: 2
Enter the customer id to update: 2
Enter the first name to update: Rubavathi
Enter the last name to update: P
Enter the email to update: ruba@gmail.com
Enter the phone_number to update: 6382474871
Enter the address to update: Karaikal
Enter the username to update: ruba
Enter the password to update: ruba@123
Customer updated successfully!
```

Selecting option 3 from the customer menu allows the user to delete a specific account from the database. The customer can proceed with the deletion only by providing their username and password correctly.



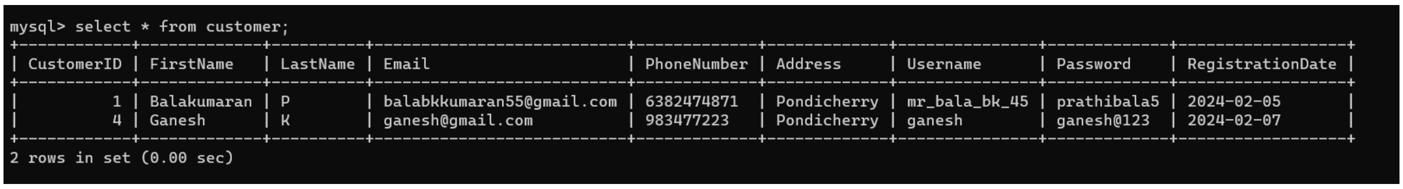
```
Enter your choice: 3
Enter the customer ID: 3
Enter your username and password to confirm deletion...!
Enter your username:bharath
Enter your password :bharath@123
Your details have been deleted successfully....
```

Before deleting the account for the customer ID 3:



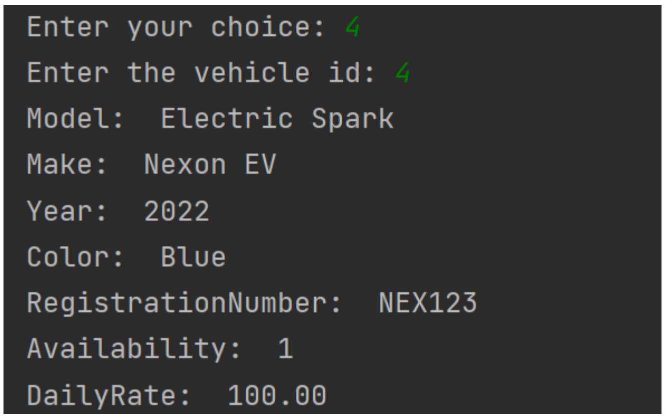
```
mysql> select * from customer;
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
| CustomerID | FirstName   | LastName | Email                     | PhoneNumber | Address     | Username       | Password     | RegistrationDate |
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
|          1 | Balakumaran | P        | balabkkumaran55@gmail.com | 6382474871  | Pondicherry | mr_bala_bk_45  | prathibala5  | 2024-02-05       |
|          3 | Bharath     | V        | bharath@gmail.co          | 7284782     | Thindivanam | bharath        | bharath@123  | 2024-02-07       |
|          4 | Ganesh      | K        | ganesh@gmail.com          | 983477223   | Pondicherry | ganesh         | ganesh@123   | 2024-02-07       |
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
3 rows in set (0.00 sec)
```

After the deletion process, the record has been successfully removed from the database.

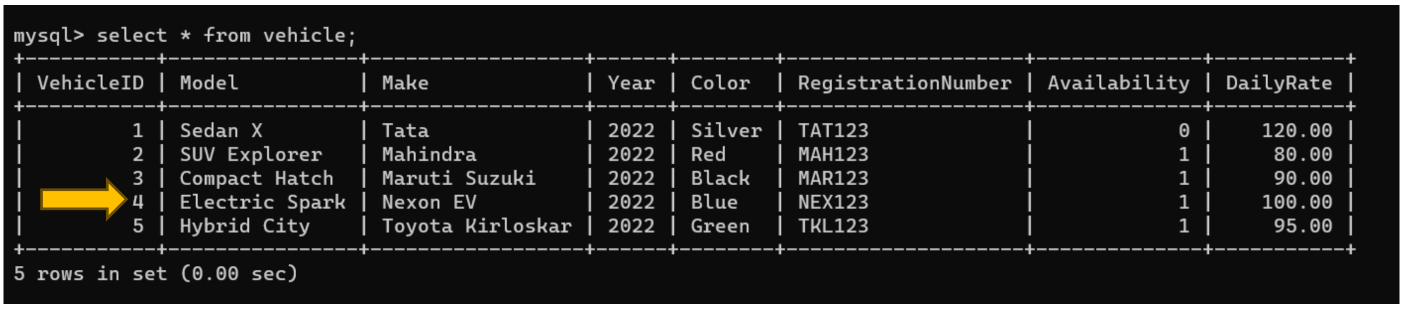


```
mysql> select * from customer;
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
| CustomerID | FirstName   | LastName | Email                     | PhoneNumber | Address     | Username       | Password     | RegistrationDate |
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
|          1 | Balakumaran | P        | balabkkumaran55@gmail.com | 6382474871  | Pondicherry | mr_bala_bk_45  | prathibala5  | 2024-02-05       |
|          4 | Ganesh      | K        | ganesh@gmail.com          | 983477223   | Pondicherry | ganesh         | ganesh@123   | 2024-02-07       |
+------------+-------------+----------+---------------------------+-------------+-------------+----------------+--------------+------------------+
2 rows in set (0.00 sec)
```

Selecting option 4 from the customer menu prompts the user to provide the ID of a specific vehicle to view its details.

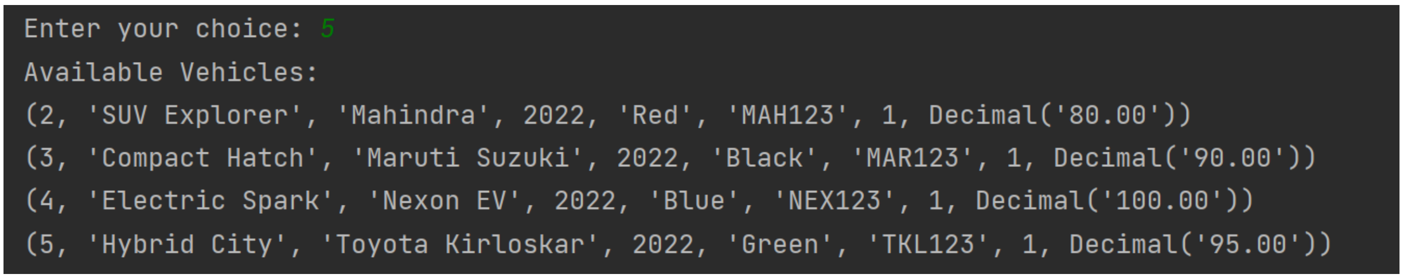


