**CASE STUDY – CAR CONNECT**

**SQL Schema Creation:**

Create database CarConnectDB;

Use CarConnectDB;

**Creation of Customer Table**

CREATE TABLE Customer (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(50) NOT NULL ,

LastName VARCHAR(50) NOT NULL ,

Email VARCHAR(100),

PhoneNumber VARCHAR(18),

Address VARCHAR(255),

Username VARCHAR(50) NOT NULL UNIQUE,

Password VARCHAR(255)NOT NULL ,

RegistrationDate DATE

);

**Creation of Vehicle Table**

CREATE TABLE Vehicle (

VehicleID INT PRIMARY KEY,

Model VARCHAR(100),

Make VARCHAR(100),

Year INT,

Color VARCHAR(50),

RegistrationNumber VARCHAR(20)NOT NULL UNIQUE,

Availability BOOLEAN,

DailyRate DECIMAL(10, 2)

);

**Creation of Reservation Table**

CREATE TABLE Reservation (

ReservationID INT PRIMARY KEY,

CustomerID INT,

VehicleID INT,

StartDate DATETIME,

EndDate DATETIME,

TotalCost DECIMAL(10, 2),

Status ENUM('pending','confirmed', 'completed'),

FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID) ON DELETE CASCADE,

FOREIGN KEY (VehicleID) REFERENCES Vehicle(VehicleID) ON DELETE CASCADE

);

**Creation of Admin Table**

CREATE TABLE Admin (

AdminID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

PhoneNumber VARCHAR(18),

Username VARCHAR(50) UNIQUE,

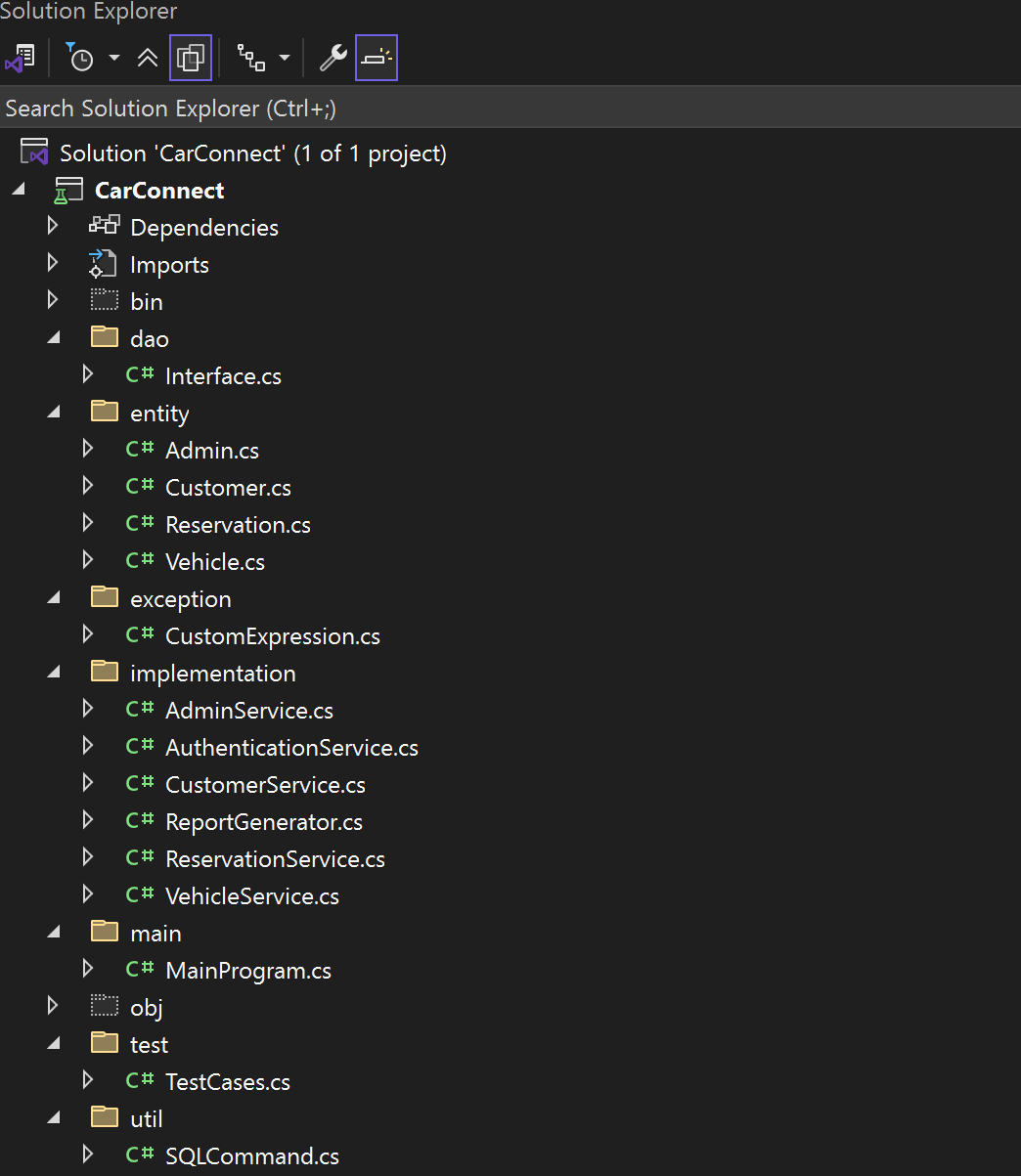
Password VARCHAR(255),

Role ENUM('Super\_admin','fleet manager'),

JoinDate DATE

);

**Layering of the classes from Solution Manager**



**Entity**

1. **Customer.cs**

using System;

namespace entity

{

public class Customer

{

private int CustomerID;

private string FirstName;

private string LastName;

private string Email;

private string PhoneNumber;

private string Address;

private string Username;

private string Password;

private string RegistrationDate;

public Customer()

{

CustomerID = 0;

FirstName = "None";

LastName = "None";

Email = "None";

PhoneNumber = "None";

Address = "None";

Username = "None";

Password = "None";

RegistrationDate = "None";

}

public Customer(int cid, string f, string l, string e, string p, string a, string u, string pass, string rdate)

{

CustomerID = cid;

FirstName = f;

LastName = l;

Email = e;

PhoneNumber = p;

Address = a;

Username = u;

Password = pass;

RegistrationDate = rdate;

}

public int customerID

{

get { return CustomerID; }

set { CustomerID = value; }

}

public string firstName

{

get { return FirstName; }

set { FirstName = value; }

}

public string lastName

{

get { return LastName; }

set { LastName = value; }

}

public string email

{

get { return Email; }

set { Email = value; }

}

public string phoneNumber

{

get { return PhoneNumber; }

set { PhoneNumber = value; }

}

public string address

{

get { return Address; }

set { Address = value; }

}

public string username

{

get { return Username; }

set { Username = value; }

}

public string password

{

get { return Password; }

set { Password = value; }

}

public string registrationDate

{

get { return RegistrationDate; }

set { RegistrationDate = value; }

}

public void Authenticate(string pass)

{

if (password == pass)

{

Console.WriteLine("The user is Authenticated");

}

}

public void PrintInfo()

{

Console.WriteLine("CustomerID:" + customerID);

Console.WriteLine("First Name:" + firstName);

Console.WriteLine("Last Name:" + lastName);

Console.WriteLine("Email:" + email);

Console.WriteLine("Phone Number:" + phoneNumber);

Console.WriteLine("Address:" + address);

Console.WriteLine("Username:" + username);

Console.WriteLine("Password:" + password);

Console.WriteLine("Registration Date:" + registrationDate);

}

}

}

1. **Vehicle.cs**

using System;

namespace entity

{

public class Vehicle

{

private int VehicleID;

private string Model;

private string Make;

private int Year;

private string Color;

private string RegistrationNumber;

private bool Availability;

private double DailyRate;

public Vehicle()

