LONDON METROPOLITAN UNIVERSITY

CS6004ES –Application Development

Individual Coursework – (2023/24)

ABC Car Traders Order Management System

Name: Senomi Ruhansa Jayasinghe

ID Number: 23059286

Eno: E124475

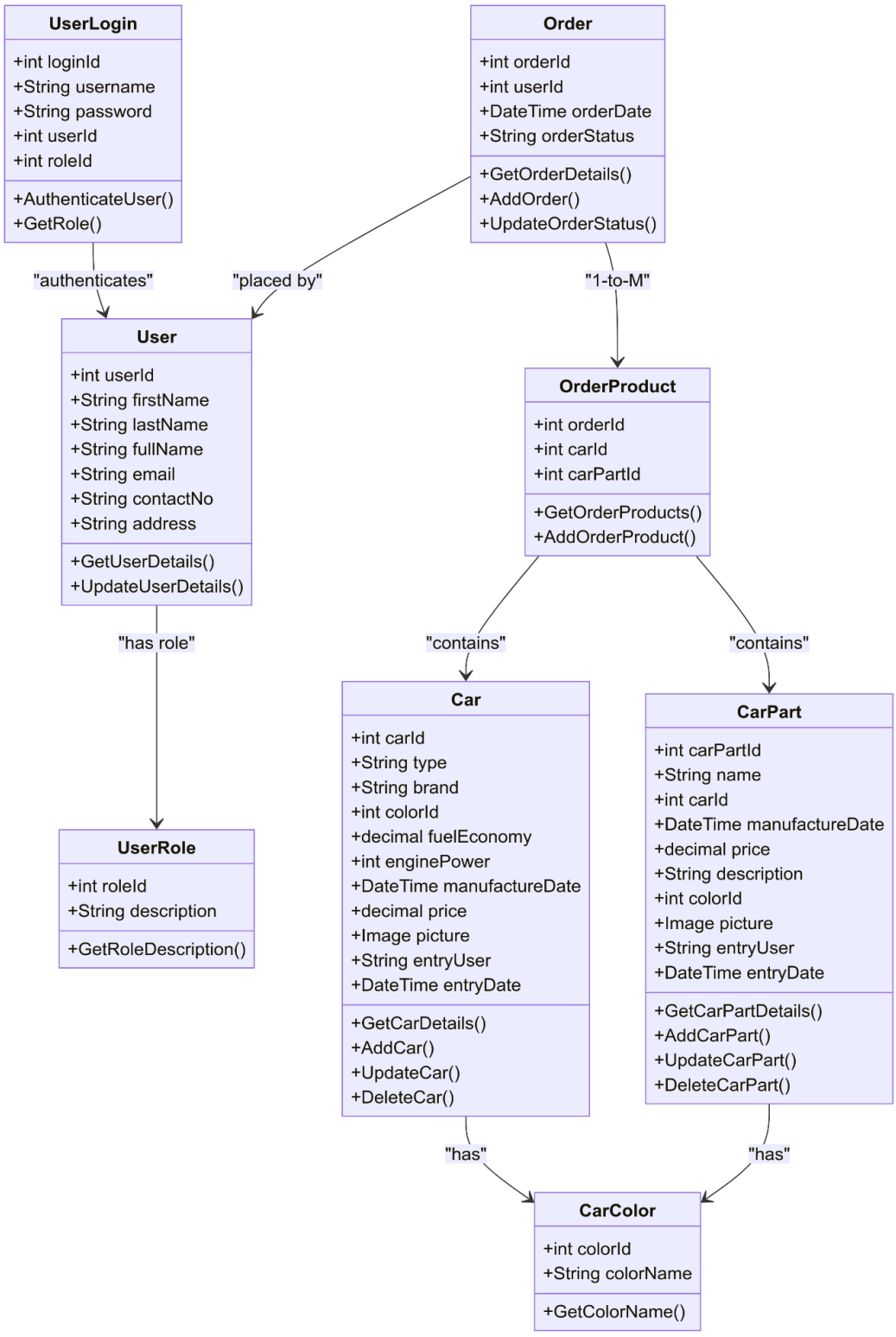
Table of Contents

1. Introduction

ABC Car Traders aims to maintain its position as a trusted provider of high-quality vehicles and transport solutions. To enhance its services and improve customer engagement, ABC Car Traders seeks to develop a software application that integrates both customer and administrative functionalities. This project, developed as part of the Application Development module, focuses on creating a user-friendly application using C# and Visual Studio, following object-oriented principles. The software allows admins to manage car and car part details, customer orders, and generate reports, while customers can register, search for vehicles or parts, place orders, and track their order status. This report outlines the development process, design decisions, and challenges encountered while building this solution.

2. Diagrams

2.1 Class Diagram



1. UserLogin

- Properties
  - loginId (int): A unique identifier for each login record.
  - username (string): The username used for authentication.
  - password (string): The password used for authentication.
  - userId (int): A reference to the user associated with this login.
  - roleId (int): A reference to the role (Admin or Customer) associated with the user.
- Methods
  - AuthenticateUser(): Validates the username and password. It ensures that the provided credentials match a user in the system.
  - GetRole(): Retrieves the role associated with the current user by roleId, mapping it to the UserRole clas.

2. UserRole

- Properties
  - roleId (int): A unique identifier for each role.
  - description (string): A textual description of the role "Admin" or "Customer."
- Methods
  - GetRoleId (): Returns the Id of the role, to determine which user role is logging into the system.

3. User

- Properties
  - userId (int): A unique identifier for each user.
  - firstName (string): The user's first name.
  - lastName (string): The user's last name.
  - fullName (string): The full name of the user, a concatenation of firstName and lastName.
  - email (string): The user's email address used to login to the system.
  - contactNo (string): The user's phone number.
  - address (string): The user's billing address.
- Methods
  - GetUserDetails(): Retrieves all the details of the user, including name, email, contact number, and address. This method is used to display user profiles.
  - UpdateUserDetails(): Allows a user to update their personal information. This is used in Profile page where the user can change their email, address, or contact information.

4. Car

- Properties
    - carId (int): A unique identifier for each car in the system.
    - type (string): The car's model or type, such as "SUV" or "Sedan".
    - brand (string): The car's manufacturer, such as "Toyota" or "Honda".
    - colorId (int): A reference to the car's color from the CarColor class.
    - fuelEconomy (decimal): The car's fuel efficiency.
    - enginePower (int): The engine's horsepower.
    - manufactureDate (DateTime): The date when the car was manufactured.
    - price (decimal): The price of the car in the marketplace.
    - picture (Image): A picture of the car, used in Order page for the customer to view.
    - entryUser (int): The userId of the user who added or updated the car details.
    - entryDate (DateTime): The date when the car details were added or last updated.
- Methods
    - GetCarDetails(): Retrieves all the properties related to the car, which is used to display car information in CarForm.
    - AddCar(): Adds a new car to the system. This method ensures the provided car information is valid before creating a new entry in the database.
    - UpdateCar(): Updates an existing car's details, allowing to make changes to cars already in the system.
    - DeleteCar(): Removes a car from the system.

5. CarPart

- Properties
    - carPartId (int): A unique identifier for each car part in the system.
    - name (string): The name of the car part, such as "Engine" or "Transmission".
    - carId (int): A reference to the car associated with this part.
    - manufactureDate (DateTime): The date when the car part was manufactured.
    - price (decimal): The cost of the car part in the marketplace.
    - description (string): A brief description of the car part, providing details such as condition or specifications.
    - colorId (int): A reference to the part's color, linking to the CarColor class.
    - picture (Image): A picture of the car part, used in the Order Page for the customer to view.
    - entryUser (int): The userId of the user who added or updated the car part details.
    - entryDate (DateTime): The date when the car part details were added or last updated.
- Methods

    - GetCarPartDetails(): Retrieves details of the car part, such as name, description, and price, which is displayed in CarParts Table and CarParts Form.
    - AddCarPart(): Adds a new car part to the system. It validates the part's information before inserting it into the database.
    - UpdateCarPart(): Updates the details of an existing car part.
    - DeleteCarPart(): Removes a car part from the system.

6. CarColor

- Properties
    - colorId (int): A unique identifier for each color in the system.
    - colorName (string): The name of the color, such as "Red" or "Blue".
- Methods
    - GetColorName(): Retrieves the color name based on the colorId. This is used for dropdowns.

7. Order

- Properties
    - orderId (int): A unique identifier for each order.
    - userId (int): A reference to the user who placed the order.
    - orderDate (DateTime): The date when the order was placed.
    - orderStatus (string): The current status of the order, such as "Placed," "Processing," or "Delivered."
- Methods
    - GetOrderDetails(): Retrieves all details of the order, including the user, status, and associated products.
    - AddOrder(): Places a new order in the system.
    - UpdateOrderStatus(): Updates the status of an existing order (e.g., from "Placed" to "Processing").

8. OrderProduct

- Properties
    - orderId (int): A reference to the order associated with the product.
    - carId (int, nullable): A reference to the car associated with the order, if applicable.
    - carPartId (int, nullable): A reference to the car part associated with the order, if applicable.
- Methods
    - GetOrderProducts(): Retrieves all products (cars or car parts) associated with a specific order.

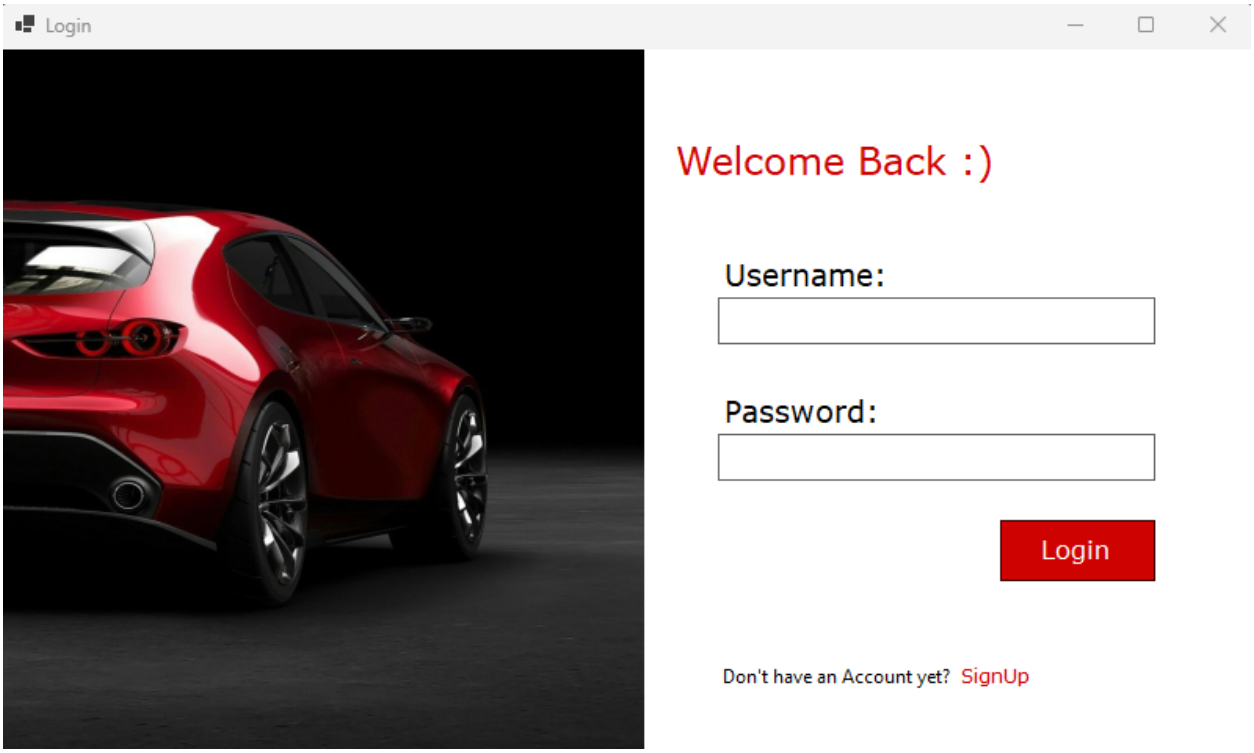
- AddOrderProduct(): Adds a car or car part to an order, linking the order to the products being purchased.

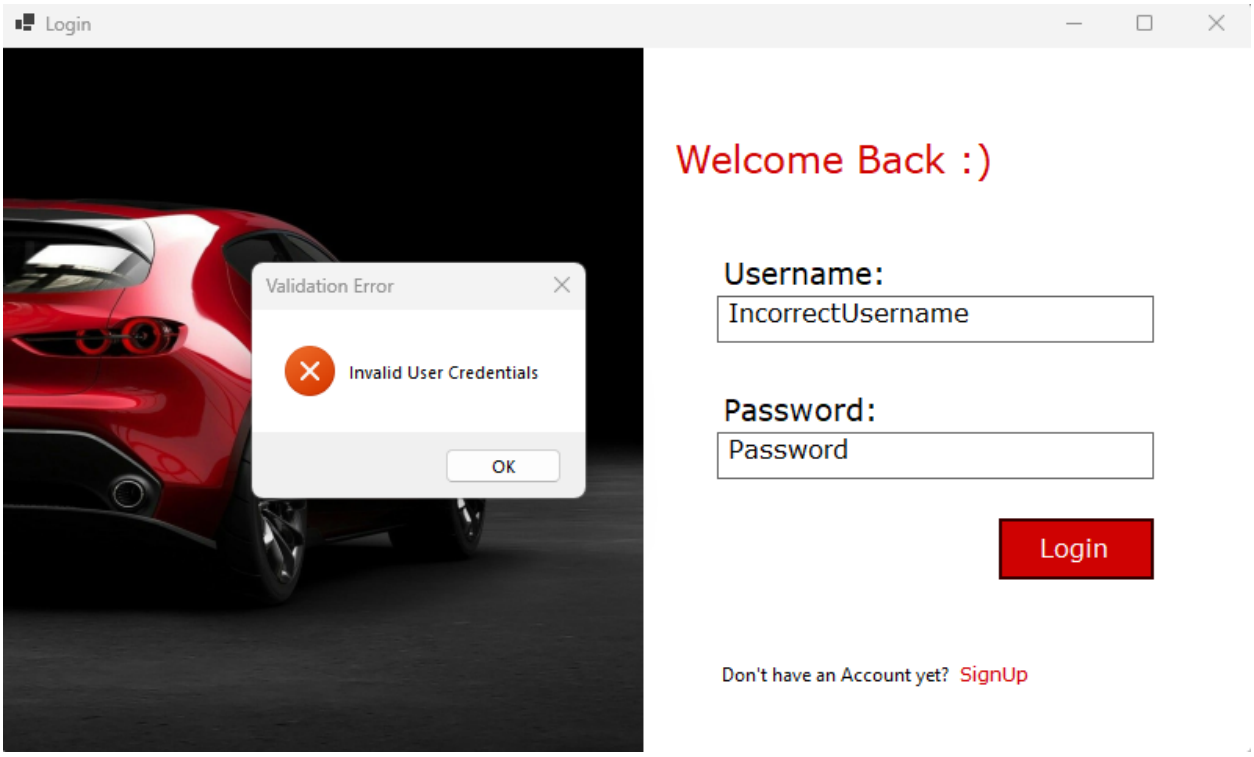
Relationships

- **UserLogin**: Manages the user's authentication and role information.
- **UserRole**: Defines the user's role (admin or customer) and manages permissions.
- **User**: Manages the personal details of users in the system.
- **Car, CarPart, CarColor**: Handles car and car part information, including their colors and associations.
- **Order, OrderProduct**: Manages customer orders and the specific products (cars or car parts) linked to each order.