```
Enter your choice: 4
Enter the vehicle id: 4
Model:  Electric Spark
Make:  Nexon EV
Year:  2022
Color:  Blue
RegistrationNumber:  NEX123
Availability:  1
DailyRate:  100.00
```



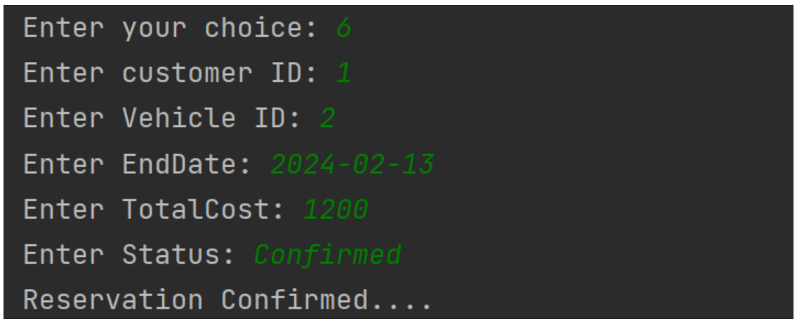
```
mysql> select * from vehicle;
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
| VehicleID | Model          | Make             | Year | Color  | RegistrationNumber | Availability | DailyRate |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
|         1 | Sedan X        | Tata             | 2022 | Silver | TAT123             |            0 |    120.00 |
|         2 | SUV Explorer   | Mahindra         | 2022 | Red    | MAH123             |            1 |     80.00 |
|         3 | Compact Hatch  | Maruti Suzuki    | 2022 | Black  | MAR123             |            1 |     90.00 |
|         4 | Electric Spark | Nexon EV         | 2022 | Blue   | NEX123             |            1 |    100.00 |
|         5 | Hybrid City    | Toyota Kirloskar | 2022 | Green  | TKL123             |            1 |     95.00 |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
5 rows in set (0.00 sec)
```

Selecting option 5 to list all the vehicles that are available for the reservation.



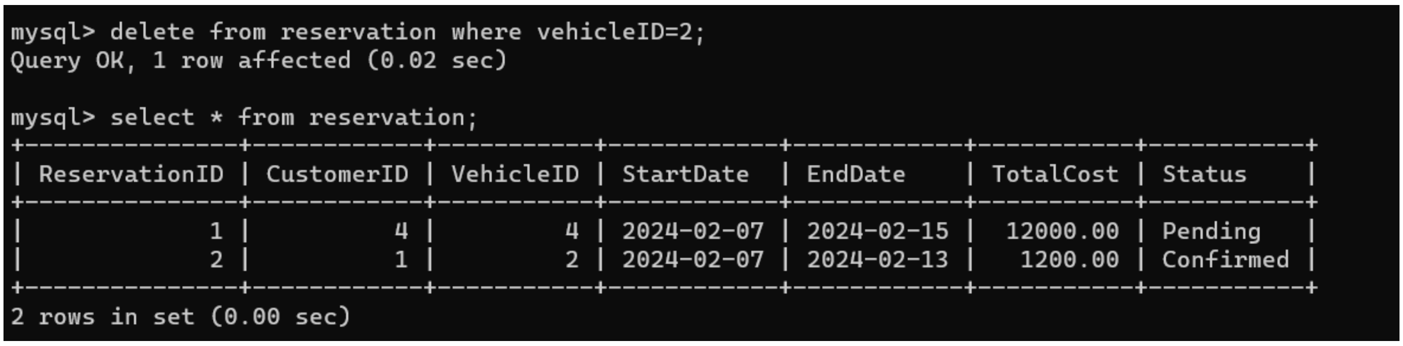
```
Enter your choice: 5
Available Vehicles:
(2, 'SUV Explorer', 'Mahindra', 2022, 'Red', 'MAH123', 1, Decimal('80.00'))
(3, 'Compact Hatch', 'Maruti Suzuki', 2022, 'Black', 'MAR123', 1, Decimal('90.00'))
(4, 'Electric Spark', 'Nexon EV', 2022, 'Blue', 'NEX123', 1, Decimal('100.00'))
(5, 'Hybrid City', 'Toyota Kirloskar', 2022, 'Green', 'TKL123', 1, Decimal('95.00'))
```

Selecting option 6 allows the user to register a reservation, requiring the respective customer ID and the available vehicle ID.



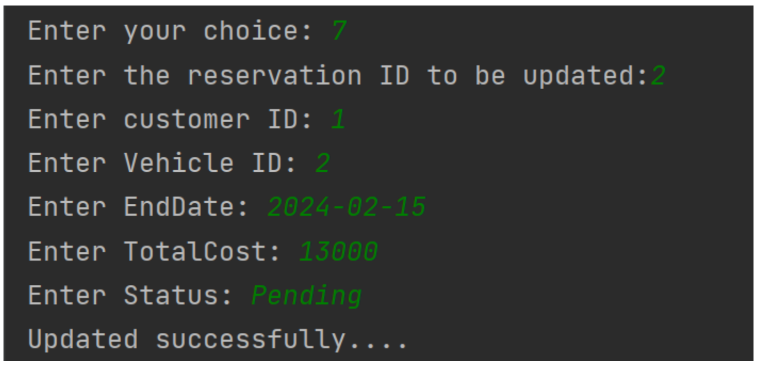
```
Enter your choice: 6
Enter customer ID: 1
Enter Vehicle ID: 2
Enter EndDate: 2024-02-13
Enter TotalCost: 1200
Enter Status: Confirmed
Reservation Confirmed....
```

The reservation history has been successfully inserted into the database.

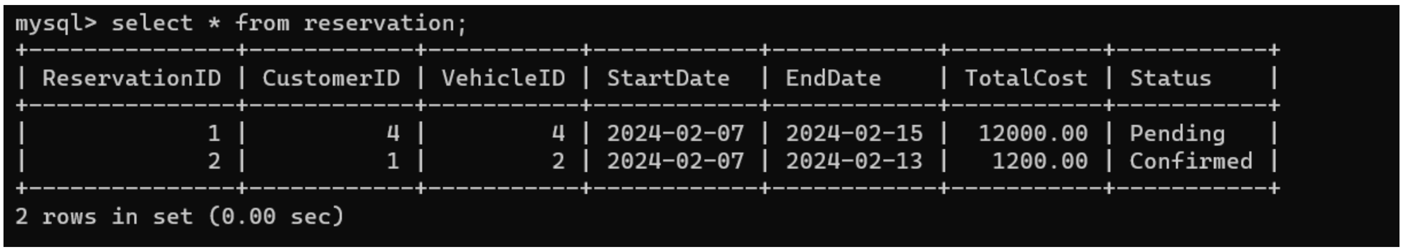


```
mysql> delete from reservation where vehicleID=2;
Query OK, 1 row affected (0.02 sec)

mysql> select * from reservation;
+---------------+------------+-----------+------------+------------+-----------+-----------+
| ReservationID | CustomerID | VehicleID | StartDate  | EndDate    | TotalCost | Status    |
+---------------+------------+-----------+------------+------------+-----------+-----------+
|             1 |          4 |         4 | 2024-02-07 | 2024-02-15 |  12000.00 | Pending   |
|             2 |          1 |         2 | 2024-02-07 | 2024-02-13 |   1200.00 | Confirmed |
+---------------+------------+-----------+------------+------------+-----------+-----------+
2 rows in set (0.00 sec)
```

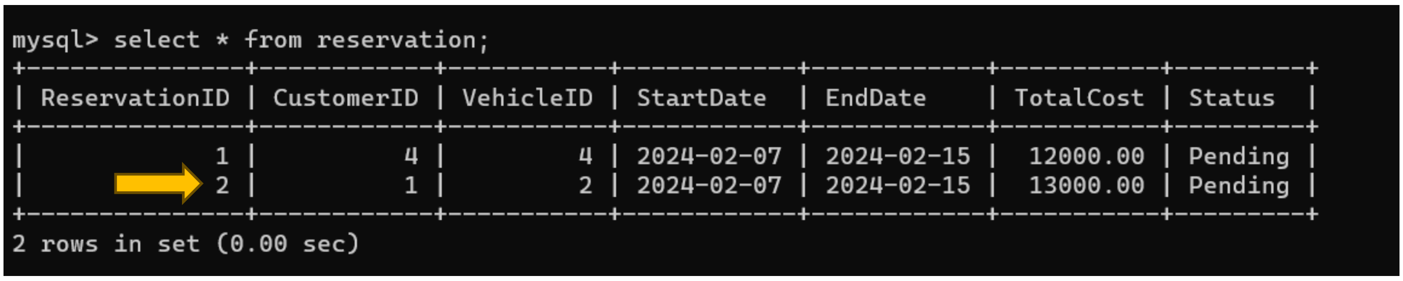
Updating a particular reservation details :