{

VehicleID = 0;

Model = "None";

Make = "None";

Year = 0;

Color = "None";

RegistrationNumber = "None";

Availability = false;

DailyRate = 0.0;

}

public Vehicle(int vid, string mo, string ma, int y, string c, string r, bool a, double ra)

{

VehicleID = vid;

Model = mo;

Make = ma;

Year = y;

Color = c;

RegistrationNumber = r;

Availability = a;

DailyRate = ra;

}

public int vehicleID

{

get { return VehicleID; }

set { VehicleID = value; }

}

public string model

{

get { return Model; }

set { Model = value; }

}

public string make

{

get { return Make; }

set { Make = value; }

}

public int year

{

get { return Year; }

set { Year = value; }

}

public string color

{

get { return Color; }

set { Color = value; }

}

public string registrationNumber

{

get { return RegistrationNumber; }

set { RegistrationNumber = value; }

}

public bool availability

{

get { return Availability; }

set { Availability = value; }

}

public double dailyRate

{

get { return DailyRate; }

set { DailyRate = value; }

}

public void PrintInfo()

{

Console.WriteLine("VehicleID: " + vehicleID);

Console.WriteLine("Model: " + model);

Console.WriteLine("Make: " + make);

Console.WriteLine("Year: " + year);

Console.WriteLine("Color: " + color);

Console.WriteLine("Registration Number: " + registrationNumber);

Console.WriteLine("Availability: " + availability);

Console.WriteLine("Daily Rate: " + dailyRate);

}

}

}

1. **Reservation.cs**

using System;

namespace entity

{

public class Reservation

{

private int ReservationID;

private int CustomerID;

private int VehicleID;

private string StartDate;

private string EndDate;

private double TotalCost;

private string Status;

public Reservation()

{

ReservationID = 0;

CustomerID = 0;

VehicleID = 0;

StartDate = "None";

EndDate = "None";

TotalCost = 0.0;

Status = "None";

}

public Reservation(int rid, int cid, int vid, string s, string end, double c, string stat)

{

ReservationID = rid;

CustomerID = cid;

VehicleID = vid;

StartDate = s;

EndDate = end;

TotalCost = c;

Status = stat;

}

public int reservationID

{

get { return ReservationID; }

set { ReservationID = value; }

}

public int customerID

{

get { return CustomerID; }

set { CustomerID = value; }

}

public int vehicleID

{

get { return VehicleID; }

set { VehicleID = value; }

}

public string startDate

{

get { return StartDate; }

set { StartDate = value; }

}

public string endDate

{

get { return EndDate; }

set { EndDate = value; }

}

public double totalCost

{

get { return TotalCost; }

set { TotalCost = value; }

}

public string status

{

get { return Status; }

set { Status = value; }

}

public void CalculateTotalCost()

{

Console.WriteLine("The total cost for reservation is:" + totalCost);

}

public void PrintInfo()

{

Console.WriteLine("Reservation ID: " + reservationID);

Console.WriteLine("Customer ID: " + customerID);

Console.WriteLine("Vehicle ID: " + vehicleID);

Console.WriteLine("Start Date: " + startDate);

Console.WriteLine("End Date: " + endDate);

Console.WriteLine("Total Cost: " + totalCost);

Console.WriteLine("Status: " + status);

}

}

}

1. **Admin.cs**

using System;

namespace entity

{

public class Admin

{

private int AdminID;

private string FirstName;

private string LastName;

private string Email;

private string PhoneNumber;

private string Username;

private string Password;

private string Role;

private string JoinDate;

public Admin()

{

AdminID = 0;

FirstName = "None";

LastName = "None";

Email = "None";

PhoneNumber = "None";

Username = "None";

Password = "None";

Role = "None";

JoinDate = "None";

}

public Admin(int aid, string f, string l, string e, string p, string u, string pass, string r, string jdate)

{

AdminID = aid;

FirstName = f;

LastName = l;

Email = e;

PhoneNumber = p;

Username = u;

Password = pass;

Role = r;

JoinDate = jdate;

}

public int adminID

{

get { return AdminID; }

set { AdminID = value; }

}

public string firstName

{

get { return FirstName; }

set { FirstName = value; }

}

public string lastName

{

get { return LastName; }

set { LastName = value; }

}

public string email

{

get { return Email; }

set { Email = value; }

}

public string phoneNumber

{

get { return PhoneNumber; }

set { PhoneNumber = value; }

}

public string username

{

get { return Username; }

set { Username = value; }

}

public string password

{

get { return Password; }

set { Password = value; }

}

public string role

{

get { return Role; }

set { Role = value; }

}

public string joinDate

{

get { return JoinDate; }

set { JoinDate = value; }

}

public void Authenticate(string pass)

{

if (Password == pass)

{

Console.WriteLine("Admin is authenticated.");

}

}

public void PrintInfo()

{

Console.WriteLine("Admin ID: " + adminID);

Console.WriteLine("First Name: " + firstName);

Console.WriteLine("Last Name: " + lastName);

Console.WriteLine("Email: " + email);

Console.WriteLine("Phone Number: " + phoneNumber);

Console.WriteLine("Username: " + username);

Console.WriteLine("Role: " + role);

Console.WriteLine("Join Date: " + joinDate);

}

}

}

**Dao**

1. **Interface.cs**

using System;

using entity;

namespace interfaces

{

public interface ICustomerService

{

Customer GetCustomerById(int customerId);

Customer GetCustomerByUsername(string username);

void RegisterCustomer(Customer customerData);

void UpdateCustomer(Customer customerData);

void DeleteCustomer(int customerId);

}

public interface IVehicleService

{

Vehicle GetVehicleById(int vehicleId);

void GetAvailableVehicles();

void AddVehicle(Vehicle vehicleData);

void UpdateVehicle(Vehicle vehicleData);

void RemoveVehicle(int vehicleId);

}

public interface IReservationService

{

void GetReservationById(int reservationId);

void GetReservationsByCustomerId(int customerId);

void CreateReservation(Reservation reservationData);

void UpdateReservation(Reservation reservationData);

void CancelReservation(int reservationId);

}

public interface IAdminService

{

void GetAdminById(int adminId);

void GetAdminByUsername(string username);

void RegisterAdmin(Admin adminData);

void UpdateAdmin(Admin adminData);

void DeleteAdmin(int adminId);

}

}

**Implementation**

1. **CustomerService.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

namespace implementation

{

class CustomerService : ICustomerService

{

public List<Customer> customers;

public CustomerService(List<Customer> customerList)

{

customers = customerList;

}

public Customer GetCustomerById(int custId)

{

bool found = false;

foreach (Customer cust in customers)

{

if (cust.customerID == custId)

{

cust.PrintInfo();

found = true;

return cust;

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

return null;

}

public Customer GetCustomerByUsername(string uname)

{

bool found = false;

foreach (Customer cust in customers)

{

if (cust.username == uname)

{

cust.PrintInfo();

found = true;

return cust;

break;

}

}

if (!found)

{

Console.WriteLine("username does not exist");

return null;

}

return null;

}

public void RegisterCustomer(Customer cust)

{

foreach (Customer ecust in customers)

{

if (ecust.customerID == cust.customerID || ecust.username == cust.username)

{

Console.WriteLine("Customer with this ID or Username already exists.");

return;

}

}

customers.Add(cust);

Console.WriteLine("Customer registered.");

}

public void UpdateCustomer(Customer cust)

{

bool found = false;

for (int i = 0; i < customers.Count; i++)

{

if (customers[i].customerID == cust.customerID)

{

customers[i] = cust;

found = true;

Console.WriteLine("Customer Info updated.");

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

}

public void DeleteCustomer(int custId)

{

bool found = false;

for (int i = 0; i < customers.Count; i++)

{

if (customers[i].customerID == custId)

{

customers.RemoveAt(i);

found = true;