3. User Interfaces and Code

Login Page





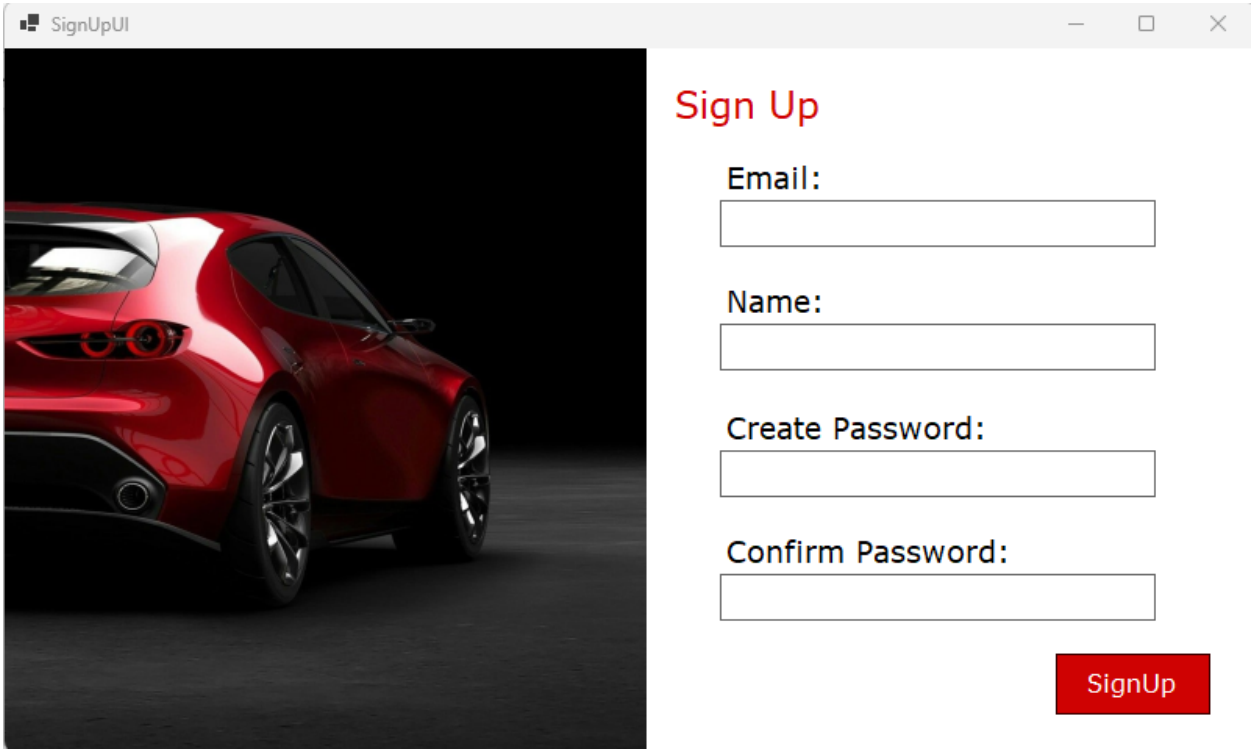

```
using ABCCarTraders.Properties;
using System.Text.RegularExpressions;
using Microsoft.Data.SqlClient;

namespace ABCCarTraders
{
    8 references
    public partial class LoginUI : Form
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-I8O0T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;Integrated Security=True;Trust Server " +
            "Certificate=True");    //CONNECTION

        3 references
        public LoginUI()
        {
            InitializeComponent();
        }
```

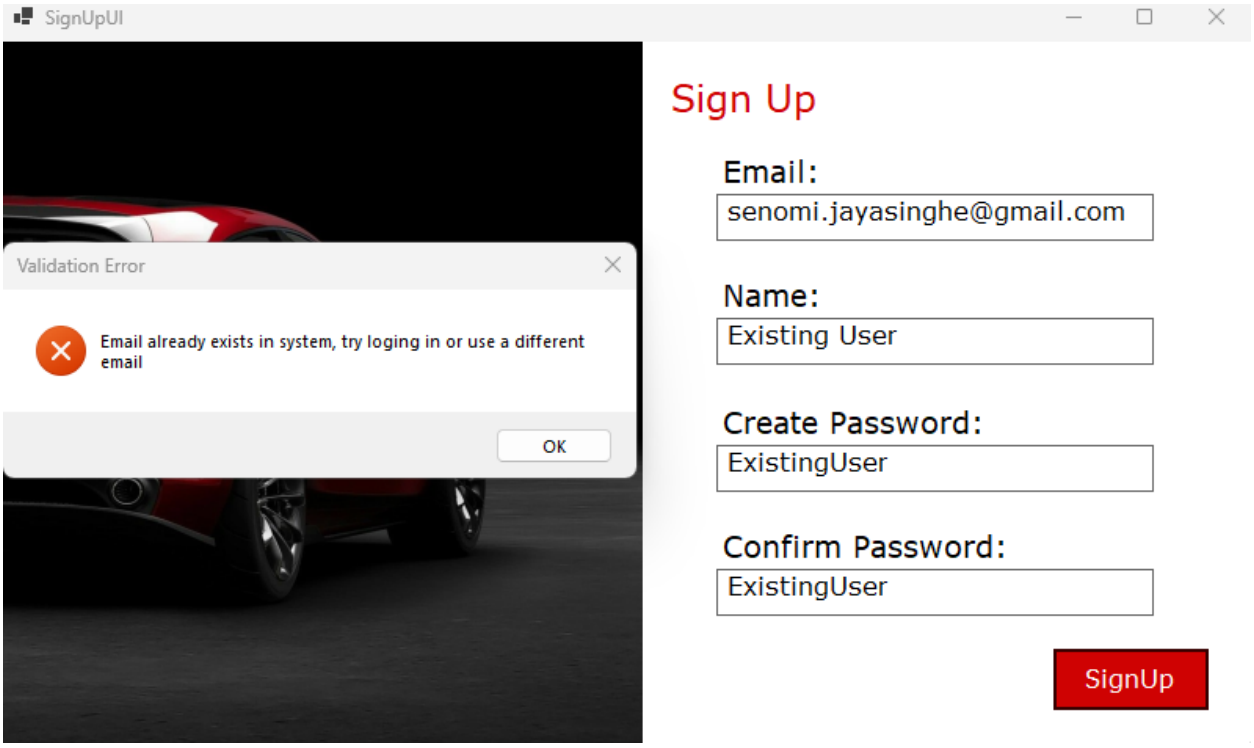
```
        private void btnLogin_Click(object sender, EventArgs e)
        {
            if (Validation()) //CHECK FOR VALIDATION
            {
                string userName = txtUsername.Text;
                string password = txtPassword.Text; //GET USRNAME AND PASSWORD

                con.Open();
                string SelectUserQuery = "Select count(1) as Count from UserLogin where Upper(login_username) = '" + userName.ToUpper() + "' " +
                    "AND login_password = '" + password + "'"; //SEARCH FOR USER
                SqlCommand cmd = new SqlCommand(SelectUserQuery, con);
                int IsUserExist = (int)cmd.ExecuteScalar();
                con.Close();
                if (IsUserExist == 1) //IF USER EXISTS
                {
                    int roleId = setSession(userName); //SET SESSION
                    if (roleId == 1) //For customers
                    {
                        CustomerHomeUI customerHomeUI = new CustomerHomeUI();
                        customerHomeUI.Show();
                    }
                    else if (roleId == 2) //For Admins
                    {
                        AdminHomeUI adminHomeUI = new AdminHomeUI();
                        adminHomeUI.Show();
                    }
                    con.Close();
                    this.Hide();
                }
                else
                {   //ERROR MESSAGE
                    MessageBox.Show("Invalid User Credentials", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
```

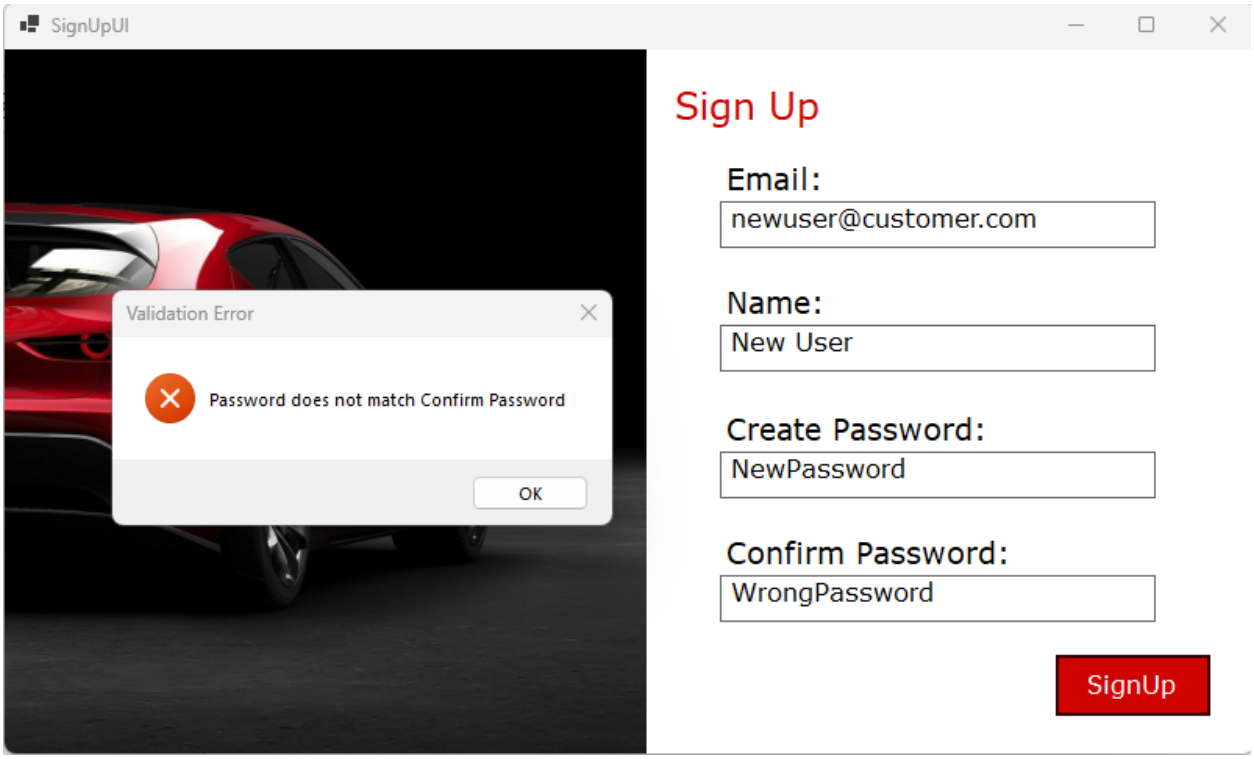
```
1 reference
private void btnSignUp_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{   //SIGNIN BUTTON REDIRECT SIGNIN PAGE
    SignUpUI signUpUI = new SignUpUI();
    signUpUI.Show();

    this.Hide();
}

1 reference
private bool Validation()
{   //VALIDATION
    ErrorProvider errorProvider = new ErrorProvider();
    errorProvider.Clear();
    if (string.IsNullOrWhiteSpace(txtUsername.Text))
    {
        //Required Email Validation
        errorProvider.SetError(txtUsername, "Username is required.");
        txtUsername.Focus();
        return false;
    }
    else if (string.IsNullOrWhiteSpace(txtPassword.Text))
    {
        //Required Password Validation
        errorProvider.SetError(txtPassword, "Password is required.");
        txtPassword.Focus();
        return false;
    }
    return true;
}
```