```
Enter your choice: 7
Enter the reservation ID to be updated:2
Enter customer ID: 1
Enter Vehicle ID: 2
Enter EndDate: 2024-02-15
Enter TotalCost: 13000
Enter Status: Pending
Updated successfully....
```

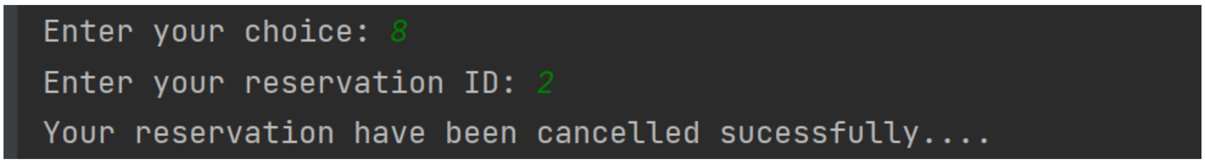
Updating the reservation details for ID 2 by modifying its end date and status.


```
mysql> select * from reservation;
+---------------+------------+-----------+------------+------------+-----------+-----------+
| ReservationID | CustomerID | VehicleID | StartDate  | EndDate    | TotalCost | Status    |
+---------------+------------+-----------+------------+------------+-----------+-----------+
|             1 |          4 |         4 | 2024-02-07 | 2024-02-15 |  12000.00 | Pending   |
|             2 |          1 |         2 | 2024-02-07 | 2024-02-13 |   1200.00 | Confirmed |
+---------------+------------+-----------+------------+------------+-----------+-----------+
2 rows in set (0.00 sec)
```

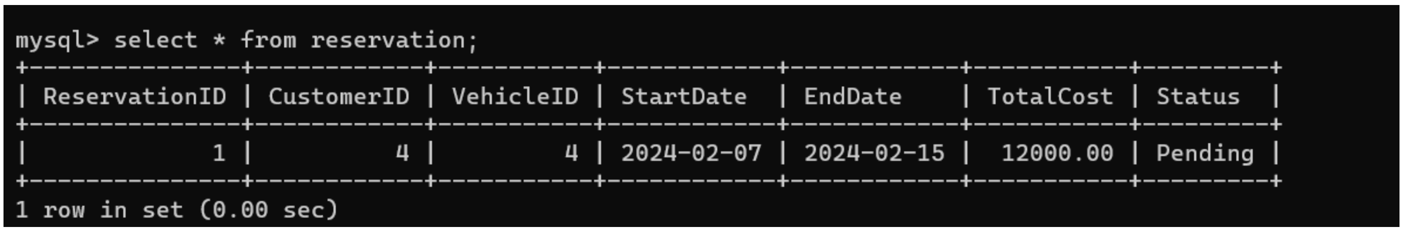
After update:


```
mysql> select * from reservation;
+---------------+------------+-----------+------------+------------+-----------+---------+
| ReservationID | CustomerID | VehicleID | StartDate  | EndDate    | TotalCost | Status  |
+---------------+------------+-----------+------------+------------+-----------+---------+
|             1 |          4 |         4 | 2024-02-07 | 2024-02-15 |  12000.00 | Pending |
|             2 |          1 |         2 | 2024-02-07 | 2024-02-15 |  13000.00 | Pending |
+---------------+------------+-----------+------------+------------+-----------+---------+
2 rows in set (0.00 sec)
```

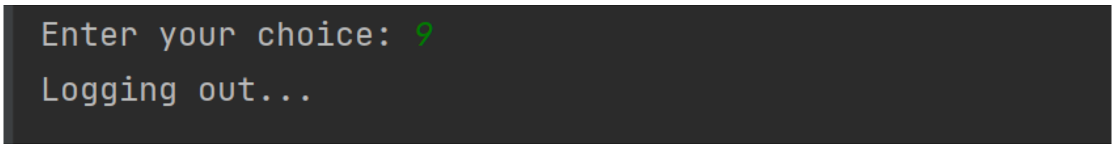
Cancelling reservation :


```
Enter your choice: 8
Enter your reservation ID: 2
Your reservation have been cancelled sucessfully....
```

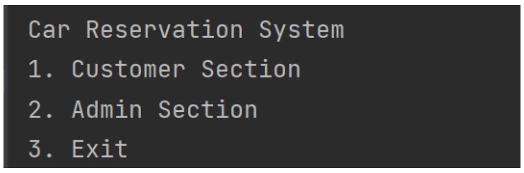
After canceling the reservation, the changes have been updated in the reservation table of the database.


```
mysql> select * from reservation;
+---------------+------------+-----------+------------+------------+-----------+---------+
| ReservationID | CustomerID | VehicleID | StartDate  | EndDate    | TotalCost | Status  |
+---------------+------------+-----------+------------+------------+-----------+---------+
|             1 |          4 |         4 | 2024-02-07 | 2024-02-15 |  12000.00 | Pending |
+---------------+------------+-----------+------------+------------+-----------+---------+
1 row in set (0.00 sec)
```

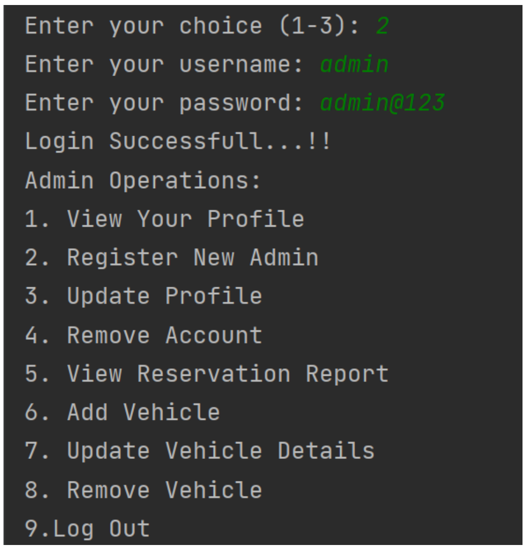
Selecting option 9 to log out from the customer section:


```
Enter your choice: 9
Logging out...
```

Selecting option 2 from the Car Reservation System leads to entering the admin section. Upon entry, the system prompts for the admin username and password. Access to the admin section is granted only after providing the correct credentials. Upon successful login, the system displays various options for the admin to perform.

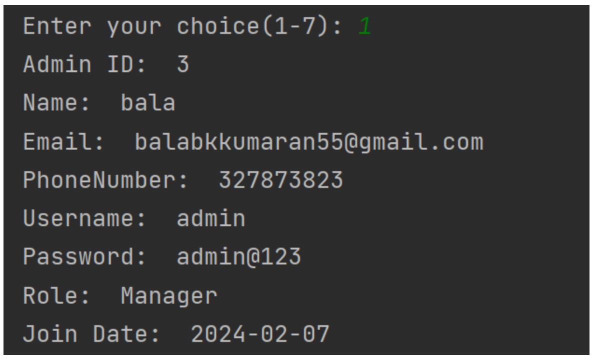