Console.WriteLine("Customer deleted");

break;

}

}

if (!found)

{

Console.WriteLine("Customer does not exist");

}

}

}

}

1. **VehicleService.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

using CustomExceptions;

namespace implementation

{

class VehicleService : IVehicleService

{

private List<Vehicle> vehicles;

public VehicleService(List<Vehicle> vehicleList)

{

vehicles = vehicleList;

}

public Vehicle GetVehicleById(int vehicleId)

{

bool found = false;

foreach (Vehicle vehicle in vehicles)

{

if (vehicle.vehicleID == vehicleId)

{

vehicle.PrintInfo();

found = true;

return vehicle;

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

return null;

}

public void GetAvailableVehicles()

{

bool found = false;

foreach (Vehicle vehicle in vehicles)

{

if (vehicle.availability)

{

vehicle.PrintInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("No available vehicles.");

}

}

public void AddVehicle(Vehicle vehicleData)

{

foreach (Vehicle eVehicle in vehicles)

{

if (eVehicle.vehicleID == vehicleData.vehicleID || eVehicle.registrationNumber == vehicleData.registrationNumber)

{

Console.WriteLine("Vehicle already exists.");

return;

}

}

vehicles.Add(vehicleData);

Console.WriteLine("Vehicle added.");

}

public void UpdateVehicle(Vehicle vehicleData)

{

bool found = false;

for (int i = 0; i < vehicles.Count; i++)

{

if (vehicles[i].vehicleID == vehicleData.vehicleID)

{

vehicles[i] = vehicleData;

found = true;

Console.WriteLine("Vehicle Info updated.");

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

}

public void RemoveVehicle(int vehicleId)

{

bool found = false;

for (int i = 0; i < vehicles.Count; i++)

{

if (vehicles[i].vehicleID == vehicleId)

{

vehicles.RemoveAt(i);

found = true;

Console.WriteLine("Vehicle removed successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Vehicle does not exist.");

}

}

}

}

1. **ReservationService.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

namespace implementation

{

class ReservationService : IReservationService

{

private List<Reservation> reservations;

public ReservationService(List<Reservation> reservationList)

{

reservations = reservationList;

}

public void GetReservationById(int reservationId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.reservationID == reservationId)

{

r.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

public void GetReservationsByCustomerId(int customerId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.customerID == customerId)

{

r.PrintInfo();

found = true;

}

}

if (!found)

{

Console.WriteLine("No reservations found for the customer.");

}

}

public void CreateReservation(Reservation reservationData)

{

foreach (Reservation er in reservations)

{

if (er.reservationID == reservationData.reservationID)

{

Console.WriteLine("Reservation with this ID already exists.");

return;

}

}

reservations.Add(reservationData);

Console.WriteLine("Reservation created successfully.");

}

public void UpdateReservation(Reservation reservationData)

{

bool found = false;

for (int i = 0; i < reservations.Count; i++)

{

if (reservations[i].reservationID == reservationData.reservationID)

{

reservations[i] = reservationData;

found = true;

Console.WriteLine("Reservation updated successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

public void CancelReservation(int reservationId)

{

bool found = false;

for (int i = 0; i < reservations.Count; i++)

{

if (reservations[i].reservationID == reservationId)

{

reservations.RemoveAt(i);

found = true;

Console.WriteLine("Reservation cancelled successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Reservation does not exist.");

}

}

}

}

1. **AdminService.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

namespace implementation

{

class AdminService : IAdminService

{

private List<Admin> admins;

public AdminService(List<Admin> adminList)

{

admins = adminList;

}

public void GetAdminById(int adminId)

{

bool found = false;

foreach (Admin a in admins)

{

if (a.adminID == adminId)

{

a.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Admin does not exist.");

}

}

public void GetAdminByUsername(string username)

{

bool found = false;

foreach (Admin a in admins)

{

if (a.username == username)

{

a.PrintInfo();

found = true;

break;

}

}

if (!found)

{

Console.WriteLine("Admin does not exist.");

}

}

public void RegisterAdmin(Admin adminData)

{

foreach (Admin ea in admins)

{

if (ea.username == adminData.username)

{

Console.WriteLine("Username exists.");

return;

}

}

admins.Add(adminData);

Console.WriteLine("Admin registered.");

}

public void UpdateAdmin(Admin adminData)

{

bool found = false;

for (int i = 0; i < admins.Count; i++)

{

if (admins[i].adminID == adminData.adminID)

{

admins[i] = adminData;

found = true;

Console.WriteLine("Admin information updated successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Admin with this ID does not exist.");

}

}

public void DeleteAdmin(int adminId)

{

bool found = false;

for (int i = 0; i < admins.Count; i++)

{

if (admins[i].adminID == adminId)

{

admins.RemoveAt(i);

found = true;

Console.WriteLine("Admin deleted successfully.");

break;

}

}

if (!found)

{

Console.WriteLine("Admin with this ID does not exist.");

}

}

}

}

1. **AuthenticationService.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

using CustomExceptions;

namespace implementation

{

public class AuthenticationService

{

private List<Customer> customers;

private List<Admin> admins;

public AuthenticationService(List<Customer> customers, List<Admin> admins)

{

this.customers = customers;

this.admins = admins;

}

public void AuthenticateCustomer(string uname, string pass)

{

foreach (Customer customer in customers)

{

if (customer.username == uname && customer.password == pass)

{

Console.WriteLine("Customer is Autheticated");

}

}

throw new AuthenticationException("Authentication Failure");

}

public void AuthenticateAdmin(string uname, string pass)

{

foreach (Admin admin in admins)

{

if (admin.username == uname && admin.password == pass)

{

Console.WriteLine("Admin is authenticated");

}

}

throw new AuthenticationException("Admin Authentication Failure");

}

}

}

1. **ReportGenerator.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

using CustomExceptions;

namespace implementation

{

public class ReportGenerator

{

private List<Vehicle> vehicles;

private List<Reservation> reservations;

public ReportGenerator(List<Vehicle> vehicles, List<Reservation> reservations)

{

this.vehicles = vehicles;

this.reservations = reservations;

}

public void VehicleInfo(int id)

{

foreach (Vehicle v in vehicles)

{

if (v.vehicleID == id)

{

v.PrintInfo();

return;

}

}

throw new VehicleNotFoundException("Vehicle Not Found");

}

public void ReservationInfo(int customerId)

{

bool found = false;

foreach (Reservation r in reservations)

{

if (r.customerID == customerId)

{

found = true;

r.PrintInfo();

}

}

if (!found)

{

throw new ReservationException("No Reservations Found");

}

}

}

}

1. **CustomException.cs**

using System;

namespace CustomExceptions

{

public class AuthenticationException : Exception

{

public AuthenticationException(string msg) : base(msg) { }

}

public class ReservationException : Exception

{

public ReservationException(string msg) : base(msg) { }

}

public class VehicleNotFoundException : Exception

{

public VehicleNotFoundException(string msg) : base(msg) { }

}

public class AdminNotFoundException : Exception

{

public AdminNotFoundException(string msg) : base(msg) { }

}

public class InvalidInputException : Exception

{

public InvalidInputException(string msg) : base(msg) { }

}

public class DatabaseConnectionException : Exception

{

public DatabaseConnectionException(string msg) : base(msg) { }

}

public class InvalidBooleanInputException : Exception

{

public InvalidBooleanInputException() : base("The input is not a valid boolean value.")