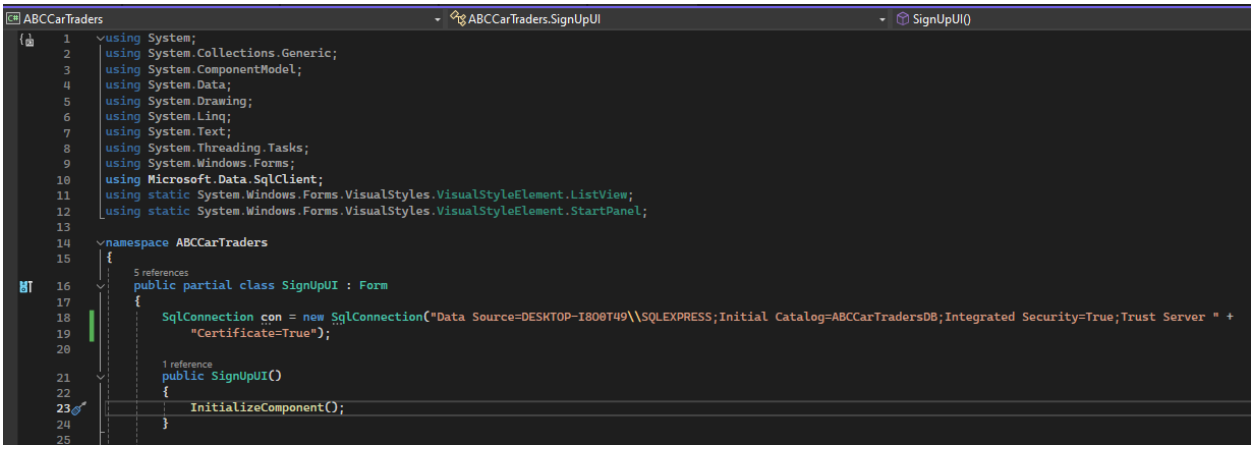
```
1 reference
private int setSession(string userName)
{   //SET SESSION
    con.Open();
    int loginRoleId = 0;
    string getSessionQuery = "Select login_role_id, login_user_id from UserLogin where Upper(login_username) = '" + userName.ToUpper() + "'";
    //GET ROLE OF USER
    try
    {
        SqlCommand cmd = new SqlCommand(getSessionQuery, con);
        using (SqlDataReader reader = cmd.ExecuteReader())
        {
            if (reader.Read())
            {
                loginRoleId = reader.GetInt32(reader.GetOrdinal("login_role_id"));
                int loginUserId = reader.GetInt32(reader.GetOrdinal("login_user_id"));

                GlobalSession.UserName = userName; //SET SESSION TO GLOBAL SESSION
                GlobalSession.RoleId = loginRoleId;
                GlobalSession.UserId = loginUserId;
            }
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("An error occurred: " + ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    return loginRoleId;
}
}
}
```

Signup Page (Customer Registration)









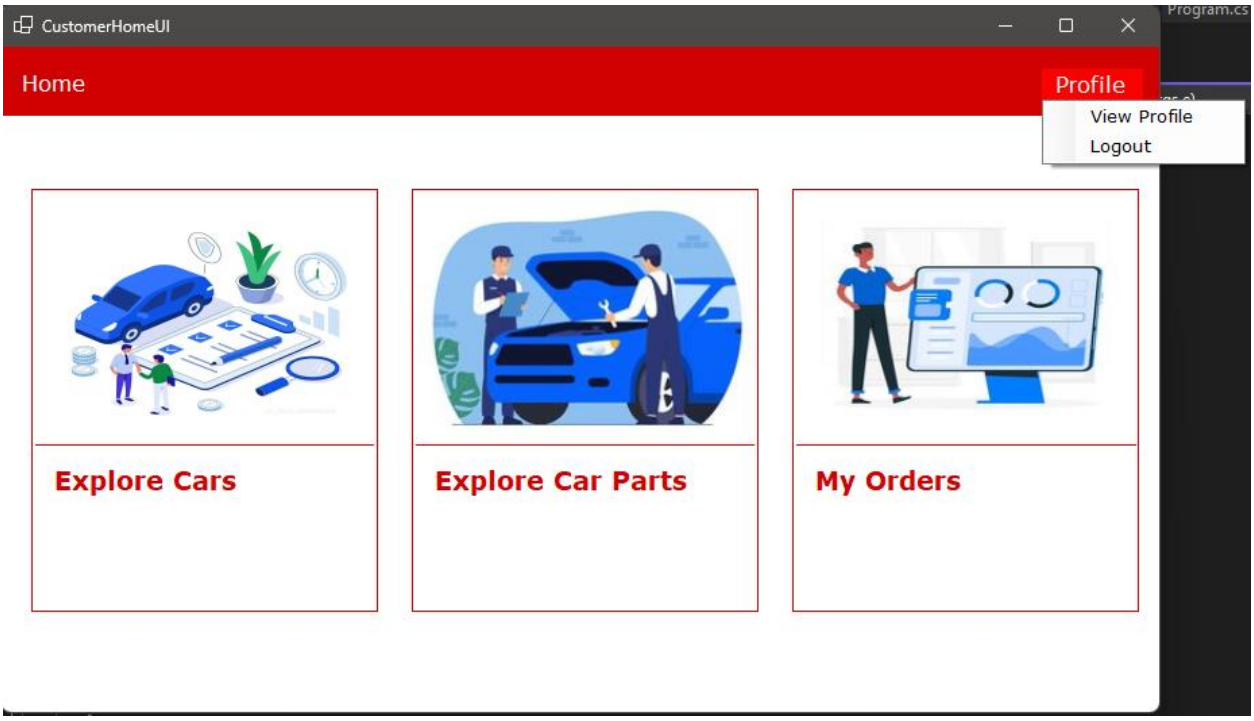
```
ABCCarTraders
ABCCarTraders.SignUpUI
SignUpUI()
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Data.SqlClient;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace ABCCarTraders
{
    5 references
    public partial class SignUpUI : Form
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-I8O0T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;Integrated Security=True;Trust Server " +
            "Certificate=True");

        1 reference
        public SignUpUI()
        {
            InitializeComponent();
        }
```

```
1 reference
private void btnSignUp_Click(object sender, EventArgs e)
{
    if (Validation()) //CHECK VALIDATION
    {
        string username = txtEmail.Text;
        string fullname = txtName.Text;
        string password = txtConfirmPassword.Text; //GET VALUES

        if (!IsEmailExist(username)) //IF THE USER IS NOT AN EXISTING USER
        {
            EnterDetails(username, fullname, password); //Register User to System
            CustomerHomeUI customerHomeUI = new CustomerHomeUI();
            customerHomeUI.Show();
            this.Hide();
        }
        else
        {
            MessageBox.Show("Email already exists in system, try loging in or use a different email", "Validation Error",
                MessageBoxButtons.OK, MessageBoxIcon.Error); //ERROR MESSAGE
        }
    }
}
```

```
1 reference
private bool Validation() //CHECK VALIDATION
{
    ErrorProvider errorProvider = new ErrorProvider();
    errorProvider.Clear();
    if (string.IsNullOrWhiteSpace(txtEmail.Text))  //Required Email Validation
    {
        errorProvider.SetError(txtEmail, "Email is required.");
        txtEmail.Focus();
        return false;
    }
    else if (string.IsNullOrWhiteSpace(txtName.Text)) //Required Password Validation
    {
        errorProvider.SetError(txtName, "Name is required.");
        txtName.Focus();
        return false;
    }
    else if (string.IsNullOrWhiteSpace(txtNewPassword.Text)) //Required Password Validation
    {
        errorProvider.SetError(txtNewPassword, "Enter a Password.");
        txtNewPassword.Focus();
        return false;
    }
    else if (string.IsNullOrWhiteSpace(txtConfirmPassword.Text)) //Required Confirm Password Validation
    {
        errorProvider.SetError(txtConfirmPassword, "Please confirm your Password.");
        txtConfirmPassword.Focus();
        return false;
    }
    else if (!(txtNewPassword.Text == txtConfirmPassword.Text)) //CHECK IF PASSWORD IS EQUAL TO CONFIRMED PASSWORD
    {
        MessageBox.Show("Password does not match Confirm Password", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        txtConfirmPassword.Focus();
        return false;
    }
    return true;
}
```

```
private bool IsEmailExist(string username)
{
    //Check whether email already exists in the system
    con.Open();
    string SelectUserQuery = "Select count(1) as Count from UserLogin where Upper(login_username) = '" + username.ToUpper() + "'";
    SqlCommand cmd = new SqlCommand(SelectUserQuery, con);
    int IsExist = (int)cmd.ExecuteScalar();
    con.Close();
    return Convert.ToBoolean(IsExist);
}

1 reference
private void EnterDetails(string username, string fullname, string password)
{
    //Entering User Details to System
    string InsertUserDetails = "Insert into Users (user_fullname, user_email) VALUES ('" + fullname + "', '" + username + "'); " +
        "select CAST(scope_identity() as int)";
    SqlCommand cmduser = new SqlCommand(InsertUserDetails, con);
    int user_id = (int)cmduser.ExecuteScalar(); //Get user_id

    //Entering Login Credentials to the System
    string InsertLoginDetails = "Insert into UserLogin (login_username, login_password, login_user_id, login_role_id) " +
        "VALUES ('" + username + "', '" + password + "', " + user_id + ", 1)"; //Only customers can signup, hence the user_role is 1 (Customer)

    SqlCommand cmdlogin = new SqlCommand(InsertLoginDetails, con);
    cmdlogin.ExecuteNonQuery();

    //Set Session
    GlobalSession.UserName = username;
    GlobalSession.UserId = user_id;
    GlobalSession.RoleId = 1;
}
```

Customer Dashboard



```
private void btnHome_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{   //REDIRECT TO CUSTOMER HOME PAGE
    CustomerHomeUI customerHomeUI = new CustomerHomeUI();
    customerHomeUI.Show();
    this.Hide();
}

1 reference
private void btnMenu_Click(object sender, EventArgs e)
{   //SHOW DROP DOWN OF MENU WHEN PROFILE IS CLICKED
    ddlMenu.Show(btnMenu, new Point(0, btnMenu.Height));
}

1 reference
private void ddlProfile_Click(object sender, EventArgs e)
{   //REDIRECT TO USER PROFILE
    UserProfileUI userProfileUI = new UserProfileUI();
    userProfileUI.Show();
}

1 reference
private void ddlLogOut_Click(object sender, EventArgs e)
{
    // Clear session
    GlobalSession.UserName = string.Empty;
    GlobalSession.RoleId = 0;
    GlobalSession.UserId = 0;

    // Redirect to login form
    LoginUI loginUI = new LoginUI();
    loginUI.Show();
    this.Hide();
}
```

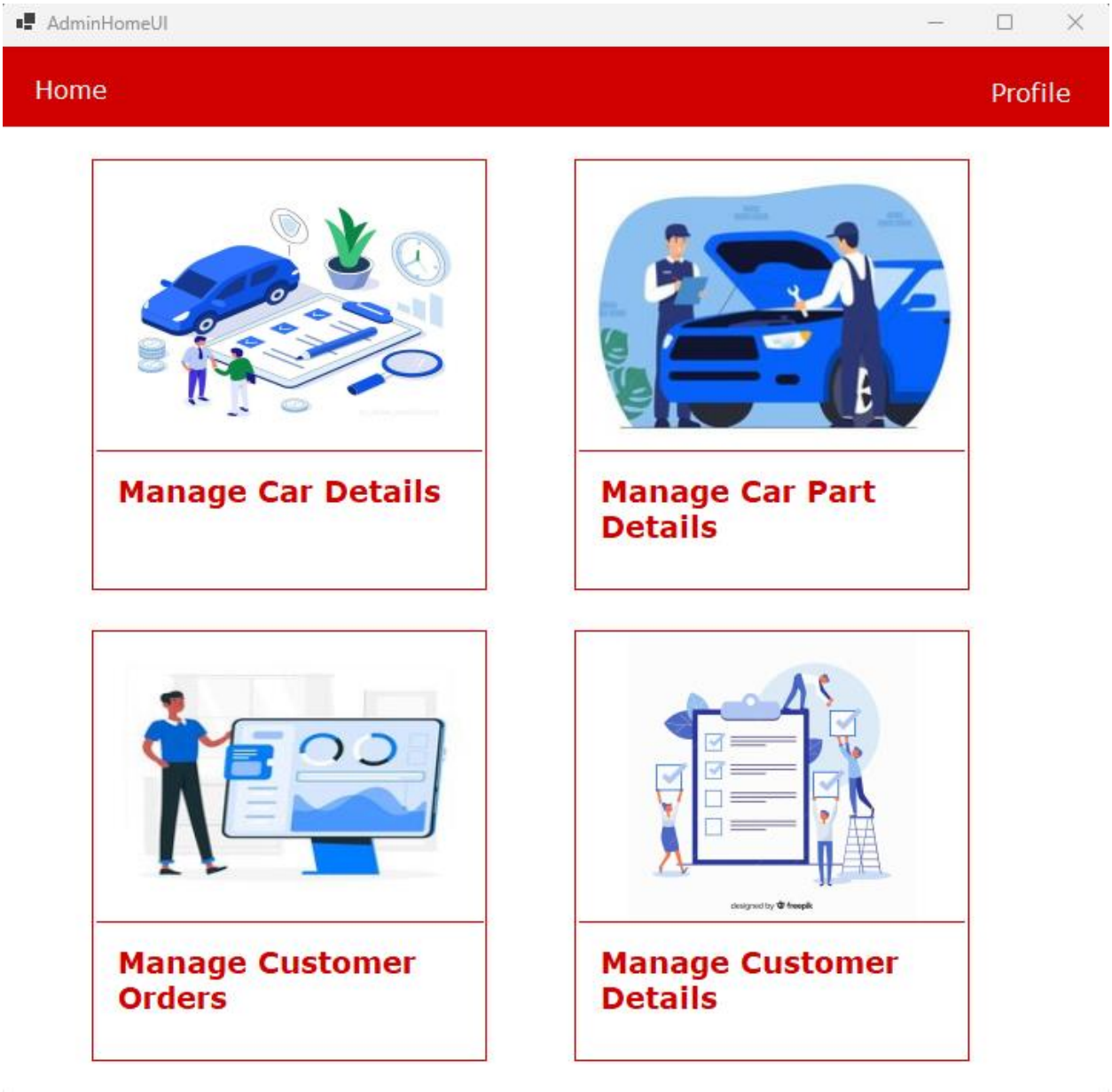
```
1 reference
private void pnlCars_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlCars.Width - 1, pnlCars.Height - 1);
    }
}

1 reference
private void picCar_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picCar.Height - 1, picCar.Width - 1, picCar.Height - 1);
    }
}
```

```
1 reference
private void pnlCarParts_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlCarParts.Width - 1, pnlCarParts.Height - 1);
    }
}

1 reference
private void picCarparts_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picCarparts.Height - 1, picCarparts.Width - 1, picCarparts.Height - 1);
    }
}
```