```
Car Reservation System
1. Customer Section
2. Admin Section
3. Exit
```



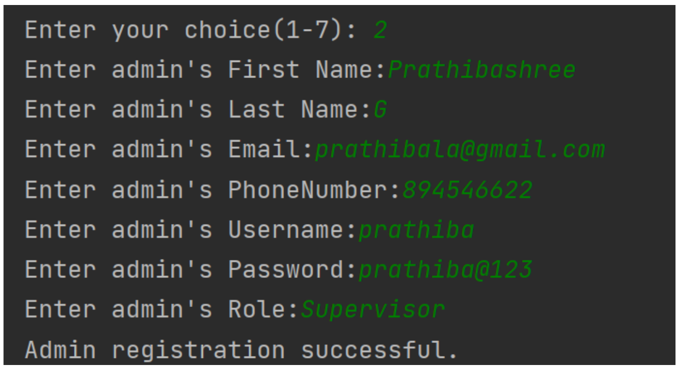
```
Enter your choice (1-3): 2
Enter your username: admin
Enter your password: admin@123
Login Successfull...!!
Admin Operations:
1. View Your Profile
2. Register New Admin
3. Update Profile
4. Remove Account
5. View Reservation Report
6. Add Vehicle
7. Update Vehicle Details
8. Remove Vehicle
9.Log Out
```

Selecting option 1 from the Admin operation menu allows viewing the details of the current admin who is logged in.



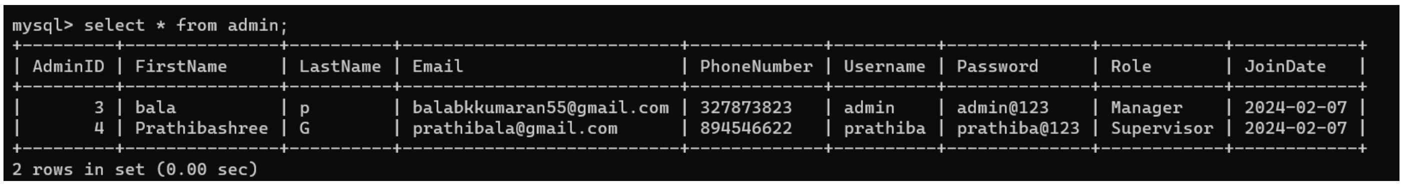
```
Enter your choice(1-7): 1
Admin ID:  3
Name:  bala
Email:  balabkkumaran55@gmail.com
PhoneNumber:  327873823
Username:  admin
Password:  admin@123
Role:  Manager
Join Date:  2024-02-07
```

Selecting option 1 from the Admin operation menu allows creating new admin



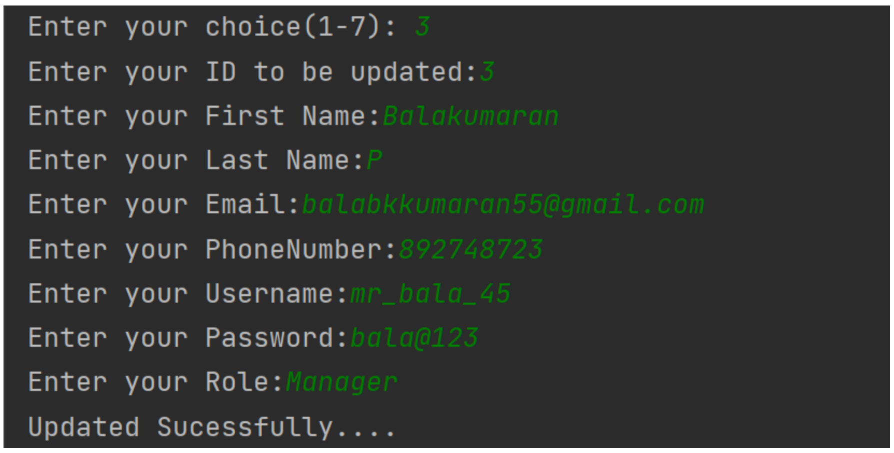
```
Enter your choice(1-7): 2
Enter admin's First Name:Prathibashree
Enter admin's Last Name:G
Enter admin's Email:prathibala@gmail.com
Enter admin's PhoneNumber:894546622
Enter admin's Username:prathiba
Enter admin's Password:prathiba@123
Enter admin's Role:Supervisor
Admin registration successful.
```

The new admin details have been added, and the joining date is set to the current date.



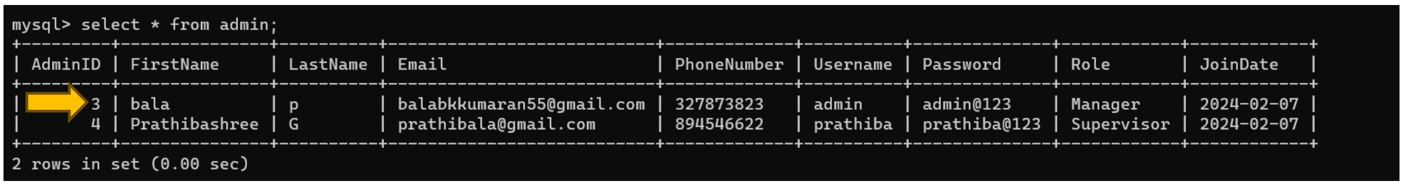
```
mysql> select * from admin;
+---------+---------------+----------+---------------------------+-------------+----------+--------------+------------+------------+
| AdminID | FirstName     | LastName | Email                     | PhoneNumber | Username | Password     | Role       | JoinDate   |
+---------+---------------+----------+---------------------------+-------------+----------+--------------+------------+------------+
|       3 | bala          | p        | balabkkumaran55@gmail.com | 327873823   | admin    | admin@123    | Manager    | 2024-02-07 |
|       4 | Prathibashree | G        | prathibala@gmail.com      | 894546622   | prathiba | prathiba@123 | Supervisor | 2024-02-07 |
+---------+---------------+----------+---------------------------+-------------+----------+--------------+------------+------------+
2 rows in set (0.00 sec)
```

Selecting option 3 to update the admin details:



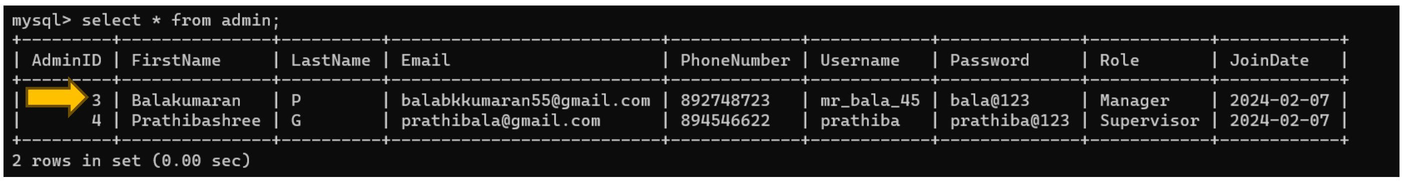
```
Enter your choice(1-7): 3
Enter your ID to be updated:3
Enter your First Name:Balakumaran
Enter your Last Name:P
Enter your Email:balabkkumaran55@gmail.com
Enter your PhoneNumber:892748723
Enter your Username:mr_bala_45
Enter your Password:bala@123
Enter your Role:Manager
Updated Sucessfully....
```

Before updating the admin with ID 3 by modifying their username and password:



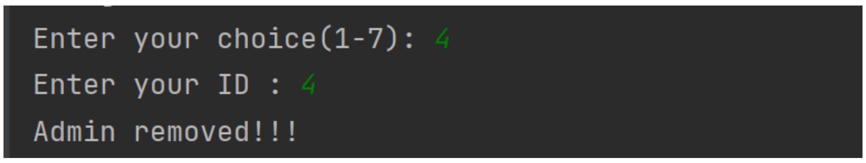
```
mysql> select * from admin;
+---------+--------------+----------+--------------------------+-------------+----------+--------------+------------+------------+
| AdminID | FirstName    | LastName | Email                    | PhoneNumber | Username | Password     | Role       | JoinDate   |
+---------+--------------+----------+--------------------------+-------------+----------+--------------+------------+------------+
|       3 | bala         | p        | balabkkumaran55@gmail.com | 327873823  | admin    | admin@123    | Manager    | 2024-02-07 |
|       4 | Prathibashree | G       | prathibala@gmail.com     | 894546622   | prathiba | prathiba@123 | Supervisor | 2024-02-07 |
+---------+--------------+----------+--------------------------+-------------+----------+--------------+------------+------------+
2 rows in set (0.00 sec)
```

After Updating:



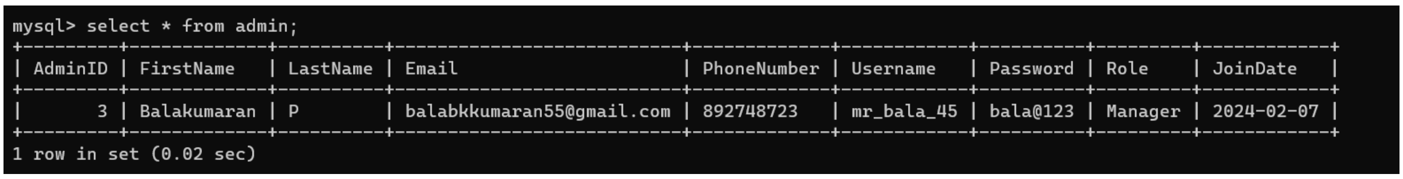
```
mysql> select * from admin;
+---------+---------------+----------+---------------------------+-------------+------------+--------------+------------+------------+
| AdminID | FirstName     | LastName | Email                     | PhoneNumber | Username   | Password     | Role       | JoinDate   |
+---------+---------------+----------+---------------------------+-------------+------------+--------------+------------+------------+
|       3 | Balakumaran   | P        | balabkkumaran55@gmail.com | 892748723   | mr_bala_45 | bala@123     | Manager    | 2024-02-07 |
|       4 | Prathibashree | G        | prathibala@gmail.com      | 894546622   | prathiba   | prathiba@123 | Supervisor | 2024-02-07 |
+---------+---------------+----------+---------------------------+-------------+------------+--------------+------------+------------+
2 rows in set (0.00 sec)
```

Selecting option 4 to remove or delete a particular admin by providing their ID:



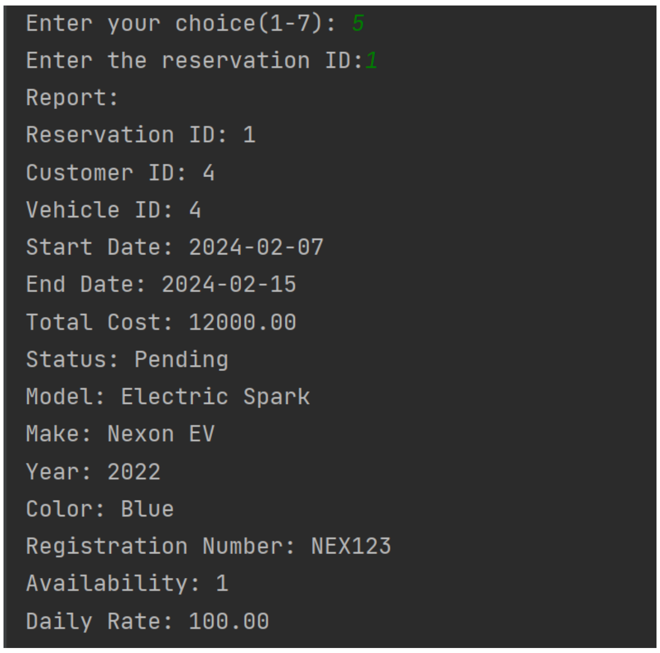
```
Enter your choice(1-7): 4
Enter your ID : 4
Admin removed!!!
```

After removing it successfully updated the same in the database:



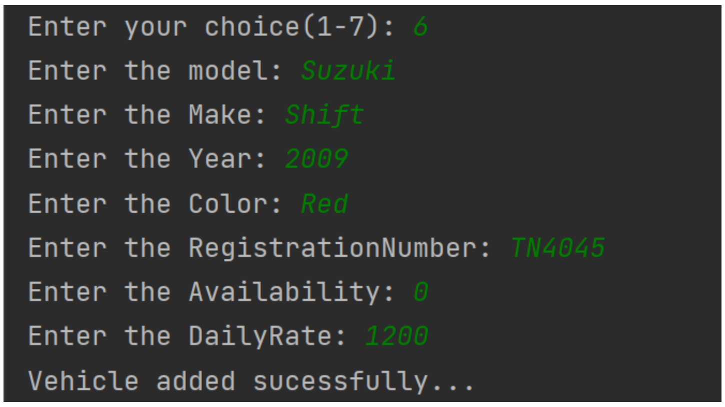
```
mysql> select * from admin;
+---------+-------------+----------+---------------------------+-------------+------------+----------+---------+------------+
| AdminID | FirstName   | LastName | Email                     | PhoneNumber | Username   | Password | Role    | JoinDate   |
+---------+-------------+----------+---------------------------+-------------+------------+----------+---------+------------+
|       3 | Balakumaran | P        | balabkkumaran55@gmail.com | 892748723   | mr_bala_45 | bala@123 | Manager | 2024-02-07 |
+---------+-------------+----------+---------------------------+-------------+------------+----------+---------+------------+
1 row in set (0.02 sec)
```

Selecting option 5 to generate a report for a particular reservation :

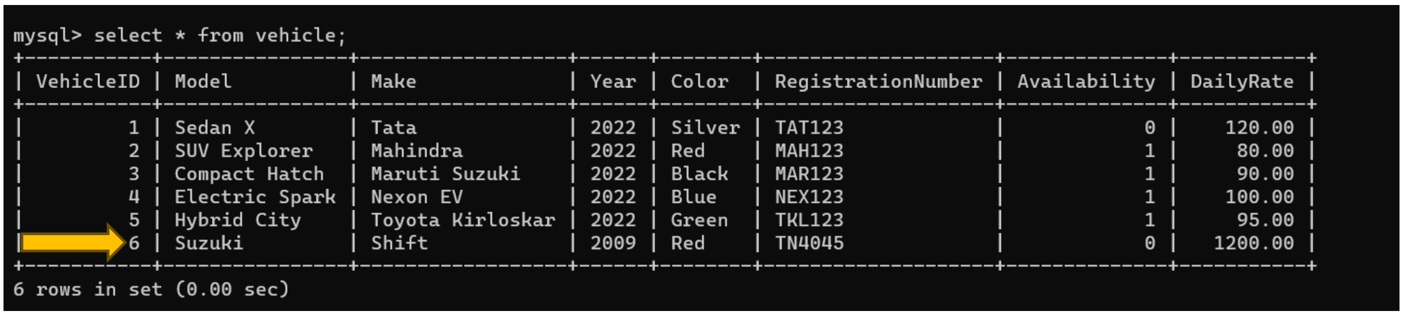