{

}

public InvalidBooleanInputException(string message) : base(message)

{

}

public InvalidBooleanInputException(string message, Exception inner) : base(message, inner)

{

}

}

}

1. **SQLCommand.cs**

using System;

using System.Data;

using System.Data.SqlClient;

namespace CarRentalSystem

{

public class CarRentalService

{

private SqlConnection con = null;

private SqlCommand cmd;

public SqlConnection GetConnection()

{

con = new SqlConnection("data source=AMSPC12\\SQLEXPRESS; initial catalog=CarConnectDB; Integrated security=true;");

con.Open();

return con;

}

public void InsertCustomerData(int id1, string f, string l, string e, string p, string a, string u, string pass)

{

con = GetConnection();

DateTime rd = DateTime.Now;

cmd = new SqlCommand("INSERT INTO Customer VALUES (@customerID, @firstName, @lastName, @email, @phoneNumber, @address, @username, @password, @registrationDate)", con);

cmd.Parameters.AddWithValue("@CustomerID", id1);

cmd.Parameters.AddWithValue("@FirstName", f);

cmd.Parameters.AddWithValue("@LastName", l);

cmd.Parameters.AddWithValue("@Email", e);

cmd.Parameters.AddWithValue("@PhoneNumber", p);

cmd.Parameters.AddWithValue("@Address", a);

cmd.Parameters.AddWithValue("@Username", u);

cmd.Parameters.AddWithValue("@Password", pass);

cmd.Parameters.AddWithValue("@RegistrationDate", rd);

ExecuteNonQuery("Customer");

}

public void InsertVehicleData(int id2, string mo, string ma, int y, string c, string rn, bool a)

{

con = GetConnection();

cmd = new SqlCommand("INSERT INTO Vehicle VALUES (@vehicleID, @model, @make, @year, @color, @registrationNumber, @availability)", con);

cmd.Parameters.AddWithValue("@VehicleID", id2);

cmd.Parameters.AddWithValue("@Model", mo);

cmd.Parameters.AddWithValue("@Make", ma);

cmd.Parameters.AddWithValue("@Year", y);

cmd.Parameters.AddWithValue("@Color", c);

cmd.Parameters.AddWithValue("@RegistrationNumber", rn);

cmd.Parameters.AddWithValue("@Availability", a);

ExecuteNonQuery("Vehicle");

}

public void InsertReservationData(int rid, int cid, int vid, DateTime sd, DateTime ed, decimal tc, string stat)

{

con = GetConnection();

cmd = new SqlCommand("INSERT INTO Reservation VALUES (@reservationID, @customerID, @vehicleID, @startDate, @endDate, @totalCost, @status)", con);

cmd.Parameters.AddWithValue("@ReservationID", rid);

cmd.Parameters.AddWithValue("@CustomerID", cid);

cmd.Parameters.AddWithValue("@VehicleID", vid);

cmd.Parameters.AddWithValue("@StartDate", sd);

cmd.Parameters.AddWithValue("@EndDate", ed);

cmd.Parameters.AddWithValue("@TotalCost", tc);

cmd.Parameters.AddWithValue("@Status", stat);

ExecuteNonQuery("Reservation");

}

public void InsertAdminData(int aid, string fn, string ln, string em, string pn, string un, string pas, string rol)

{

con = GetConnection();

DateTime jd = DateTime.Now;

cmd = new SqlCommand("INSERT INTO Admin VALUES (@adminID, @firstName, @lastName, @email, @phoneNumber, @username, @password, @role, @joinDate)", con);

cmd.Parameters.AddWithValue("@AdminID", aid);

cmd.Parameters.AddWithValue("@FirstName", fn);

cmd.Parameters.AddWithValue("@LastName", ln);

cmd.Parameters.AddWithValue("@Email", em);

cmd.Parameters.AddWithValue("@PhoneNumber", pn);

cmd.Parameters.AddWithValue("@Username", un);

cmd.Parameters.AddWithValue("@Password", pas);

cmd.Parameters.AddWithValue("@Role", rol);

cmd.Parameters.AddWithValue("@JoinDate", jd);

ExecuteNonQuery("Admin");

}

private void ExecuteNonQuery(string tableName)

{

try

{

int rows = cmd.ExecuteNonQuery();

if (rows > 0)

{

Console.WriteLine("Record added successfully.");

}

else

{

Console.WriteLine("Unable to add the record");

}

}

catch (Exception ex)

{

Console.WriteLine("Error inserting data: {ex.Message}");

}

finally

{

con.Close();

}

}

}

}

1. **Main Program.cs**

using System;

using System.Collections.Generic;

using interfaces;

using entity;

using CustomExceptions;

using implementation;

using CarRentalSystem;

using System.Data.SqlClient;

using System;

using System.Collections.Generic;

namespace CarRentalSystem

{

class Program

{

static void Main(string[] args)

{

List<Customer> customerList = new List<Customer>();

List<Vehicle> vehicleList = new List<Vehicle>();

List<Admin> adminList = new List<Admin>();

List<Reservation> reservationList = new List<Reservation>();

CarRentalService dbService = new CarRentalService();

CustomerService customerService = new CustomerService(customerList);

VehicleService vehicleService = new VehicleService(vehicleList);

AdminService adminService = new AdminService(adminList);

ReservationService reservationService = new ReservationService(reservationList);

Customer customer = new Customer();

bool running = true;

while (running)

{

Console.WriteLine("\n--- CAR RENTAL SYSTEM ---");

Console.WriteLine("1. Customer");

Console.WriteLine("2. Vehicle");

Console.WriteLine("3. Reservation");

Console.WriteLine("4. Admin");

Console.WriteLine("5. Exit");

Console.Write("Select an option: ");

string choice = Console.ReadLine();

switch (choice)

{

case "1":

HandleCustomerMenu(customerService, dbService);

break;

case "2":

HandleVehicleMenu(vehicleService, dbService);

break;

case "3":

HandleReservationMenu(reservationService, dbService);

break;

case "4":

HandleAdminMenu(adminService, dbService);

break;

case "5":

running = false;

Console.WriteLine("Exiting...");

break;

default:

Console.WriteLine("Invalid input.");

break;

}

}

}

static void HandleCustomerMenu(CustomerService customerService, CarRentalService db)

{

Console.WriteLine("\n--- Customer Menu ---");

Console.WriteLine("1. Register Customer");

Console.WriteLine("2. Get Customer by ID");

Console.WriteLine("3. Get Customer by Username");

Console.Write("Choice: ");

string input = Console.ReadLine();

if (input == "1")

{

Console.WriteLine("ID: ");

int id1 = Convert.ToInt32(Console.ReadLine());

Console.Write("First Name: ");

string fn = Console.ReadLine();

Console.Write("Last Name: ");

string ln = Console.ReadLine();

Console.Write("Email: ");

string email = Console.ReadLine();

Console.Write("Phone: ");

string phone = Console.ReadLine();

Console.Write("Address: ");

string addr = Console.ReadLine();

Console.Write("Username: ");

string uname = Console.ReadLine();

Console.Write("Password: ");

string pass = Console.ReadLine();

Customer c = new Customer

{

customerID = id1,

firstName = fn,

lastName = ln,

email = email,

phoneNumber = phone,

address = addr,

username = uname,

password = pass

};

customerService.RegisterCustomer(c);

db.InsertCustomerData(id1,fn, ln, email, phone, addr, uname, pass);

}

else if (input == "2")

{

Console.Write("Enter Customer ID: ");

int id = int.Parse(Console.ReadLine());

customerService.GetCustomerById(id);

}

}

static void HandleVehicleMenu(VehicleService vehicleService, CarRentalService db)

{

Console.WriteLine("\n--- Vehicle Menu ---");

Console.WriteLine("1. Add Vehicle");

Console.WriteLine("2. Get Vehicle by ID");

Console.WriteLine("3. Get Available Vehicles");

Console.Write("Choice: ");

string input = Console.ReadLine();

if (input == "1")

{

Console.WriteLine("ID: ");

int id2 = Convert.ToInt32(Console.ReadLine());

Console.Write("Model: ");

string model = Console.ReadLine();

Console.Write("Make: ");

string make = Console.ReadLine();

Console.Write("Year: ");

int year = int.Parse(Console.ReadLine());

Console.Write("Color: ");

string color = Console.ReadLine();