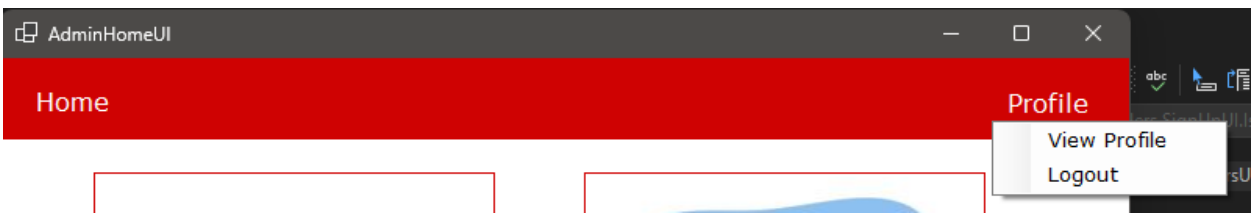
```
1 reference
private void pnlOrders_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlOrders.Width - 1, pnlOrders.Height - 1);
    }
}

1 reference
private void picOrders_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picOrders.Height - 1, picOrders.Width - 1, picOrders.Height - 1);
    }
}

1 reference
private void pnlCars_Click(object sender, EventArgs e)
{   //REDIRECT TO CAR SEARCH AND ORDER PAGE
    CarOrder carOrder = new CarOrder();
    carOrder.Show();
    this.Hide();
}
```

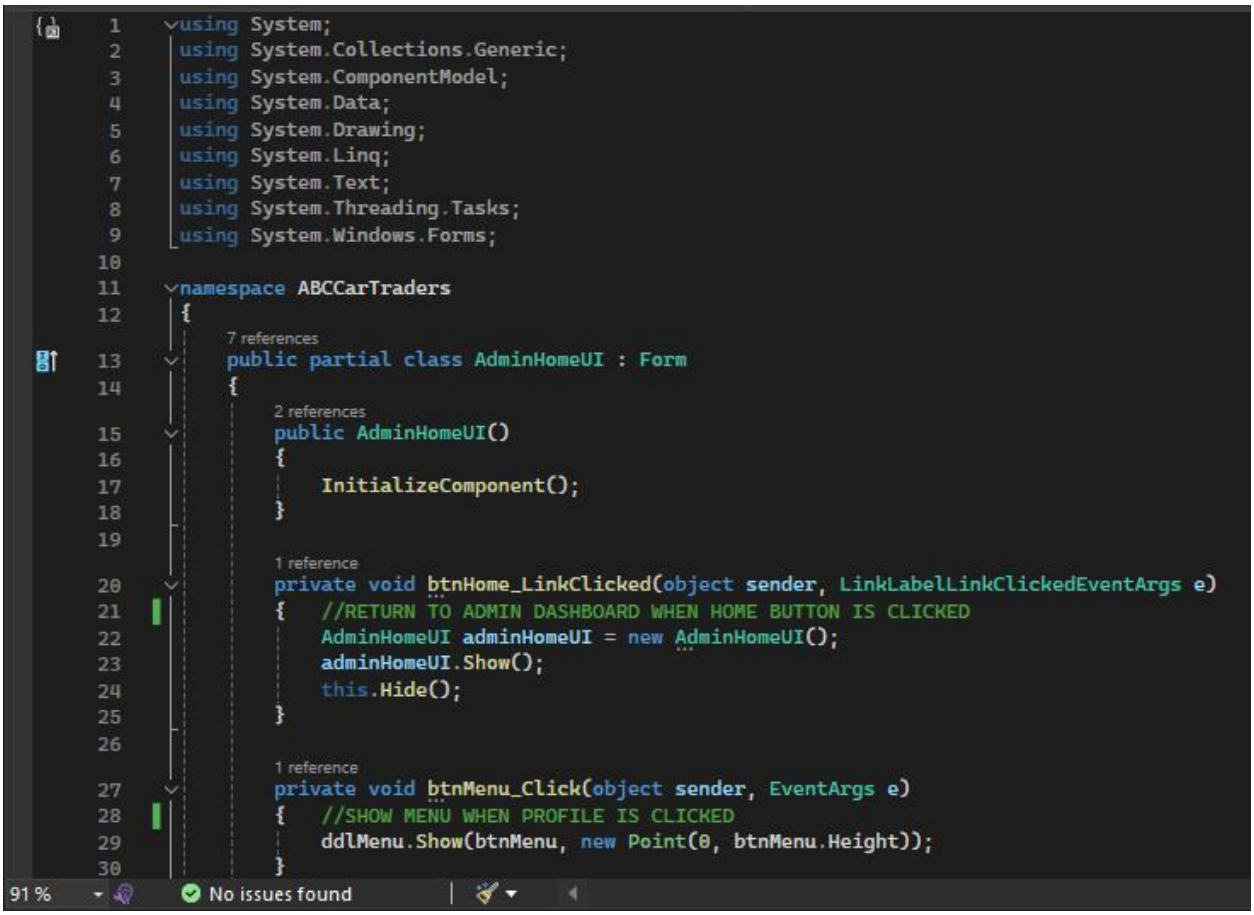
```
1 reference
private void pnlCarParts_Click(object sender, EventArgs e)
{   //REDIRECT TO CARPART SEARCH AND ORDER PAGE
    CarPartOrder carPartOrder = new CarPartOrder();
    carPartOrder.Show();
    this.Hide();
}

1 reference
private void pnlOrders_Click(object sender, EventArgs e)
{   //TRACK AND VIEW ORDERS
    OrdersUI ordersUI = new OrdersUI();
    ordersUI.Show();
    this.Hide();
}
```

Admin Dashboard



When Profile Button is Clicked





```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ABCCarTraders
{
    7 references
    public partial class AdminHomeUI : Form
    {
        2 references
        public AdminHomeUI()
        {
            InitializeComponent();
        }

        1 reference
        private void btnHome_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {   //RETURN TO ADMIN DASHBOARD WHEN HOME BUTTON IS CLICKED
            AdminHomeUI adminHomeUI = new AdminHomeUI();
            adminHomeUI.Show();
            this.Hide();
        }

        1 reference
        private void btnMenu_Click(object sender, EventArgs e)
        {   //SHOW MENU WHEN PROFILE IS CLICKED
            ddlMenu.Show(btnMenu, new Point(0, btnMenu.Height));
        }
```

```
1 reference
private void ddlProfile_Click(object sender, EventArgs e)
{
    //WHEN PROFILE IS SELECTED GO TO USER PROFILE
    UserProfileUI userProfileUI = new UserProfileUI();
    userProfileUI.Show();
}

1 reference
private void ddlLogOut_Click(object sender, EventArgs e)
{
    // Clear session
    GlobalSession.UserName = string.Empty;
    GlobalSession.RoleId = 0;
    GlobalSession.UserId = 0;

    // Redirect to login form
    LoginUI loginUI = new LoginUI();
    loginUI.Show();
    this.Hide();
}

1 reference
private void pnlCars_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlCars.Width - 1, pnlCars.Height - 1);
    }
}
```

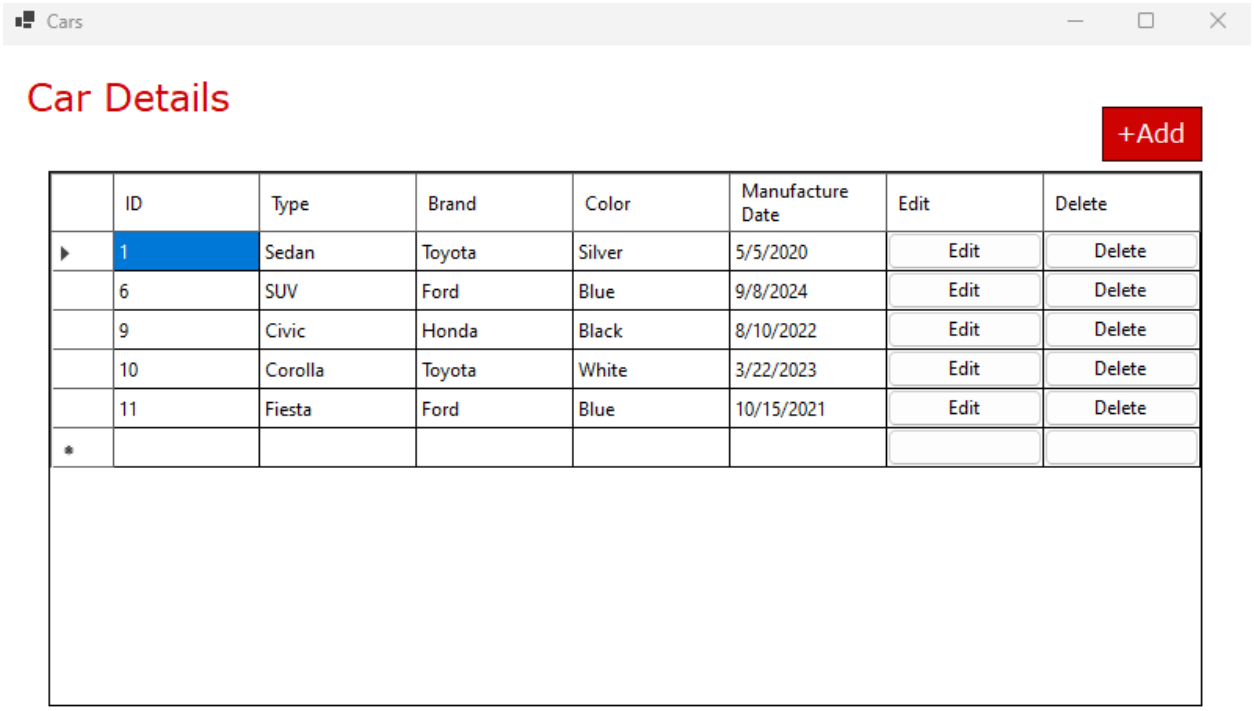
```
1 reference
private void pnlCars_Click(object sender, EventArgs e)
{   //NAVIGATE TO CARS INTERFACE
    CarsUI carsUI = new CarsUI();
    carsUI.Show();
    this.Hide();
}

1 reference
private void picCar_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picCar.Height - 1, picCar.Width - 1, picCar.Height - 1);
    }
}

1 reference
private void pnlCarParts_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlCarParts.Width - 1, pnlCarParts.Height - 1);
    }
}
```

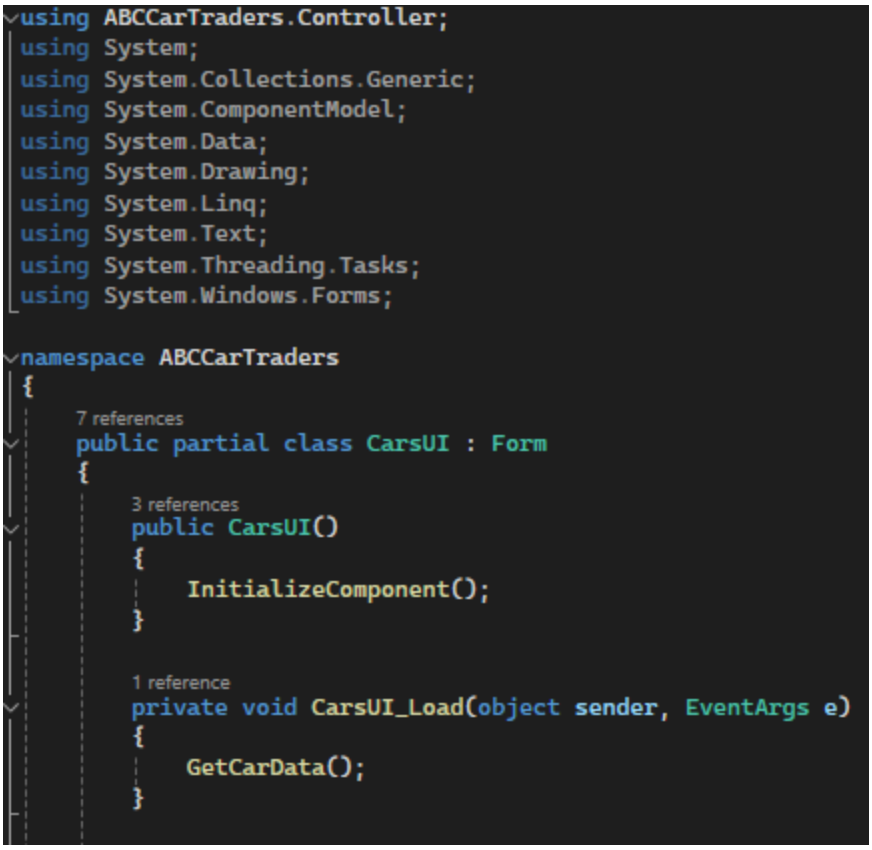
```
private void picCarparts_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picCarparts.Height - 1, picCarparts.Width - 1, picCarparts.Height - 1);
    }
}

1 reference
private void pnlCarParts_Click(object sender, EventArgs e)
{
    //NAVIGATE TO CAR PARTS INTERFACE
    CarPartsUI carPartsUI = new CarPartsUI();
    carPartsUI.Show();
    this.Hide();
}

1 reference
private void pnlOrders_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlOrders.Width - 1, pnlOrders.Height - 1);
    }
}
```

```
private void picOrders_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picOrders.Height - 1, picOrders.Width - 1, picOrders.Height - 1);
    }
}

1 reference
private void pnlOrders_Click(object sender, EventArgs e)
{
    //NAVIGATE TO ORDERS INTERFACE
    OrdersUI ordersUI = new OrdersUI();
    ordersUI.Show();
    this.Hide();
}

1 reference
private void picCustomer_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the bottom border
    {
        // Style line at the bottom of the PictureBox
        e.Graphics.DrawLine(pen, 0, picCustomer.Height - 1, picCustomer.Width - 1, picCustomer.Height - 1);
    }
}
```

```
1 reference
private void pnlCustomer_Paint(object sender, PaintEventArgs e)
{
    Color customRed = Color.FromArgb(207, 2, 2);

    // Created a Pen object with the custom color and width
    using (Pen pen = new Pen(customRed, 1)) // 1 is the thickness of the border
    {
        // Draw the rectangle border inside the panel
        e.Graphics.DrawRectangle(pen, 0, 0, pnlCustomer.Width - 1, pnlCustomer.Height - 1);
    }
}

1 reference
private void pnlCustomer_Click(object sender, EventArgs e)
{
    //NAVIGATE TO CUSTOMER INTERFACE
    CustomerUI customerUI = new CustomerUI();
    customerUI.Show();
    this.Hide();
}
```

Admin Managing Car Details




using ABCCarTraders.Controller;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace ABCCarTraders
{
7 references
public partial class CarsUI : Form
{
3 references
public CarsUI()
{
InitializeComponent();
}
1 reference
private void CarsUI_Load(object sender, EventArgs e)
{
GetCarData();
}