```
Enter your choice(1-7): 5
Enter the reservation ID:1
Report:
Reservation ID: 1
Customer ID: 4
Vehicle ID: 4
Start Date: 2024-02-07
End Date: 2024-02-15
Total Cost: 12000.00
Status: Pending
Model: Electric Spark
Make: Nexon EV
Year: 2022
Color: Blue
Registration Number: NEX123
Availability: 1
Daily Rate: 100.00
```

Adding vehicle to the databse:

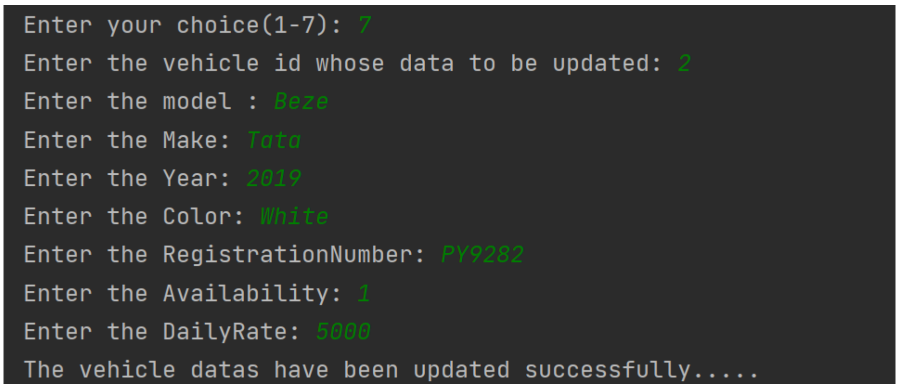