Console.Write("Registration Number: ");

string reg = Console.ReadLine();

Console.Write("Available (true/false): ");

bool avail = bool.Parse(Console.ReadLine());

Vehicle v = new Vehicle

{

vehicleID = id2,

model = model,

make = make,

year = year,

color = color,

registrationNumber = reg,

availability = avail

};

vehicleService.AddVehicle(v);

db.InsertVehicleData(id2, model, make, year, color, reg, avail);

}

else if (input == "2")

{

Console.Write("Enter Vehicle ID: ");

int id = int.Parse(Console.ReadLine());

vehicleService.GetVehicleById(id);

}

else if (input == "3")

{

vehicleService.GetAvailableVehicles();

}

}

static void HandleReservationMenu(ReservationService reservationService, CarRentalService db)

{

Console.WriteLine("\n--- Reservation Menu ---");

Console.WriteLine("1. Make Reservation");

Console.WriteLine("2. Get Reservation by ID");

Console.Write("Choice: ");

string input2 = Console.ReadLine();

if (input2 == "1")

{

Console.Write("Reservation ID: ");

int rid = int.Parse(Console.ReadLine());

Console.Write("Customer ID: ");

int cid = int.Parse(Console.ReadLine());

Console.Write("Vehicle ID: ");

int vid = int.Parse(Console.ReadLine());

Console.Write("Start Date (yyyy-mm-dd): ");

DateTime sd = DateTime.Parse(Console.ReadLine());

Console.Write("End Date (yyyy-mm-dd): ");

DateTime ed = DateTime.Parse(Console.ReadLine());

Console.Write("Total Cost: ");

decimal tc = decimal.Parse(Console.ReadLine());

Console.Write("Status: ");

string status = Console.ReadLine();

db.InsertReservationData(rid, cid, vid, sd, ed, tc, status);

}

else if (input2 == "2")

{

Console.Write("Enter Reservation ID: ");

int id3 = int.Parse(Console.ReadLine());

reservationService.GetReservationById(id3);

}

}

static void HandleAdminMenu(AdminService adminService, CarRentalService db)

{

Console.WriteLine("\n--- Admin Menu ---");

Console.WriteLine("1. Register Admin");

Console.WriteLine("2. Get Admin by ID");

Console.WriteLine("3. Get Admin by Username");

Console.Write("Choice: ");

string input = Console.ReadLine();

if (input == "1")

{

Console.Write("Admin ID: ");

int aid = int.Parse(Console.ReadLine());

Console.Write("First Name: ");

string fn = Console.ReadLine();

Console.Write("Last Name: ");

string ln = Console.ReadLine();

Console.Write("Email: ");

string email = Console.ReadLine();

Console.Write("Phone: ");

string phone = Console.ReadLine();

Console.Write("Username: ");

string uname = Console.ReadLine();

Console.Write("Password: ");

string pass = Console.ReadLine();

Console.Write("Role: ");

string role = Console.ReadLine();

Admin admin = new Admin

{

adminID = aid,

firstName = fn,

lastName = ln,

email = email,

phoneNumber = phone,

username = uname,

password = pass,

role = role

};

adminService.RegisterAdmin(admin);

db.InsertAdminData(aid, fn, ln, email, phone, uname, pass, role);

}

else if (input == "2")

{

Console.Write("Enter Admin ID: ");

int id = int.Parse(Console.ReadLine());

adminService.GetAdminById(id);

}

else if (input == "3")

{

Console.Write("Enter Username: ");

string uname = Console.ReadLine();

adminService.GetAdminByUsername(uname);

}

}

}

}

1. **TestCases.cs**

using NUnit.Framework;

using NUnit.Framework.Legacy;

using System;

using System.Collections.Generic;

using interfaces;

using entity;

using CustomExceptions;

using implementation;

namespace CarConnect.test

{

[TestFixture]

public class CarRentalSystemUnitTests

{

private AuthenticationService authService;

private ICustomerService customerService;

private IVehicleService vehicleService;

private List<Customer> customers;

private List<Vehicle> vehicles;

[SetUp]

public void Setup()

{

customers = new List<Customer>

{

new Customer

{

customerID = 1,

firstName = "Karan",

lastName = "Das",

username = "karandass23",

password = "xyzabc",

email = "karandas23@gmail.com",

phoneNumber = "9876543210",

address = "Chennai",

registrationDate = "2025-04-31"

}

};

vehicles = new List<Vehicle>

{

new Vehicle

{

vehicleID = 1,

make = "Land Cruiser",

model = "Prado",

year = 2020,

color = "White",

registrationNumber = "TN25RN7389",

availability = true,

dailyRate = 75.00

}

};

authService = new AuthenticationService(customers, new List<Admin>());

customerService = new CustomerService(customers);

vehicleService = new VehicleService(vehicles);

}

[Test]

public void TestCustomerAuthentication()

{

try

{

authService.AuthenticateCustomer("karandas23", "xyzabc");

ClassicAssert.Fail("Test Fail");

}

catch (AuthenticationException ex)

{

ClassicAssert.AreEqual("Authentication failed", ex.Message);

}

}

[Test]

public void TestUpdateCustomerInformation()

{

Customer customer = customerService.GetCustomerById(1);

customer.email = "kiaramilan@gmail.com";

customerService.UpdateCustomer(customer);

Customer uc = customerService.GetCustomerById(1);

ClassicAssert.AreEqual("kiaramilan@gmail.com", uc.email);

}

[Test]

public void TestAddNewVehicle()

{

Vehicle newVehicle = new Vehicle

{

vehicleID = 2,

make = "Hyundai",

model = "Elantra",

year = 2025,

color = "White",

registrationNumber = "TN26YZ6789",

availability = true,

dailyRate = 80.00

};

vehicleService.AddVehicle(newVehicle);

Vehicle v = vehicleService.GetVehicleById(2);

ClassicAssert.AreEqual("Hyundai", v.make);

}

[Test]

public void TestUpdateVehicleDetails()

{

Vehicle vehicle = vehicleService.GetVehicleById(1);

vehicle.dailyRate = 90.00;

vehicleService.UpdateVehicle(vehicle);

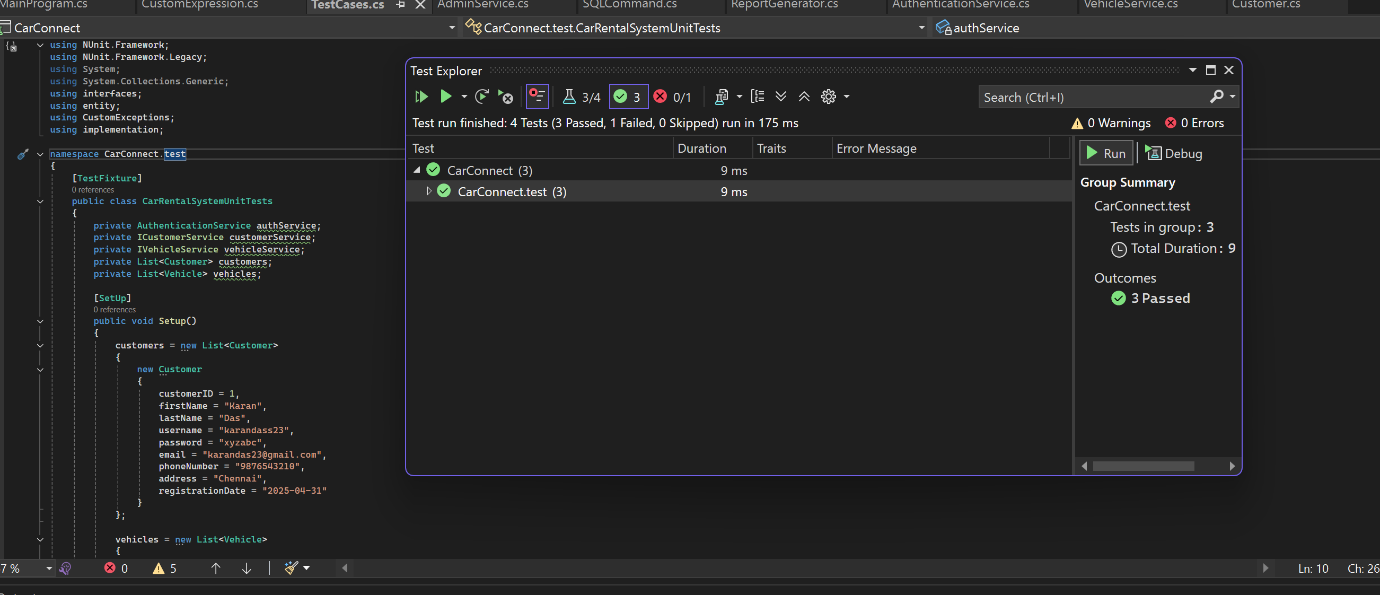
Vehicle uv = vehicleService.GetVehicleById(1);