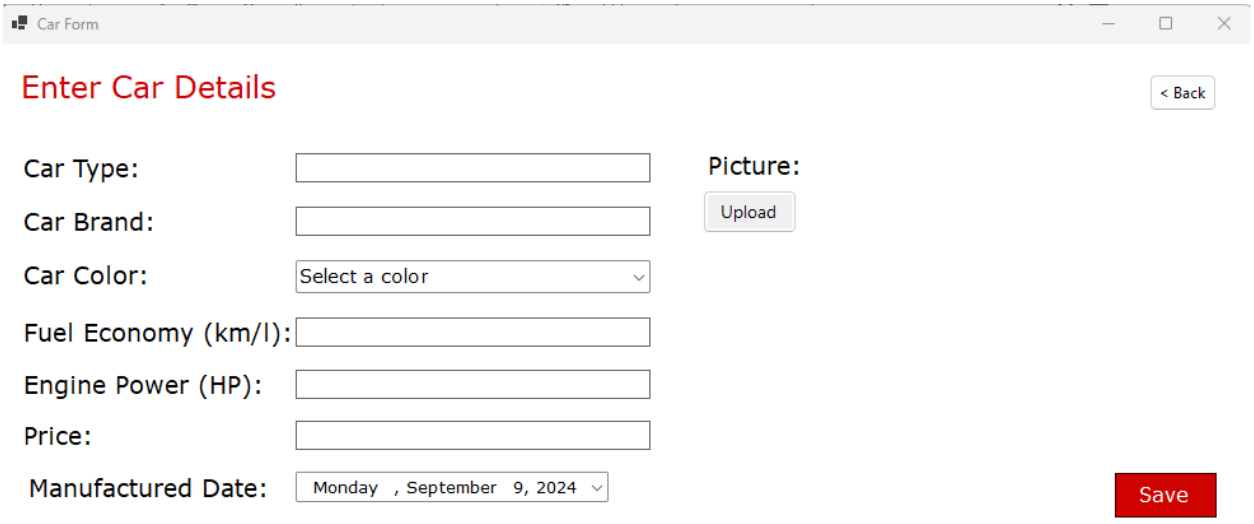
```
private void GetCarData()
{
    CarController carController = new CarController();
    grdCars.DataSource = carController.GetCars();
    //Set Data to Table
    grdCars.Columns["car_id"].HeaderText = "ID";
    grdCars.Columns["car_type"].HeaderText = "Type";
    grdCars.Columns["car_brand"].HeaderText = "Brand";
    grdCars.Columns["color_name"].HeaderText = "Color";
    grdCars.Columns["car_manufacture_date"].HeaderText = "Manufacture Date";

    if (GlobalSession.RoleId == 2) //IF THE USER IS AN ADMIN SHOW EDIT AND DELETE BUTTONS
    {
        DataGridViewButtonColumn editButtonColumn = new DataGridViewButtonColumn
        {
            HeaderText = "Edit",
            Text = "Edit",
            UseColumnTextForButtonValue = true,
            Name = "EditColumn"
        };
        grdCars.Columns.Add(editButtonColumn);

        DataGridViewButtonColumn deleteButtonColumn = new DataGridViewButtonColumn
        {
            HeaderText = "Delete",
            Text = "Delete",
            UseColumnTextForButtonValue = true,
            Name = "DeleteColumn"
        };
        grdCars.Columns.Add(deleteButtonColumn);
    }
    grdCars.CellClick += grdCars_CellClick;
}
```

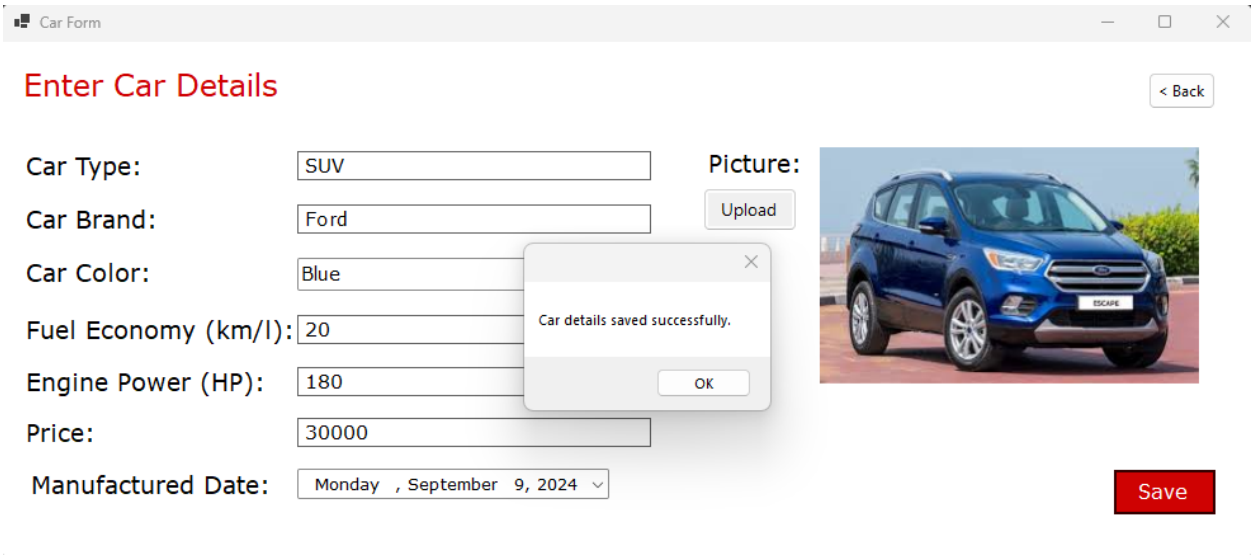
```
1 rererence
private void grdCars_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0) // Ensure the user clicked on a valid row
    {
        int carId = Convert.ToInt32(grdCars.Rows[e.RowIndex].Cells["car_id"].Value);
        // Determine which column was clicked (Edit, Delete, or Order)
        if (grdCars.Columns[e.ColumnIndex].Name == "EditColumn")
        {
            OpenCarForm(carId, "U");  // Open in Edit Mode
        }
        else if (grdCars.Columns[e.ColumnIndex].Name == "DeleteColumn")
        {
            OpenCarForm(carId, "D");  // Open in Delete Mode
        }
    }
}

2 references
private void OpenCarForm(int carId, string mode)
{
    // Open CarFormUI in the specified mode (D: Delete, U: Update)
    CarFormUI carForm = new CarFormUI();
    carForm.SetMode(carId, mode);
    carForm.Show();
}

1 reference
private void btnAdd_Click(object sender, EventArgs e)
{   //OPEN AN EMPTY CAR FORM
    CarFormUI carFormUI = new CarFormUI();
    carFormUI.Show();
    this.Hide();
}
```

Controller

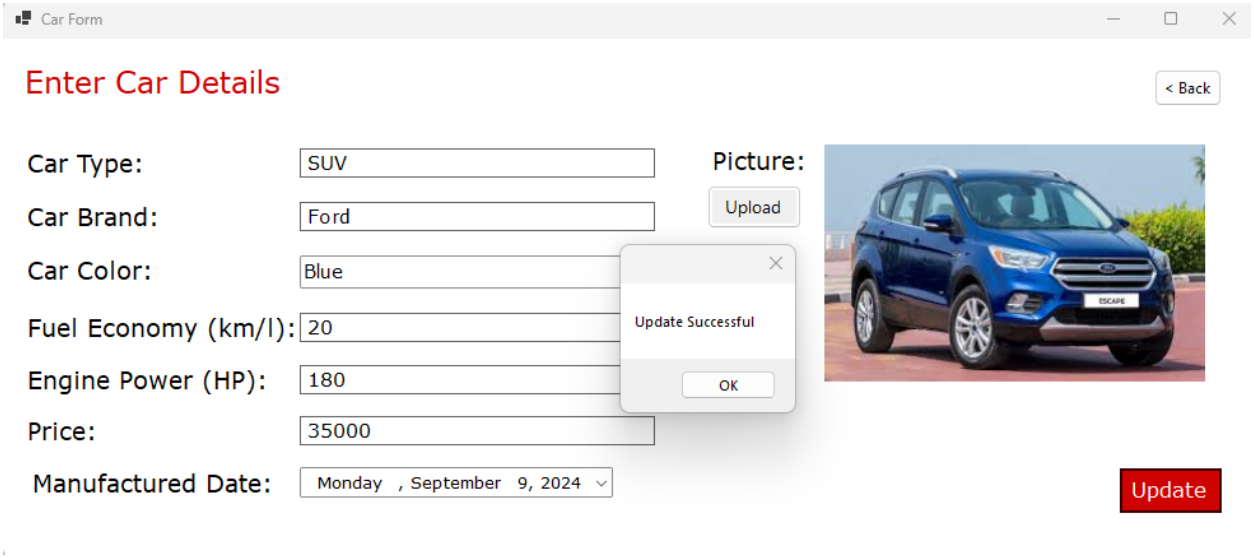
```
12 references
public class CarController
{
    SqlConnection con = new SqlConnection("Data Source=DESKTOP-I8O0T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;Integrated Security=True;Tr

    1 reference
    public DataTable GetCars()
    {
        string selectCars = "SELECT c.car_id, c.car_type, c.car_brand, cc.color_name, c.car_manufacture_date FROM Car c JOIN CarColor cc " +
            "ON c.car_color = cc.color_id";
        DataTable dtCars = new DataTable();
        try
        {
            con.Open();
            using (SqlCommand cmd = new SqlCommand(selectCars, con))
            {
                using (SqlDataAdapter da = new SqlDataAdapter(cmd))
                {
                    da.Fill(dtCars); // Fills the DataTable with the result from the database
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
        finally
        {
            con.Close();
        }
        return dtCars;
    }
```

When Clicked on Add, the CarFormUI is displayed:





When Edit or Delete button is clicked, Form is Opened in Edit or Delete Mode. Hence, if its opened in Edit mode, the Update Query is carried out otherwise the Delete Query.



Car Form

Enter Car Details

< Back

Car Type: SUV

Car Brand: Ford

Car Color: Blue

Fuel Economy (km/l): 20

Engine Power (HP): 180

Price: 35000

Manufactured Date: Monday , September 9, 2024

Picture:

Delete Successful

OK

Delete

Customer Ordering Cars

CarOrder

Search and Order Cars

Type Brand Color Select a color

Search

	ID	Type	Brand	Color	Manufacture Date	Order
▶	1	Sedan	Toyota	Silver	5/5/2020	Order
	6	SUV	Ford	Blue	9/8/2024	Order
	9	Civic	Honda	Black	8/10/2022	Order
	10	Corolla	Toyota	White	3/22/2023	Order
	11	Fiesta	Ford	Blue	10/15/2021	Order
*						

```
private void LoadColorsIntoComboBox()
{
    try
    {
        con.Open();
        string query = "SELECT color_id, color_name FROM CarColor"; // Query to fetch color_id and color_name from the database
        SqlCommand cmd = new SqlCommand(query, con);

        using (SqlDataReader reader = cmd.ExecuteReader()) // Execute the command and read the results
        {
            List<CarColor> colors = new List<CarColor>(); // Create a list to hold color items

            colors.Add(new CarColor { ColorId = null, ColorName = "Select a color" }); // Add "Select a color" as default

            while (reader.Read()) // Populate the list with colors from the database
            {
                colors.Add(new CarColor
                {
                    ColorId = (int)reader["color_id"],
                    ColorName = reader["color_name"].ToString()
                });
            }
            // Bind the list to the ComboBox
            ddlColors.DataSource = colors;
            ddlColors.DisplayMember = "ColorName";  // Show the color name
            ddlColors.ValueMember = "ColorId";      // Use color_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

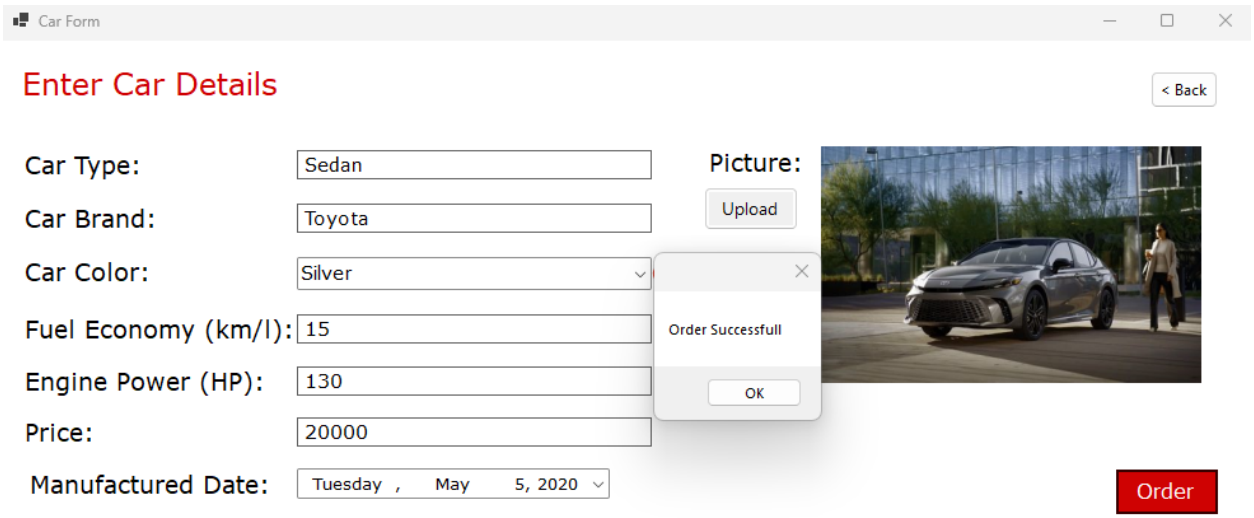
```
    private void CarOrder_Load(object sender, EventArgs e)
    {
        LoadColorsIntoComboBox(); //LOAD COLORS INTO COMBO BOX
    }