```
Enter your choice(1-7): 6
Enter the model: Suzuki
Enter the Make: Shift
Enter the Year: 2009
Enter the Color: Red
Enter the RegistrationNumber: TN4045
Enter the Availability: 0
Enter the DailyRate: 1200
Vehicle added sucessfully...
```



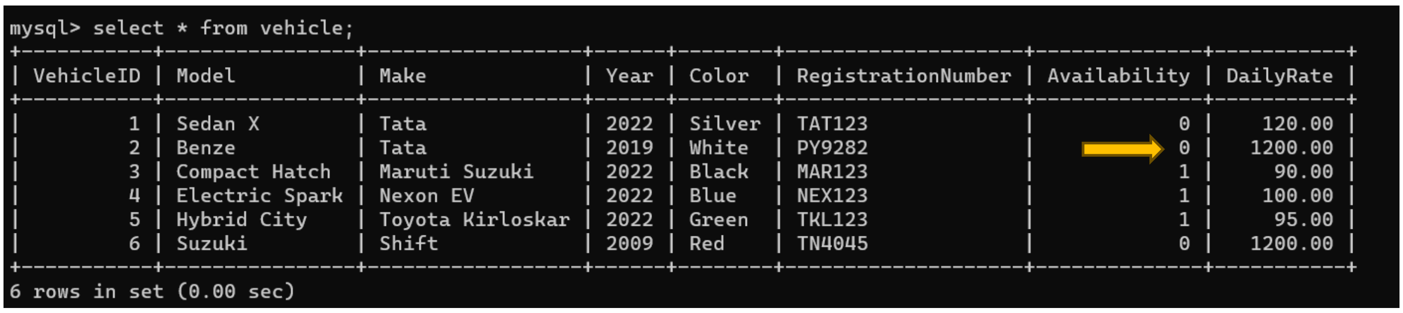
```
mysql> select * from vehicle;
+-----------+----------------+-------------------+------+--------+--------------------+--------------+-----------+
| VehicleID | Model          | Make              | Year | Color  | RegistrationNumber | Availability | DailyRate |
+-----------+----------------+-------------------+------+--------+--------------------+--------------+-----------+
|         1 | Sedan X        | Tata              | 2022 | Silver | TAT123             |            0 |    120.00 |
|         2 | SUV Explorer   | Mahindra          | 2022 | Red    | MAH123             |            1 |     80.00 |
|         3 | Compact Hatch  | Maruti Suzuki     | 2022 | Black  | MAR123             |            1 |     90.00 |
|         4 | Electric Spark | Nexon EV          | 2022 | Blue   | NEX123             |            1 |    100.00 |
|         5 | Hybrid City    | Toyota Kirloskar  | 2022 | Green  | TKL123             |            1 |     95.00 |
|         6 | Suzuki         | Shift             | 2009 | Red    | TN4045             |            0 |   1200.00 |
+-----------+----------------+-------------------+------+--------+--------------------+--------------+-----------+
6 rows in set (0.00 sec)
```

Selecting option 7 to update a particular vehicle by giving their ID:



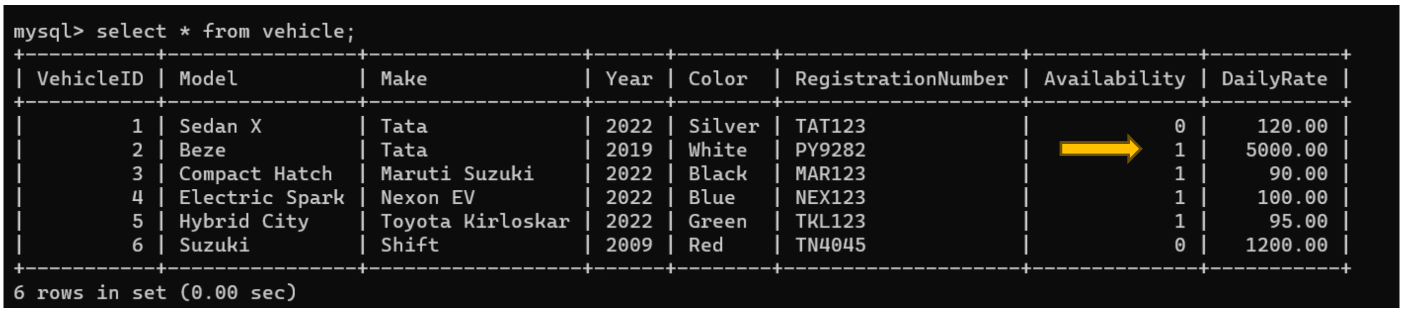
```
Enter your choice(1-7): 7
Enter the vehicle id whose data to be updated: 2
Enter the model : Beze
Enter the Make: Tata
Enter the Year: 2019
Enter the Color: White
Enter the RegistrationNumber: PY9282
Enter the Availability: 1
Enter the DailyRate: 5000
The vehicle datas have been updated successfully.....
```

Before Updating its availability and daily rate:



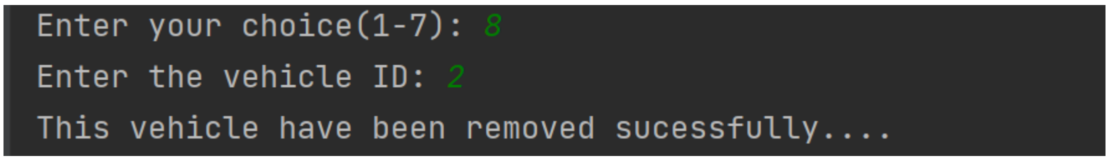
```
mysql> select * from vehicle;
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
| VehicleID | Model          | Make             | Year | Color  | RegistrationNumber | Availability | DailyRate |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
|         1 | Sedan X        | Tata             | 2022 | Silver | TAT123             |            0 |    120.00 |
|         2 | Benze          | Tata             | 2019 | White  | PY9282             |            0 |   1200.00 |
|         3 | Compact Hatch  | Maruti Suzuki    | 2022 | Black  | MAR123             |            1 |     90.00 |
|         4 | Electric Spark | Nexon EV         | 2022 | Blue   | NEX123             |            1 |    100.00 |
|         5 | Hybrid City    | Toyota Kirloskar | 2022 | Green  | TKL123             |            1 |     95.00 |
|         6 | Suzuki         | Shift            | 2009 | Red    | TN4045             |            0 |   1200.00 |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
6 rows in set (0.00 sec)
```

Before Updating its availability and daily rate:



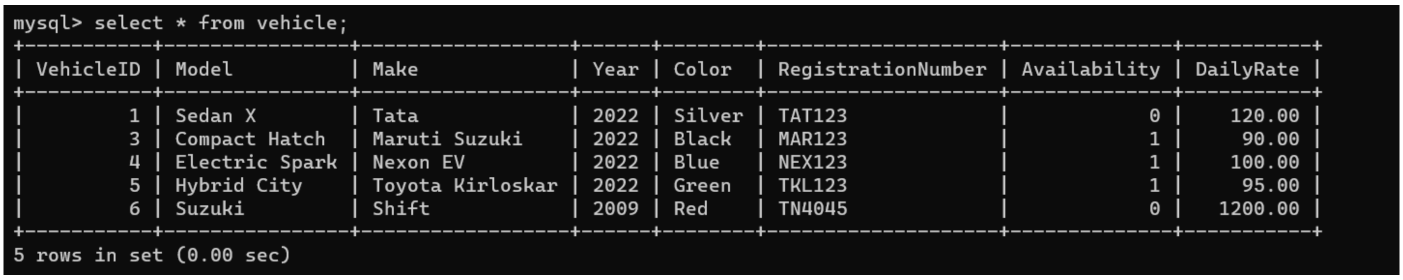
```
mysql> select * from vehicle;
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
| VehicleID | Model          | Make             | Year | Color  | RegistrationNumber | Availability | DailyRate |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
|         1 | Sedan X        | Tata             | 2022 | Silver | TAT123             |            0 |    120.00 |
|         2 | Beze           | Tata             | 2019 | White  | PY9282             |            1 |   5000.00 |
|         3 | Compact Hatch  | Maruti Suzuki    | 2022 | Black  | MAR123             |            1 |     90.00 |
|         4 | Electric Spark | Nexon EV         | 2022 | Blue   | NEX123             |            1 |    100.00 |
|         5 | Hybrid City    | Toyota Kirloskar | 2022 | Green  | TKL123             |            1 |     95.00 |
|         6 | Suzuki         | Shift            | 2009 | Red    | TN4045             |            0 |   1200.00 |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
6 rows in set (0.00 sec)
```

Selecting option 8 to remove a particular vehicle :



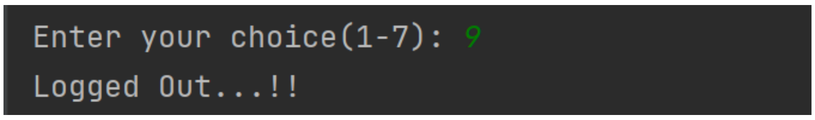
```
Enter your choice(1-7): 8
Enter the vehicle ID: 2
This vehicle have been removed sucessfully....
```

Successfully removed the vehicle with vehicle ID 2:



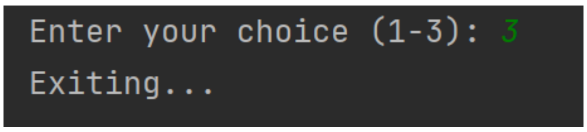
```
mysql> select * from vehicle;
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
| VehicleID | Model          | Make             | Year | Color  | RegistrationNumber | Availability | DailyRate |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
|         1 | Sedan X        | Tata             | 2022 | Silver | TAT123             |            0 |    120.00 |
|         3 | Compact Hatch  | Maruti Suzuki    | 2022 | Black  | MAR123             |            1 |     90.00 |
|         4 | Electric Spark | Nexon EV         | 2022 | Blue   | NEX123             |            1 |    100.00 |
|         5 | Hybrid City    | Toyota Kirloskar | 2022 | Green  | TKL123             |            1 |     95.00 |
|         6 | Suzuki         | Shift            | 2009 | Red    | TN4045             |            0 |   1200.00 |
+-----------+----------------+------------------+------+--------+--------------------+--------------+-----------+
5 rows in set (0.00 sec)
```

Selecting option 9 to log out:



```
Enter your choice(1-7): 9
Logged Out...!!
```

Selecting option 3 from the Car Reservation System Menu to exit the application:



```
Enter your choice (1-3): 3
Exiting...
```

Unit Testing:

Create NUnit test cases for car rental System are essential to ensure the correctness and reliability of your system. Below are some example questions to guide the creation of NUnit test cases for various components of the system:

1. Test customer authentication with invalid credentials.

2. Test updating customer information.

3. Test adding a new vehicle.

4. Test updating vehicle details.

5. Test getting a list of available vehicles.

6. Test getting a list of all vehicles

```python
import unittest
from datetime import date

from dao.service.AuthenticationService import AuthenticationService
from dao.service.CustomerService import CustomerService
from dao.service.DatabaseContext import DatabaseContext
from dao.service.VehicleService import VehicleService
from exception.CustomExceptions import AuthenticationException
from util.DBPropertyUtil import DBPropertyUtil
from util.DBConnUtil import DBConnUtil

class TestCustomerAuthentication(unittest.TestCase):
    def setUp(self):
        self.db_context = DatabaseContext()
        self.db_context.connect()
        self.auth_service = AuthenticationService(self.db_context)
        self.customer_service = CustomerService(self.db_context)
        self.existing_customer_id = 1
        self.existing_vehicle_id = 1
        self.vehicle_service = VehicleService(self.db_context)
    def tearDown(self):
        try:
            if self.db_context:
                self.db_context.connection.close()
        except Exception as e:
            print(f"Error closing database connection: {e}")

    def test_invalid_credentials(self):
        invalid_username = "invalid_username"
        invalid_password = "invalid_password"
        with self.assertRaises(AuthenticationException) as context:
            self.auth_service.authenticate_user(invalid_username,
invalid_password)

        self.assertEqual(str(context.exception), "Incorrect username or
password during customer or admin login.")