ClassicAssert.AreEqual(90.00M, uv.dailyRate);

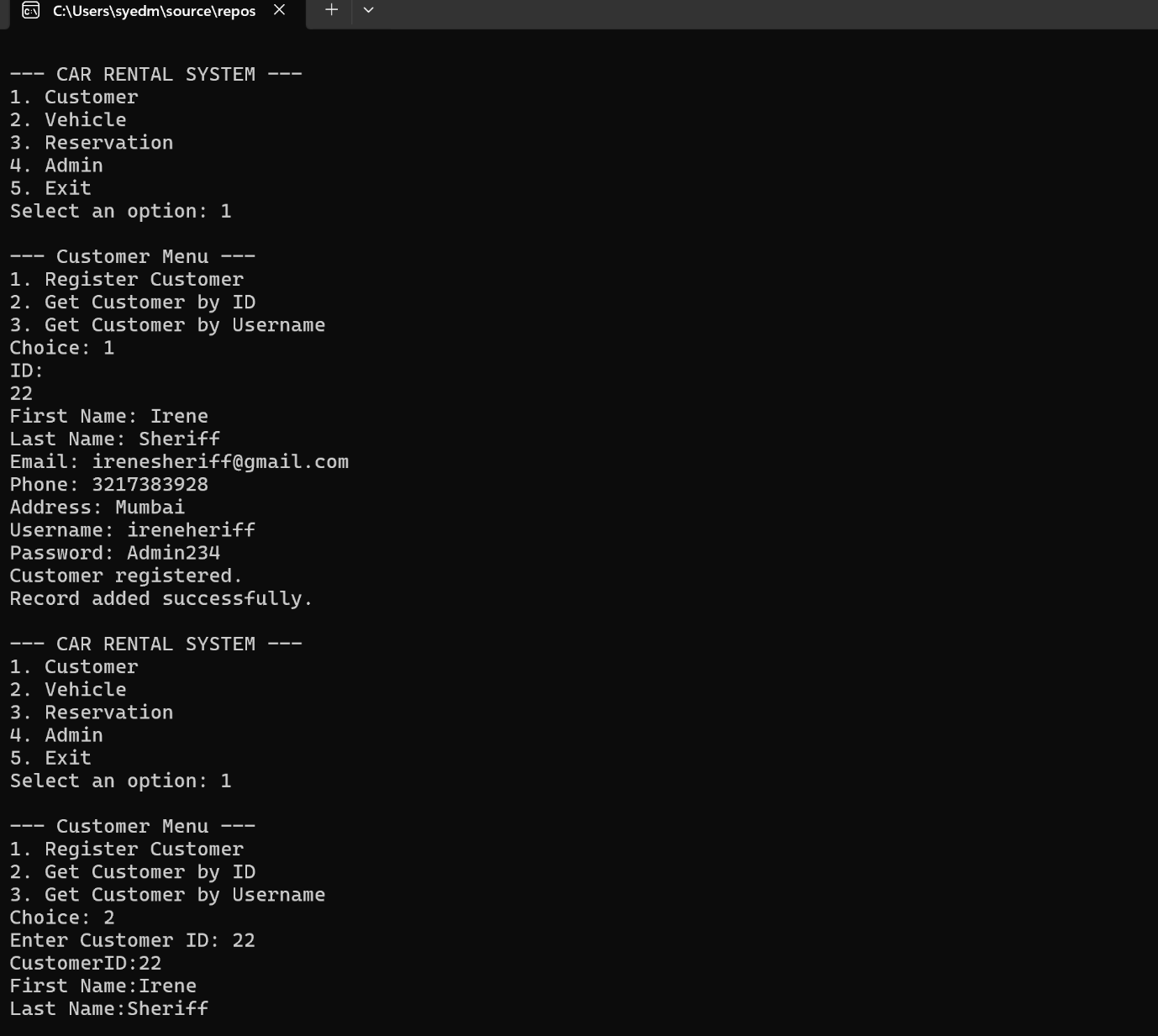
}

}

}



The output screenshots are attached below:



A screenshot of a computer

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

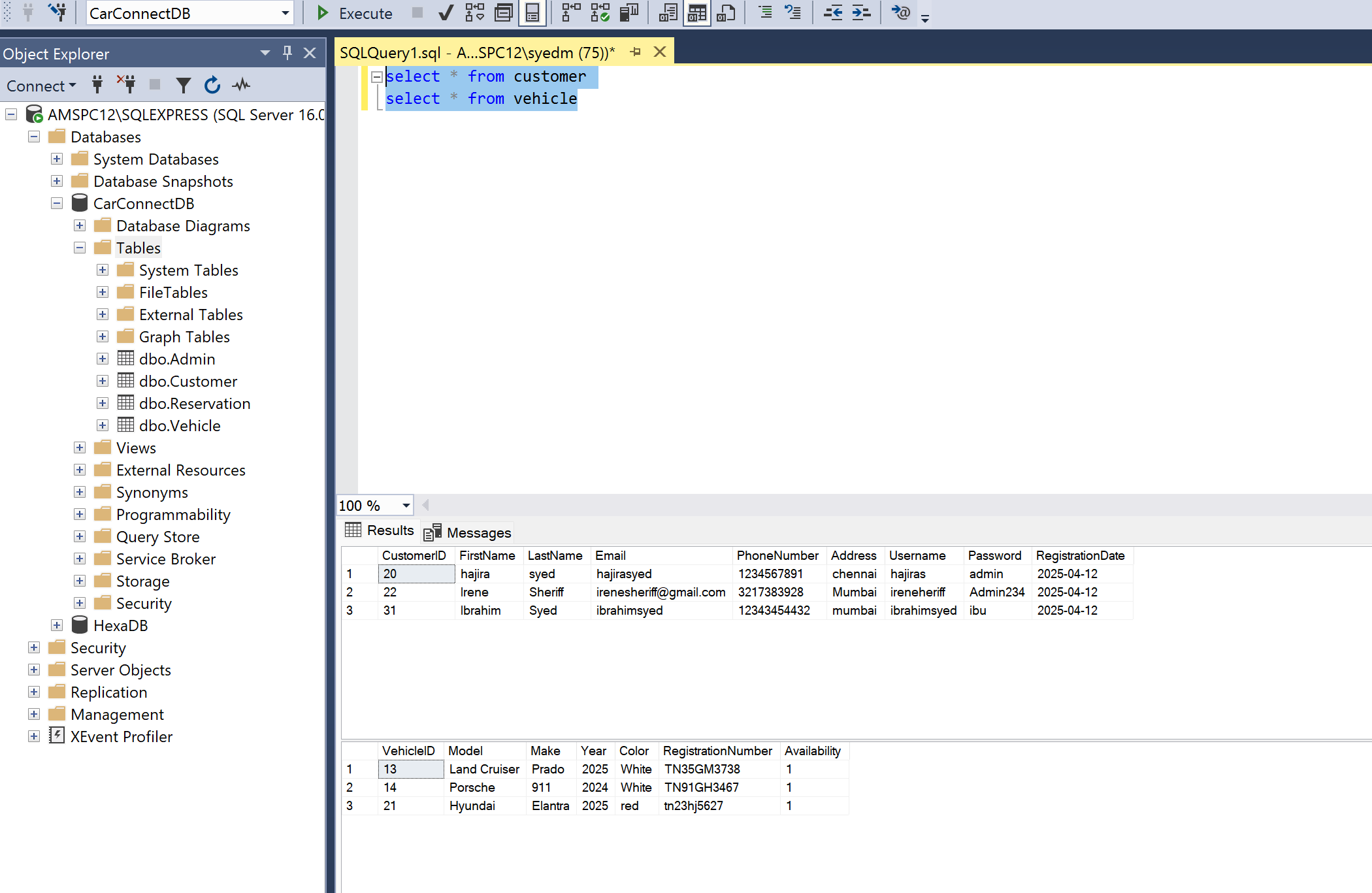
The entered data has been reflected in the Database in the customer Table.