    1 reference
    private void btnSearch_Click(object sender, EventArgs e)
    {
        string type = txtType.Text; //GET VALUES FROM INPUTS
        string brand = txtBrand.Text;
        int color = Convert.ToInt32(ddlColors.SelectedValue);
        CarController carController = new CarController(); //SEARCH QUERY
        grdCars.DataSource = carController.SearchCar(type, brand, color); //SET DATA TO TABLE
        grdCars.Columns["car_id"].HeaderText = "ID";
        grdCars.Columns["car_type"].HeaderText = "Type";
        grdCars.Columns["car_brand"].HeaderText = "Brand";
        grdCars.Columns["color_name"].HeaderText = "Color";
        grdCars.Columns["car_manufacture_date"].HeaderText = "Manufacture Date";

        int roleId = 1;
        if (GlobalSession.RoleId == 1) //IF CUSTOMER
        {   //DISPLAY ORDER BUTTON
            DataGridViewButtonColumn orderButtonColumn = new DataGridViewButtonColumn
            {
                HeaderText = "Order",
                Text = "Order",
                UseColumnTextForButtonValue = true,
                Name = "OrderColumn"
            };
            grdCars.Columns.Add(orderButtonColumn);
        }
        grdCars.CellClick += grdCars_CellClick;
    }
```

```
1 reference
private void grdCars_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0) // Ensure the user clicked on a valid row
    {
        int carId = Convert.ToInt32(grdCars.Rows[e.RowIndex].Cells["car_id"].Value);

        if (grdCars.Columns[e.ColumnIndex].Name == "OrderColumn")
        {
            OpenCarForm(carId, "O");  // Open in Order Mode
        }
    }
}

1 reference
private void OpenCarForm(int carId, string mode)
{
    // Open CarFormUI in the specified mode (O: Order)
    CarFormUI carForm = new CarFormUI();
    carForm.SetMode(carId, mode);
    carForm.Show();
}
```

Controller

```
1 reference
public DataTable SearchCar(string type, string brand, int color)
{
    con.Open();
    string searchQuery = "SELECT c.car_id, c.car_type, c.car_brand, cc.color_name, c.car_manufacture_date FROM Car c JOIN CarColor cc " +
        "ON c.car_color = cc.color_id WHERE (c.car_type = '" + type + "' OR '' = '" + type + "') " +
        "AND (c.car_brand = '" + brand + "' OR '' = '" + brand + "') AND (c.car_color = " + color + " OR 0 = " + color + ")";
    DataTable dtCars = new DataTable();
    try
    {
        using (SqlCommand cmd = new SqlCommand(searchQuery, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dtCars); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dtCars;
}
```

When the Order button is clicked the Car Form will be opened in Order Mode.



Car Form Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using Microsoft.Data.SqlClient;
using ABCCarTraders.Model;
using ABCCarTraders.Controller;

namespace ABCCarTraders
{
    8 references
    public partial class CarFormUI : Form
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-I8O0T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;I
        //INITIALIZE VARIABLES FOR MODES EDIT, DELETE, AND ORDER
        public int carId;
        public string mode;
        public byte[] CarPicture;

        3 references
        public CarFormUI()
        {
            InitializeComponent();
        }
```

```
2 references
public void SetMode(int carId, string mode)
{
    this.carId = carId;
    this.mode = mode;
    LoadCarData(); // Load data based on the carId
    if (mode == "U") //UPDATE MODE
    {
        btnSave.Text = "Update";
    }
    else if (mode == "D") //DELETE MODE
    {
        btnSave.Text = "Delete";
    }
    else if (mode == "O") //ORDER MODE
    {
        btnSave.Text = "Order";
    }
}

1 reference
private void CarFormUI_Load(object sender, EventArgs e)
{
    LoadColorsIntoComboBox(); //SET COLORS TO COMBO BOX
}
```

```
private void LoadColorsIntoComboBox()
{
    try
    {
        con.Open();
        string query = "SELECT color_id, color_name FROM CarColor"; // Query to fetch color_id and color_name from the database

        SqlCommand cmd = new SqlCommand(query, con);
        using (SqlDataReader reader = cmd.ExecuteReader()) // Execute the command and read the results
        {
            List<CarColor> colors = new List<CarColor>();// Create a list to hold color items

            colors.Add(new CarColor { ColorId = null, ColorName = "Select a color" });// Add "Select a color" as default

            while (reader.Read())// Populate the list with colors from the database
            {
                colors.Add(new CarColor
                {
                    ColorId = (int)reader["color_id"],
                    ColorName = reader["color_name"].ToString()
                });
            }
            ddlColors.DataSource = colors;  // Bind the list to the ComboBox
            ddlColors.DisplayMember = "ColorName";  // Show the color name
            ddlColors.ValueMember = "ColorId";      // Use color_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

```
1 reference
private void btnUpload_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    // Set the filter to allow only image files
    openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";

    // Show the file dialog and check if the user selected a file
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        // Get the file path
        string filePath = openFileDialog.FileName;

        // Display the image in the PictureBox for preview
        picCar.Image = Image.FromFile(filePath);
        picCar.SizeMode = PictureBoxSizeMode.StretchImage;
    }

    CarPicture = GetImageAsByteArray(picCar.Image);
}

1 reference
private byte[] GetImageAsByteArray(Image image)
{ //GET PICTURE AS BYTE[]
    using (MemoryStream ms = new MemoryStream())
    {
        image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
        return ms.ToArray();
    }
}
```

```
private void LoadCarData()
{   //Set Deatils to fields
    CarController carController = new CarController();
    Car car = carController.GetCarByID(carId);

    txtCarId.Text = car.CarId.ToString();
    txtType.Text = car.Type;
    txtBrand.Text = car.Brand;
    ddlColors.SelectedValue = car.ColorId;
    txtFuelEco.Text = car.FuelEconomy.ToString();
    txtEnginePw.Text = car.EnginePower.ToString();
    dateManuDate.Value = car.ManufactureDate;
    txtPrice.Text = car.Price.ToString();
    CarPicture = car.Picture;
    if (car.Picture != null && car.Picture.Length > 0) // Ensure car.Picture is not null or empty
    {
        using (MemoryStream ms = new MemoryStream(car.Picture))
        {
            try
            {
                Image carImage = Image.FromStream(ms);
                picCar.SizeMode = PictureBoxSizeMode.StretchImage; // Set the size mode
                picCar.Image = carImage; // Display in PictureBox
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error loading image: " + ex.Message);
            }
        }
    }
    else
    {
        picCar.Image = null; // Clear the PictureBox if no image is available
```

```
1 reference
private void btnSave_Click(object sender, EventArgs e)
{
    if (ValidateCarForm()) //CHECK FOR VALIDATION
    {   //SAVE DATA TO CLASS CAR
        Car car = new Car();
        car.Type = txtType.Text;
        car.Brand = txtBrand.Text;

        CarColor selectedColor = (CarColor)ddlColors.SelectedItem;
        car.ColorId = selectedColor.ColorId.Value;

        car.FuelEconomy = Convert.ToDecimal(txtFuelEco.Text);
        car.EnginePower = Convert.ToDecimal(txtEnginePw.Text);
        car.ManufactureDate = Convert.ToDateTime(dateManuDate.Value.ToString("yyyy-MM-dd"));
        car.Price = Convert.ToDecimal(txtPrice.Text);

        car.EntryDate = DateTime.Now.ToString("yyyy-MM-dd");
        car.EntryUser = GlobalSession.UserId;

        car.Picture = CarPicture;

        if (btnSave.Text == "Save")
        {
            try
            {   //INSERT QUERY
                con.Open();
                string insertCar = "INSERT INTO Car (car_type, car_brand, car_color, car_fueleconomy, car_engine_power, car_manufacture_date, car_price, car_picture,
                    "entry_user, entry_date) " +
                        "VALUES (@carType, @carBrand, @carColor, @carFuelEconomy, @carEnginePower, @carManufactureDate, @carPrice, @carPicture, @entryUser, @entryDate)
```

```
    using (SqlCommand cmd = new SqlCommand(insertCar, con))
    {
        // Add parameters to the command
        cmd.Parameters.AddWithValue("@carType", car.Type);
        cmd.Parameters.AddWithValue("@carBrand", car.Brand);
        cmd.Parameters.AddWithValue("@carColor", car.ColorId);
        cmd.Parameters.AddWithValue("@carFuelEconomy", car.FuelEconomy);
        cmd.Parameters.AddWithValue("@carEnginePower", car.EnginePower);
        cmd.Parameters.AddWithValue("@carManufactureDate", car.ManufactureDate);
        cmd.Parameters.AddWithValue("@carPrice", car.Price);

        // Use byte array for the image data
        cmd.Parameters.AddWithValue("@carPicture", CarPicture ?? (object)DBNull.Value);

        cmd.Parameters.AddWithValue("@entryUser", car.EntryUser);
        cmd.Parameters.AddWithValue("@entryDate", car.EntryDate);
        cmd.ExecuteNonQuery();
    }
    con.Close();
    MessageBox.Show("Car details saved successfully.");
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
}
```

```
else if (btnSave.Text == "Update")
{
    car.CarId = Convert.ToInt32(txtCarId.Text);
    CarController carController = new CarController();//UPDATE QUERY
    carController.UpdateCar(car);
}
else if (btnSave.Text == "Delete")
{
    car.CarId = Convert.ToInt32(txtCarId.Text);
    CarController carController = new CarController();//DELETE QUERY
    carController.DeleteCar(car);
}
else if (btnSave.Text == "Order")
{
    car.CarId = Convert.ToInt32(txtCarId.Text);
    CarController carController = new CarController();//ORDER QUERY
    carController.OrderCar(car);
}
```

```
1 reference
private bool ValidateCarForm()
{
    ErrorProvider errorProvider = new ErrorProvider();
    errorProvider.Clear();

    // Validate Car Type
    if (string.IsNullOrWhiteSpace(txtType.Text))
    {
        errorProvider.SetError(txtType, "Car type is required.");
        txtType.Focus();
        return false;
    }

    // Validate Car Brand
    if (string.IsNullOrWhiteSpace(txtBrand.Text))
    {
        errorProvider.SetError(txtBrand, "Car brand is required.");
        txtBrand.Focus();
        return false;
    }

    // Validate Car Color
    CarColor selectedColor = (CarColor)ddlColors.SelectedItem;
    if (selectedColor == null || !selectedColor.ColorId.HasValue)
    {
        errorProvider.SetError(ddlColors, "Please select a color.");
        ddlColors.Focus();
        return false;
    }
```

```
    // Validate Fuel Economy
    if (string.IsNullOrWhiteSpace(txtFuelEco.Text))
    {
        errorProvider.SetError(txtFuelEco, "Valid fuel economy is required.");
        txtFuelEco.Focus();
        return false;
    }
    // Validate Engine Power
    if (string.IsNullOrWhiteSpace(txtEnginePw.Text))
    {
        errorProvider.SetError(txtEnginePw, "Valid engine power is required.");
        txtEnginePw.Focus();
        return false;
    }
    // Validate Manufacture Date
    if (dateManuDate.Value > DateTime.Now)
    {
        errorProvider.SetError(dateManuDate, "Invalid Date, Manufacture date cannot be in the future.");
        dateManuDate.Focus();
        return false;
    }
    // Validate Price
    if (string.IsNullOrWhiteSpace(txtPrice.Text))
    {
        errorProvider.SetError(txtPrice, "Valid price is required.");
        txtPrice.Focus();
        return false;
    }
    // Validate Image
    if (picCar.Image == null)
    {
        errorProvider.SetError(picCar, "Car picture is required.");
        picCar.Focus();
        return false;
    }
    return true;
```

```
1 reference
private void btnBack_Click(object sender, EventArgs e)
{   //GO BACK TO TABLE
    CarsUI carsUI = new CarsUI();
    carsUI.Show();
    this.Hide();
}
```

Controller

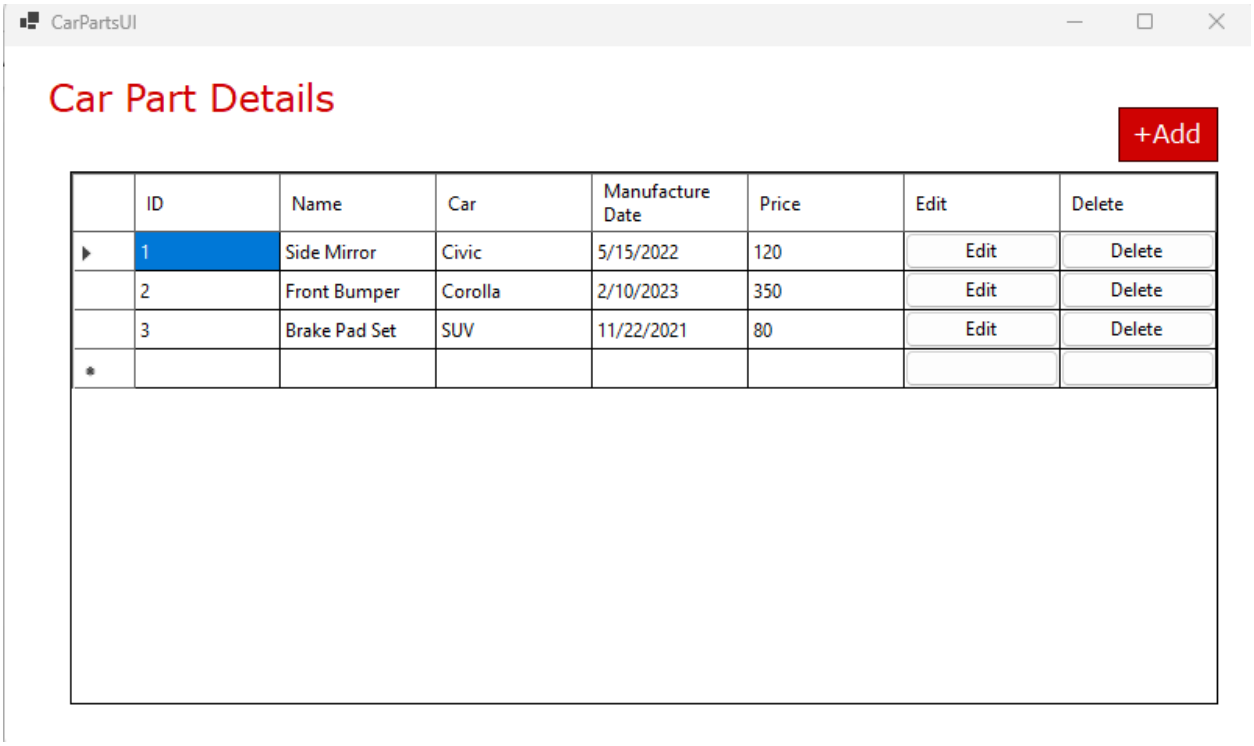
```
public Car GetCarByID(int carId)
{//GET CAR DATA BY ID TO FILL THE CAR FORM WHEN IN EDIT, DELETE OR ORDER MODE
    string selectCarById = "Select car_id, car_type, car_brand, car_color, car_fueleconomy, car_engine_power, car_manufacture_date, car_price, " +
        "car_picture from Car where car_id = " + carId;
    Car car = new Car();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarById, con))
        {
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    car.CarId = Convert.ToInt32(reader["car_id"]);
                    car.Type = reader["car_type"].ToString();
                    car.Brand = reader["car_brand"].ToString();
                    car.ColorId = Convert.ToInt32(reader["car_color"]);
                    car.FuelEconomy = Convert.ToDecimal(reader["car_fueleconomy"]);
                    car.EnginePower = Convert.ToInt32(reader["car_engine_power"]);
                    car.ManufactureDate = Convert.ToDateTime(reader["car_manufacture_date"]);
                    car.Price = Convert.ToDecimal(reader["car_price"]);
                    byte[] carPictureBytes = (byte[])reader["car_picture"];
                    car.Picture = carPictureBytes;
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
```