    def test_update_customer_information(self):

        updated_info = {
            'customer_id': self.existing_customer_id,
            'first_name': 'Balakumaran',
            'last_name': 'P',
            'email': 'balabkkumaran55@gmail.com',
            'phone_number': '6382474871',
            'address': 'Pondicherry',
            'username': 'mr_bala_bk_45',
            'password': 'prathibala5'
        }
        self.customer_service.update_customer(updated_info)
        updated_customer =
self.customer_service.get_customer_by_id(self.existing_customer_id)
        self.assertEqual(updated_customer[1], 'Balakumaran')
        self.assertEqual(updated_customer[2], 'P')
        self.assertEqual(updated_customer[3], 'balabkkumaran55@gmail.com')
        self.assertEqual(updated_customer[4], '6382474871')
        self.assertEqual(updated_customer[5], 'Pondicherry')
        self.assertEqual(updated_customer[6], 'mr_bala_bk_45')
        self.assertEqual(updated_customer[7], 'prathibala5')

    def test_update_vehicle(self):
        updated_info = {
            'vehicle_id': self.existing_vehicle_id,
            'Model': 'Aadi',
            'Make': 'Tata',
            'Year': '2023',
            'Color': 'Blue',
            'RegistrationNumber': 'PY7282',
            'Availability': 0,
            'DailyRate': '1555.00'
        }
        self.vehicle_service.update_vehicle(updated_info)

        updated_vehicle = 
self.vehicle_service.get_vehicle_by_id(self.existing_vehicle_id)

        self.assertEqual(updated_vehicle[1], 'Aadi')
        self.assertEqual(updated_vehicle[2], 'Tata')
        self.assertEqual(updated_vehicle[3], 2023)
        self.assertEqual(updated_vehicle[4], 'Blue')
        self.assertEqual(updated_vehicle[5], 'PY7282')
        self.assertEqual(updated_vehicle[6], 0)
        self.assertEqual(updated_vehicle[7], 1555.00)

    def test_all_vehicles(self):
        connection = self.db_context.get_connection()
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM Vehicle")
        all_vehicles = cursor.fetchall()

        self.assertIsNotNone(all_vehicles, "The list of all vehicles should not 
be None.")
        self.assertIsInstance(all_vehicles, list, "The result should be a 
list.")
        self.assertGreater(len(all_vehicles), 0, "There should be at least one 
vehicle in the list.")

    def test_available_vehicles(self):
        connection = self.db_context.get_connection()
        cursor = connection.cursor()
        cursor.execute("SELECT * FROM Vehicle WHERE Availability = 1")
        available_vehicles = cursor.fetchall()
        for vehicle in available_vehicles:
            self.assertEqual(vehicle[6], 1, "The 'Availability' should be 1 for 
available vehicles.")


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

1) test_invalid_credentials:

- Attempts to authenticate a user with invalid credentials.
- Expects an AuthenticationException to be raised with a specific error message.

2) test_update_customer_information:

- Updates customer information with new data.
- Retrieves the updated customer information from the database.
- Checks if the retrieved information matches the expected values.

3) test_update_vehicle:

- Updates vehicle information with new data.
- Retrieves the updated vehicle information from the database.
- Checks if the retrieved information matches the expected values.

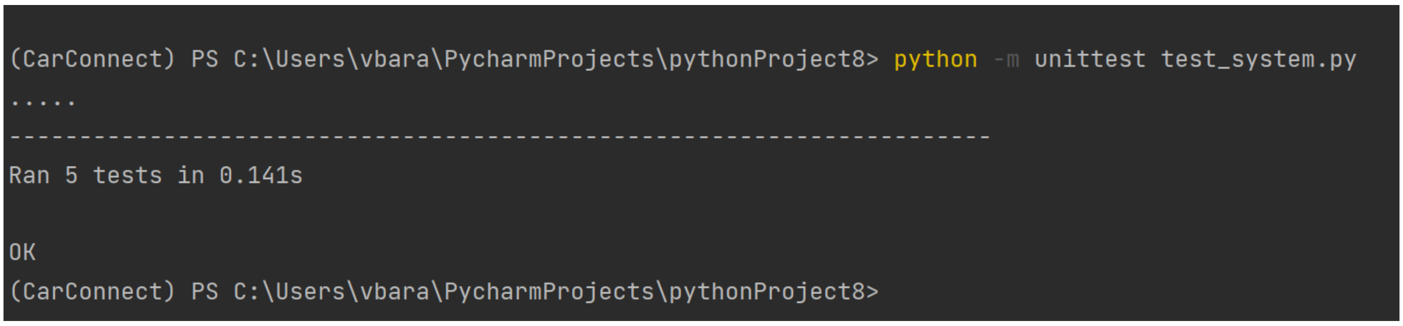
4) test_all_vehicles:

- Executes a SQL query to retrieve all vehicles from the database.
- Checks that the result is not None, is a list, and contains at least one vehicle.

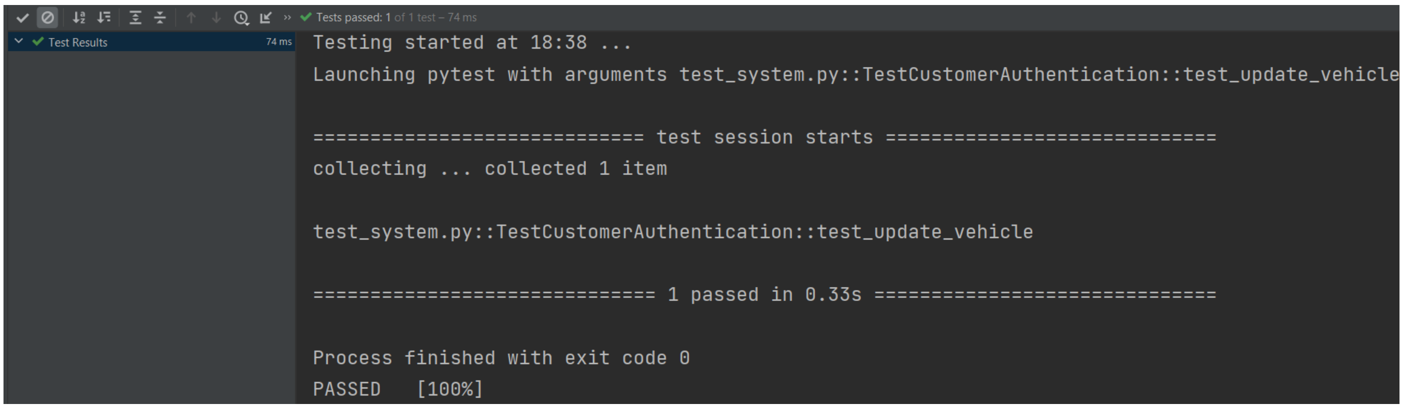
5) test_available_vehicles:

- Executes a SQL query to retrieve available vehicles from the database.
- Checks that the 'Availability' field is set to 1 for all retrieved available vehicles.

Output:



```
(CarConnect) PS C:\Users\vbara\PycharmProjects\pythonProject8> python -m unittest test_system.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.141s

OK
(CarConnect) PS C:\Users\vbara\PycharmProjects\pythonProject8>
```



```
Testing started at 18:38 ...
Launching pytest with arguments test_system.py::TestCustomerAuthentication::test_update_vehicle

============================= test session starts ==============================
collecting ... collected 1 item

test_system.py::TestCustomerAuthentication::test_update_vehicle

============================== 1 passed in 0.33s ===============================

Process finished with exit code 0
PASSED   [100%]
```