A screenshot of a computer program

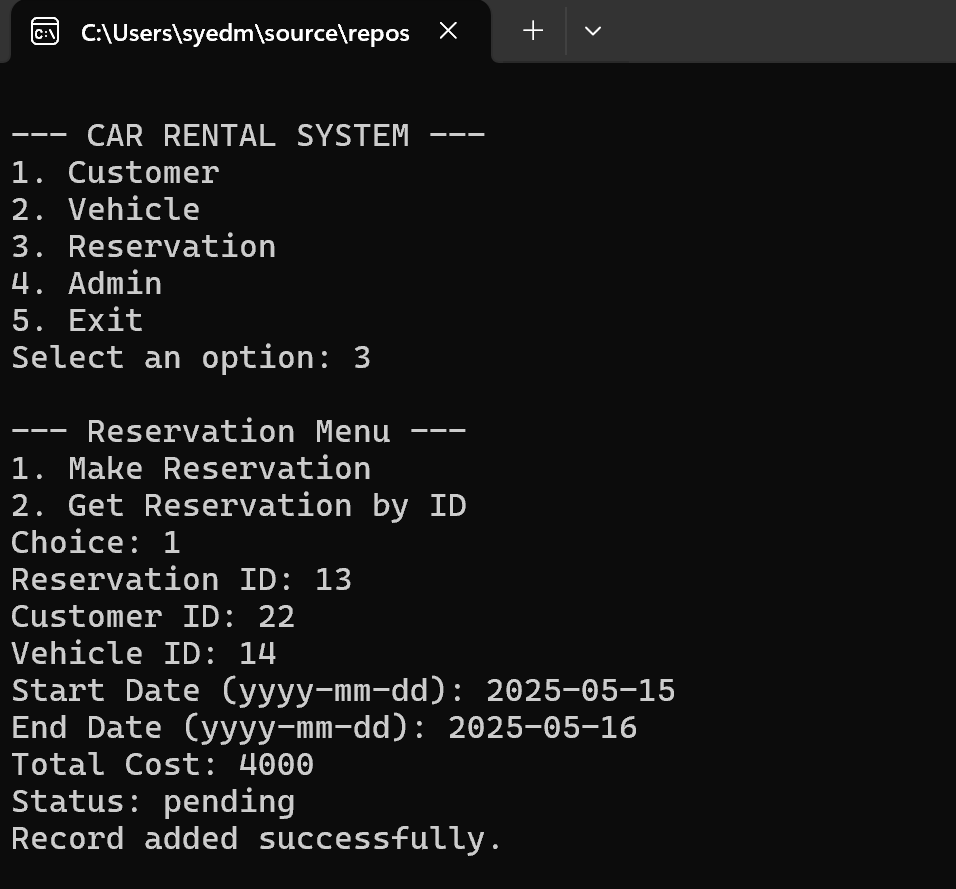
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



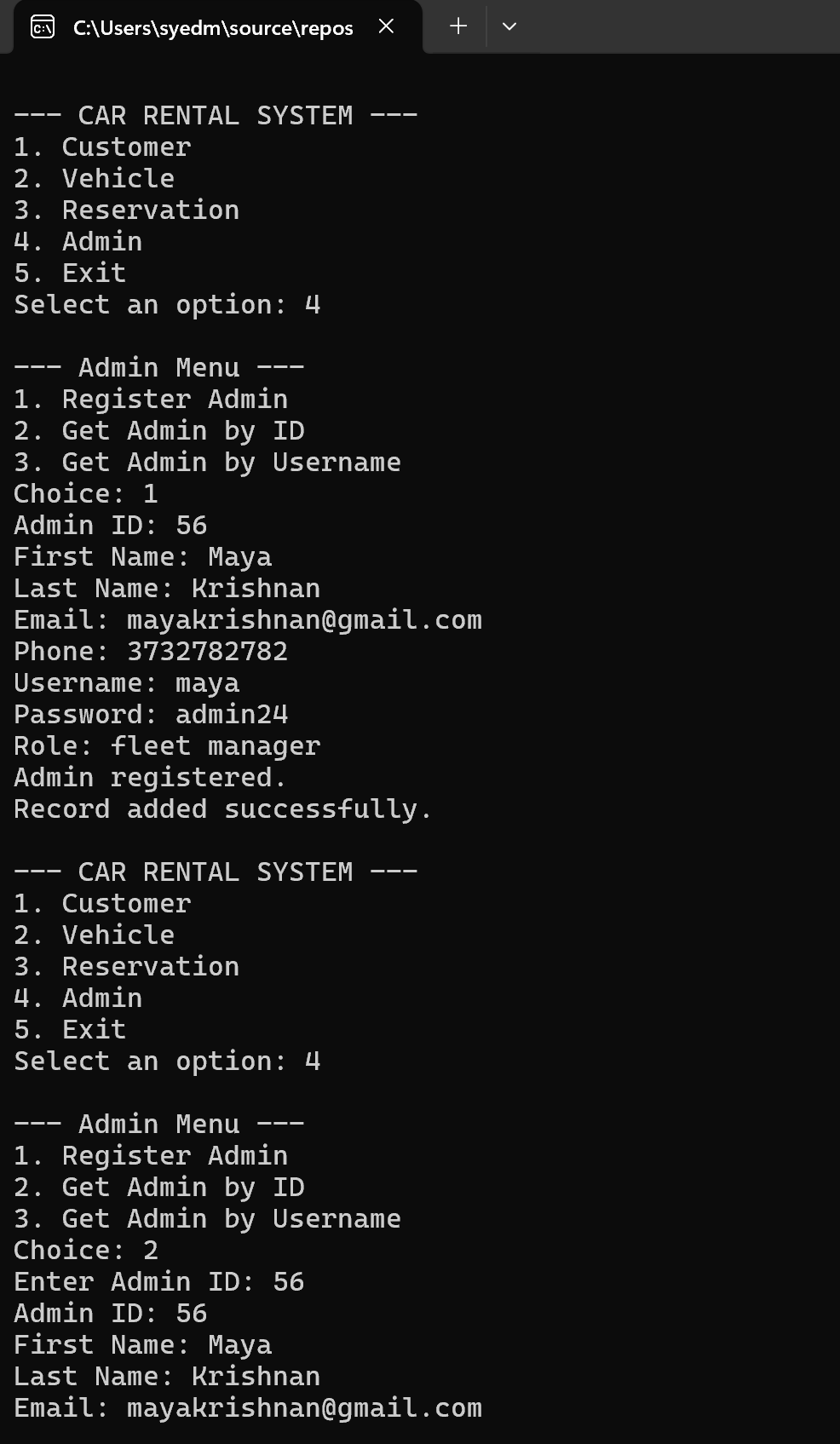
The entered data has been reflected in the Vehicle Table in the Database.