```
public void UpdateCar(Car car)
{
    try
    {
        // Define the SQL UPDATE query using parameters
        string updateCar = "UPDATE Car SET car_type = @carType, car_brand = @carBrand, car_color = @carColor, " +
                           "car_fueleconomy = @carFuelEconomy, car_engine_power = @carEnginePower, " +
                           "car_manufacture_date = @carManufactureDate, car_price = @carPrice, " +
                           "car_picture = @carPicture, " +
                           "entry_user = @entryUser, entry_date = @entryDate WHERE car_id = @carId";

        using (SqlCommand cmd = new SqlCommand(updateCar, con))
        {
            // Add parameters to the command to prevent SQL injection and handle types correctly
            cmd.Parameters.AddWithValue("@carType", car.Type);
            cmd.Parameters.AddWithValue("@carBrand", car.Brand);
            cmd.Parameters.AddWithValue("@carColor", car.ColorId);
            cmd.Parameters.AddWithValue("@carFuelEconomy", car.FuelEconomy);
            cmd.Parameters.AddWithValue("@carEnginePower", car.EnginePower);
            cmd.Parameters.AddWithValue("@carManufactureDate", car.ManufactureDate);
            cmd.Parameters.AddWithValue("@carPrice", car.Price);
            cmd.Parameters.AddWithValue("@carPicture", car.Picture);
            cmd.Parameters.AddWithValue("@entryUser", car.EntryUser);
            cmd.Parameters.AddWithValue("@entryDate", car.EntryDate);
            cmd.Parameters.AddWithValue("@carId", car.CarId); // Use car ID to identify the record to update

            con.Open(); // Open the connection

            cmd.ExecuteNonQuery(); // Execute the UPDATE query
        }
        MessageBox.Show("Update Successful");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

```
1 reference
public void DeleteCar(Car car)
{//DELETE QUERY FOR CAR
    con.Open();
    try
    {
        string deleteCar = "Delete from Car where car_id = " + car.CarId;
        using (SqlCommand cmd = new SqlCommand(deleteCar, con))
        {
            cmd.ExecuteNonQuery();
        }
        MessageBox.Show("Delete Successful");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

```
1 reference
public void OrderCar(Car car)
{//INSERT ORDER
    con.Open();
    try
    {
        string orderCar = "Insert into Orders (user_id, order_date, order_status) VALUES (" + car.EntryUser + ", '" + car.EntryDate + "', 'Placed');" +
            "select CAST(scope_identity() as int)";
        SqlCommand cmd = new SqlCommand(orderCar, con);
        int order_id = (int)cmd.ExecuteScalar(); //Get order_id";

        string orderItem = "Insert into OrderProduct (order_id, car_id) VALUES (" + order_id + ", " + car.CarId + ")";
        SqlCommand cmdorder = new SqlCommand(orderItem, con);
        cmdorder.ExecuteNonQuery();

        MessageBox.Show("Order Successfull");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

Admin Managing Car Parts Details



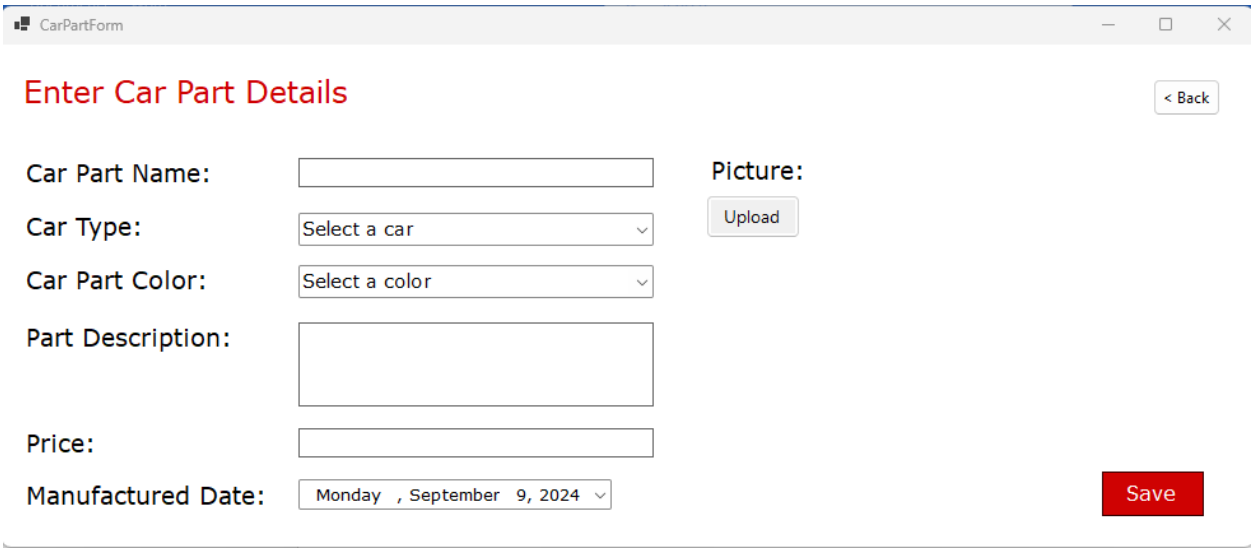
```
1 reference
private void CarPartsUI_Load(object sender, EventArgs e)
{
    GetCarPartsData(); //GET CAR PARTS FROM DB
}

1 reference
private void GetCarPartsData()
{
    CarPartController carPartController = new CarPartController();
    grdCarParts.DataSource = carPartController.GetCarParts();
    grdCarParts.Columns["carpart_id"].HeaderText = "ID";
    grdCarParts.Columns["carpart_name"].HeaderText = "Name";
    grdCarParts.Columns["car_type"].HeaderText = "Car";  // This is from the Car table for carpart_car_id
    grdCarParts.Columns["carpart_manufacture_date"].HeaderText = "Manufacture Date";
    grdCarParts.Columns["carpart_price"].HeaderText = "Price";
    if (GlobalSession.RoleId == 2) //IF ADMIN
    {
        DataGridViewButtonColumn editButtonColumn = new DataGridViewButtonColumn
        {
            HeaderText = "Edit", //SHOW EDIT BUTTON
            Text = "Edit",
            UseColumnTextForButtonValue = true,
            Name = "EditColumn"
        };
        grdCarParts.Columns.Add(editButtonColumn);
        DataGridViewButtonColumn deleteButtonColumn = new DataGridViewButtonColumn
        {
            HeaderText = "Delete", //SHOW DELETE BUTTON
            Text = "Delete",
            UseColumnTextForButtonValue = true,
            Name = "DeleteColumn"
        };
        grdCarParts.Columns.Add(deleteButtonColumn);
    }
    grdCarParts.CellClick += grdCarParts_CellClick;
```

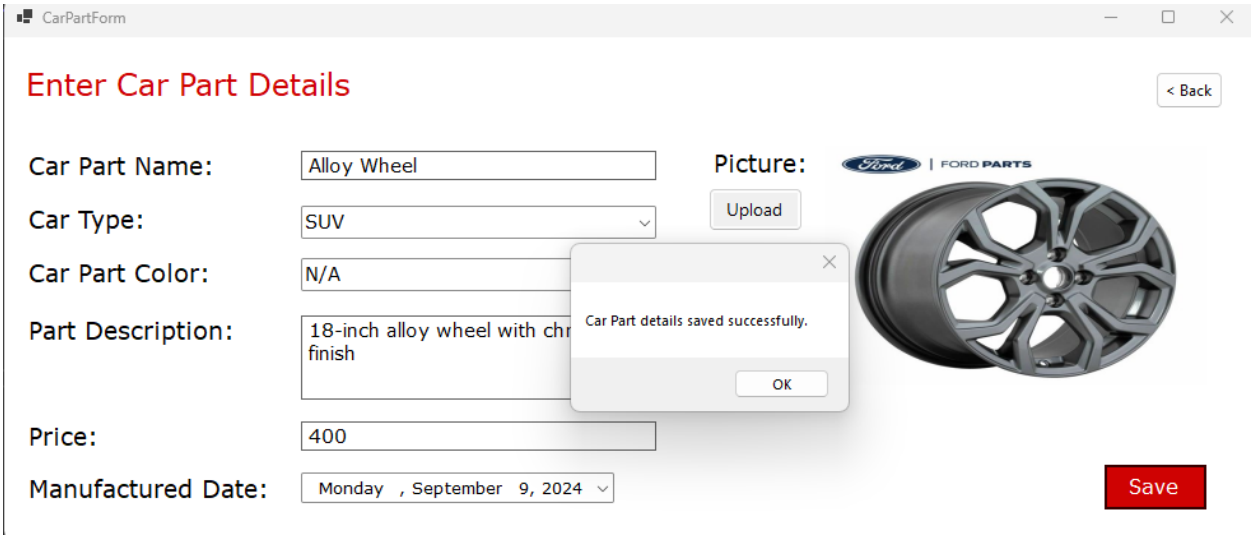
```
1 reference
private void grdCarParts_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0) // Ensure the user clicked on a valid row
    {
        int carPartId = Convert.ToInt32(grdCarParts.Rows[e.RowIndex].Cells["carpart_id"].Value);

        // Determine which column was clicked (Edit, Delete, or Order)
        if (grdCarParts.Columns[e.ColumnIndex].Name == "EditColumn")
        {
            OpenCarPartForm(carPartId, "U");  // Open in Edit Mode
        }
        else if (grdCarParts.Columns[e.ColumnIndex].Name == "DeleteColumn")
        {
            OpenCarPartForm(carPartId, "D");  // Open in Delete Mode
        }
    }
}

2 references
private void OpenCarPartForm(int carPartId, string mode)
{
    // Open CarPartForm in the specified mode (D: Delete, U: Update)
    CarPartForm carPartForm = new CarPartForm();
    carPartForm.SetMode(carPartId, mode);
    carPartForm.Show();
}
1 reference
private void btnAdd_Click(object sender, EventArgs e)
{   //GO TO CAR PART FORM
    CarPartForm carPartForm = new CarPartForm();
    carPartForm.Show();
    this.Hide();
}
```

Controller

```
using Microsoft.Data.SqlClient;

namespace ABCCarTraders.Controller
{
    12 references
    public class CarPartController
    {
        SqlConnection con = new SqlConnection("Data Source=DESKTOP-I8O0T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;Integrated Security=True;Trust Server Certificate=True");

        1 reference
        public DataTable GetCarParts() //GET CARPARTS FOR TABLE
        {
            string selectCarParts = "SELECT cp.carpart_id, cp.carpart_name, c.car_type, cp.carpart_manufacture_date, cp.carpart_price FROM CarPart cp " +
                "JOIN Car c ON cp.carpart_car_id = c.car_id";
            DataTable dtCarParts = new DataTable();
            try
            {
                con.Open();
                using (SqlCommand cmd = new SqlCommand(selectCarParts, con))
                {
                    using (SqlDataAdapter da = new SqlDataAdapter(cmd))
                    {
                        da.Fill(dtCarParts); // Fills the DataTable with the result from the database
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }
            finally
            {
                con.Close();
            }
            return dtCarParts;
        }
```

When Clicked on Add, the CarPartForm is displayed:





When Edit or Delete button is clicked, Form is Opened in Edit or Delete Mode. Hence, if its opened in Edit mode, the Update Query is carried out otherwise the Delete Query.

CarPartForm

Enter Car Part Details

< Back

Car Part Name: Side Mirror

Car Type: Civic

Car Part Color: Black

Part Description: Driver side mirror with au
dimming.