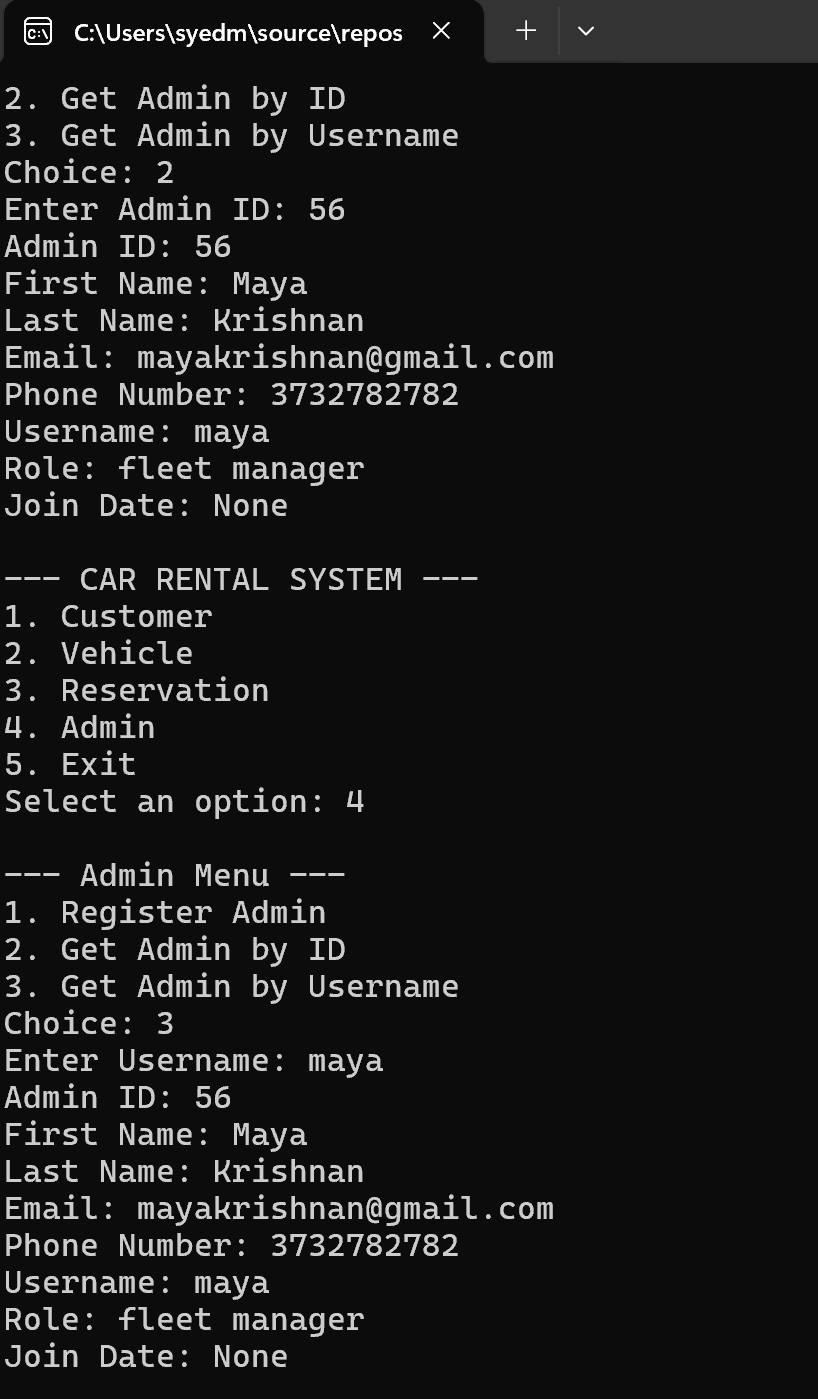


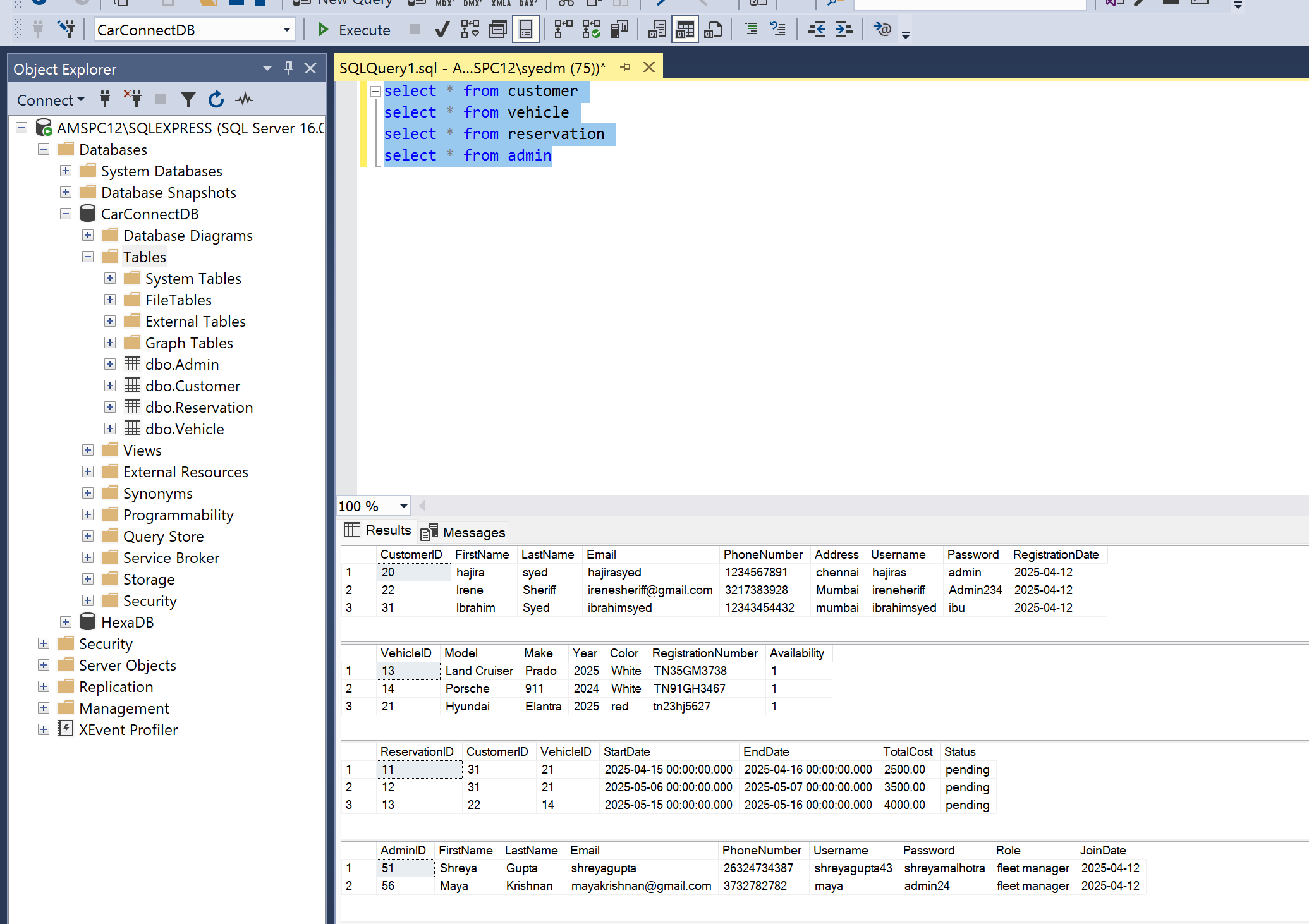
A screenshot of a computer

AI-generated content may be incorrect.

The data has been reflected in the Reservation table in the Database







The entered values are successfully reflected in the Admin table on the database.