Price: 120

Manufactured Date: Sunday , May 15, 2022

Picture:

Upload

Update Successful

OK

Update

CarPartForm

Enter Car Part Details

< Back

Car Part Name: Brake Pad Set

Car Type: SUV

Car Part Color: N/A

Part Description: Front and rear brake pad set

Price: 80

Manufactured Date: Monday , November 22, 2021

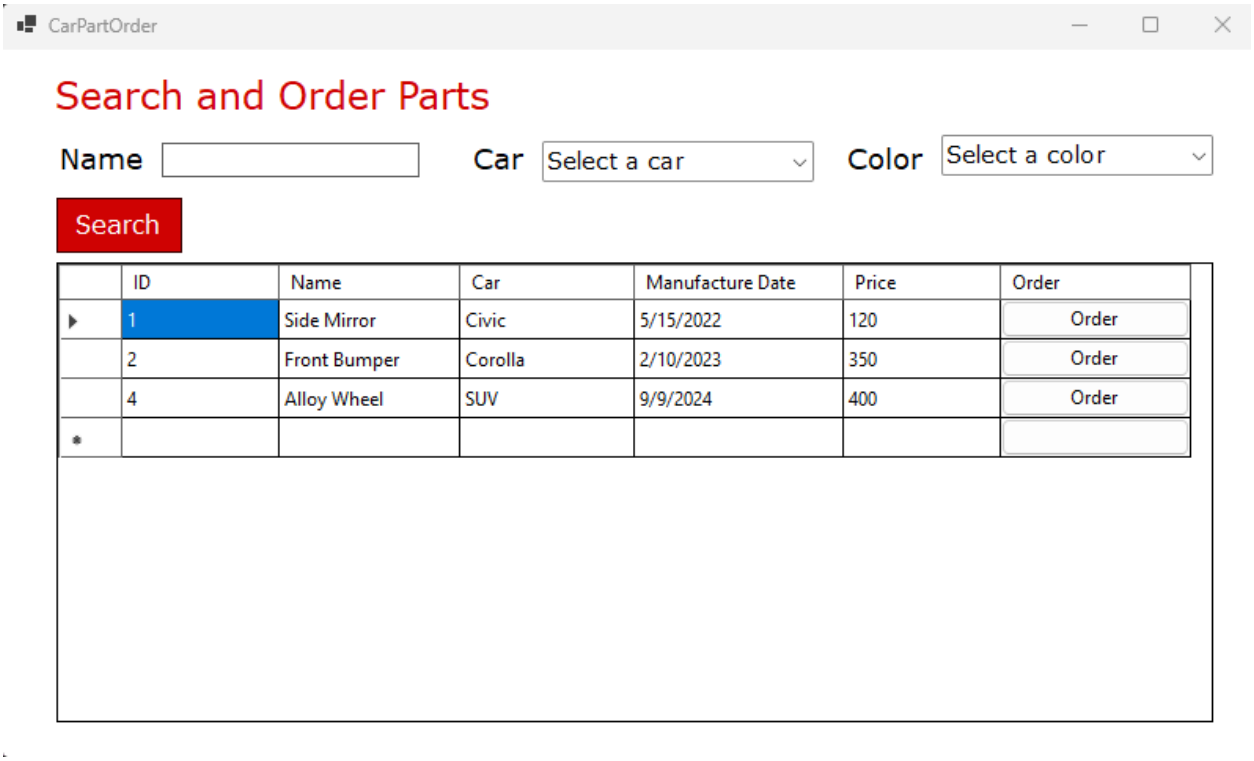
Picture:

Delete Successful

OK

Delete

Customer Ordering Car Parts



```
1 reference
private void LoadColorsIntoComboBox()
{
    try
    {
        con.Open();

        // Query to fetch color_id and color_name from the database
        string query = "SELECT color_id, color_name FROM CarColor";

        SqlCommand cmd = new SqlCommand(query, con);

        // Execute the command and read the results
        using (SqlDataReader reader = cmd.ExecuteReader())
        {
            // Create a list to hold color items
            List<CarColor> colors = new List<CarColor>();

            // Add "Select a color" as default
            colors.Add(new CarColor { ColorId = null, ColorName = "Select a color" });

            // Populate the list with colors from the database
            while (reader.Read())
            {
                colors.Add(new CarColor
                {
                    ColorId = (int)reader["color_id"],
                    ColorName = reader["color_name"].ToString()
                });
            }

            // Bind the list to the ComboBox
            ddlColors.DataSource = colors;
            ddlColors.DisplayMember = "ColorName";  // Show the color name
            ddlColors.ValueMember = "ColorId";      // Use color_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

```
private void LoadCarsIntoComboBox()
{
    try
    {
        con.Open();
        string query = "SELECT car_id, car_type FROM Car"; // Query to fetch car_id and car_type from the database

        SqlCommand cmd = new SqlCommand(query, con);
        using (SqlDataReader reader = cmd.ExecuteReader()) // Execute the command and read the results
        {
            List<Car> cars = new List<Car>();  // Create a list to hold car items

            // Add "Select a car" as default
            cars.Add(new Car { CarId = -1, Type = "Select a car" });

            // Populate the list with colors from the database
            while (reader.Read())
            {
                cars.Add(new Car
                {
                    CarId = (int)reader["car_id"],
                    Type = reader["car_type"].ToString()
                });
            }

            // Bind the list to the ComboBox
            ddlCarType.DataSource = cars;
            ddlCarType.DisplayMember = "Type";  // Show the car type
            ddlCarType.ValueMember = "CarId";      // Use car_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

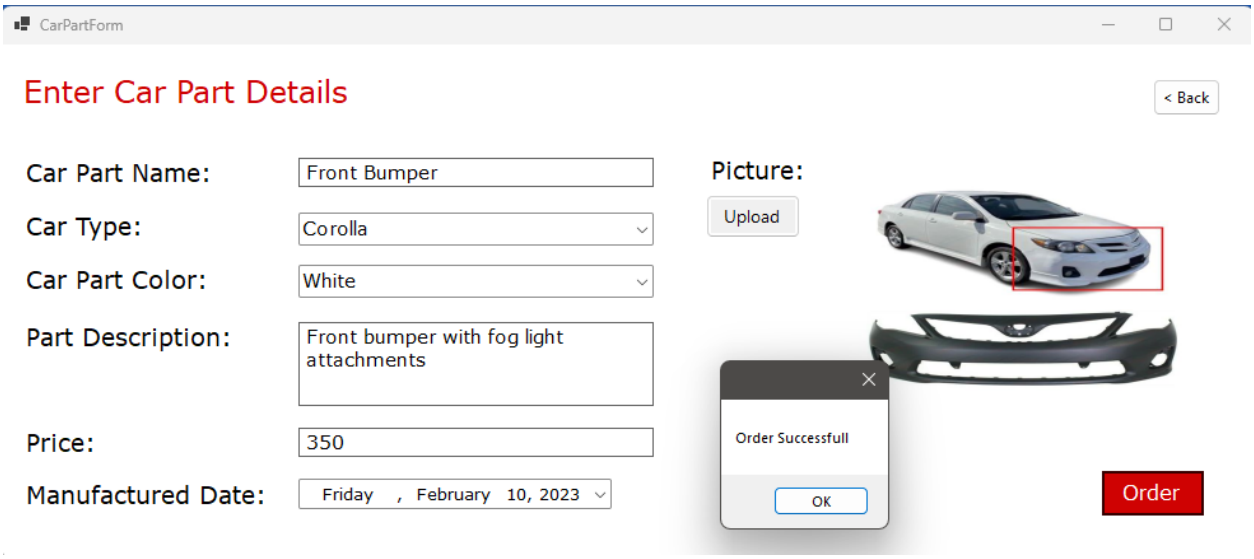
```
1 reference
private void btnSearch_Click(object sender, EventArgs e)
{   //GET VALUES FROM INPUT FIELDS
    string name = txtName.Text;
    int carType = Convert.ToInt32(ddlCarType.SelectedValue);
    int color = Convert.ToInt32(ddlColors.SelectedValue);
    //SEARCH QUERY
    CarPartController carPartController = new CarPartController();
    grdCarParts.DataSource = carPartController.SearchCarPart(name, carType, color);
    //FILL TABLE WITH DATA FROM DB
    grdCarParts.Columns["carpart_id"].HeaderText = "ID";
    grdCarParts.Columns["carpart_name"].HeaderText = "Name";
    grdCarParts.Columns["car_type"].HeaderText = "Car";  // This is from the Car table for carpart_car_id
    grdCarParts.Columns["carpart_manufacture_date"].HeaderText = "Manufacture Date";
    grdCarParts.Columns["carpart_price"].HeaderText = "Price";

    if (GlobalSession.RoleId == 1) //IF USER IS A CUSTOMER
    {
        DataGridViewButtonColumn orderButtonColumn = new DataGridViewButtonColumn
        {   //SHOW ORDER BUTTON
            HeaderText = "Order",
            Text = "Order",
            UseColumnTextForButtonValue = true,
            Name = "OrderColumn"
        };
        grdCarParts.Columns.Add(orderButtonColumn);
    }
    grdCarParts.CellClick += grdCarParts_CellClick;
}
```

```
1 reference
private void grdCarParts_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (e.RowIndex >= 0) // Ensure the user clicked on a valid row
    {
        int carPartsId = Convert.ToInt32(grdCarParts.Rows[e.RowIndex].Cells["carpart_id"].Value);

        if (grdCarParts.Columns[e.ColumnIndex].Name == "OrderColumn")
        {
            OpenCarPartForm(carPartsId, "O");  // Open in Order Mode
        }
    }
}

1 reference
private void OpenCarPartForm(int carId, string mode)
{
    // Open CarPartForm in the specified mode (O: Order)
    CarPartForm carPartForm = new CarPartForm();
    carPartForm.SetMode(carId, mode);
    carPartForm.Show();
}
```

Controller

```
1 reference
public DataTable SearchCarPart(string name, int carType, int color)
{
    con.Open();
    string searchQuery = "SELECT cp.carpart_id, cp.carpart_name, c.car_type, cp.carpart_manufacture_date, cp.carpart_price FROM CarPart cp " +
        "JOIN Car c ON cp.carpart_car_id = c.car_id WHERE (cp.carpart_name = '" + name + "' OR '' = '" + name + "') " +
        "AND (cp.carpart_car_id =" + carType + " OR -1 = " + carType + ") AND (cp.carpart_color_id = " + color + " OR 0 = " + color + ")";

    DataTable dtCarParts = new DataTable();
    try
    {
        using (SqlCommand cmd = new SqlCommand(searchQuery, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dtCarParts); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dtCarParts;
}
```

When the Order button is clicked the Car Part Form will be opened in Order Mode.



Car Part Form Code

```
SqlConnection con = new SqlConnection("Data Source=DESKTOP-I800T49\\SQLEXPRESS;Initial Catalog=ABCCarTradersDB;Integrated Security=True;Trust Server Certificate=True");
//INITIALIZE VARIABLES FOR MODES EDIT, DELETE AND ORDER
public int carPartId;
public string mode;
public byte[] CarPartPicture;

3 references
public CarPartForm()
{
    InitializeComponent();
}

1 reference
private void CarPartForm_Load(object sender, EventArgs e)
{
    LoadColorsIntoComboBox(); //LOAD COLORS TO COMBO BOX
    LoadCarsIntoComboBox(); //LOAD CARS TO COMBO BOX
}

2 references
public void SetMode(int carPartId, string mode)
{
    this.carPartId = carPartId;
    this.mode = mode;
    LoadCarPartData(); // Load data based on the carId
    if (mode == "U")
    {
        btnSave.Text = "Update"; //UPDATE MODE
    }
    else if (mode == "D")
    {
        btnSave.Text = "Delete"; //DELETE MODE
    }
    else if (mode == "O")
    {
        btnSave.Text = "Order"; //ORDER MODE
    }
```

```
private void LoadCarsIntoComboBox()
{
    try
    {
        con.Open();

        string query = "SELECT car_id, car_type FROM Car"; // Query to fetch car_id and car_type from the database
        SqlCommand cmd = new SqlCommand(query, con);

        using (SqlDataReader reader = cmd.ExecuteReader()) // Execute the command and read the results
        {
            List<Car> cars = new List<Car>(); // Create a list to hold car items

            // Add "Select a car" as default
            cars.Add(new Car { CarId = -1, Type = "Select a car" });

            // Populate the list with colors from the database
            while (reader.Read())
            {
                cars.Add(new Car
                {
                    CarId = (int)reader["car_id"],
                    Type = reader["car_type"].ToString()
                });
            }
            // Bind the list to the ComboBox
            ddlCarType.DataSource = cars;
            ddlCarType.DisplayMember = "Type";  // Show the car type
            ddlCarType.ValueMember = "CarId";       // Use car_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

```
private void LoadColorsIntoComboBox()
{
    try
    {
        con.Open();
        string query = "SELECT color_id, color_name FROM CarColor"; // Query to fetch color_id and color_name from the database

        SqlCommand cmd = new SqlCommand(query, con);
        using (SqlDataReader reader = cmd.ExecuteReader()) // Execute the command and read the results
        {
            // Create a list to hold color items
            List<CarColor> colors = new List<CarColor>();
            colors.Add(new CarColor { ColorId = null, ColorName = "Select a color" }); // Add "Select a color" as default

            // Populate the list with colors from the database
            while (reader.Read())
            {
                colors.Add(new CarColor
                {
                    ColorId = (int)reader["color_id"],
                    ColorName = reader["color_name"].ToString()
                });
            }

            // Bind the list to the ComboBox
            ddlColors.DataSource = colors;
            ddlColors.DisplayMember = "ColorName";  // Show the color name
            ddlColors.ValueMember = "ColorId";      // Use color_id as the value
        }
        con.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}
```

```
1 reference
private void btnUpload_Click(object sender, EventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();

    // Set the filter to allow only image files
    openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";

    // Show the file dialog and check if the user selected a file
    if (openFileDialog.ShowDialog() == DialogResult.OK)
    {
        // Get the file path
        string filePath = openFileDialog.FileName;

        // Display the image in the PictureBox for preview
        picCarPart.Image = Image.FromFile(filePath);
        picCarPart.SizeMode = PictureBoxSizeMode.StretchImage;
    }

    CarPartPicture = GetImageAsByteArray(picCarPart.Image);
}

1 reference
private byte[] GetImageAsByteArray(Image image)
{   // GET IMAGE AS BYTE ARRAY
    using (MemoryStream ms = new MemoryStream())
    {
        image.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
        return ms.ToArray();
    }
}
```

```
private void LoadCarPartData() //GET DATA FOR EDIT, DELETE, ORDER MODES
{
    //Set Deatils to fields
    CarPartController carPartController = new CarPartController();
    CarPart carPart = carPartController.GetCarPartById(carPartId);

    txtCarPartId.Text = carPart.CarPartId.ToString();
    txtName.Text = carPart.Name.ToString();
    ddlCarType.SelectedValue = carPart.CarId;
    dateManuDate.Value = carPart.ManufactureDate;
    txtPrice.Text = carPart.Price.ToString();
    txtDesc.Text = carPart.Description.ToString();
    ddlColors.ValueMember = carPart.ColorId.ToString();
    CarPartPicture = carPart.Picture;

    // Ensure carPart.Picture is not null or empty
    if (carPart.Picture != null && carPart.Picture.Length > 0)
    {
        using (MemoryStream ms = new MemoryStream(carPart.Picture))
        {
            try
            {
                Image carPartImage = Image.FromStream(ms);
                picCarPart.SizeMode = PictureBoxSizeMode.StretchImage; // Set the size mode
                picCarPart.Image = carPartImage; // Display in PictureBox
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error loading image: " + ex.Message);
            }
        }
    }
    else
    {
        picCarPart.Image = null; // Clear the PictureBox if no image is available
    }
}
```

```
1 reference
private void btnSave_Click(object sender, EventArgs e)
{
    if (ValidateCarPartForm()) //CHECK VALIDATION
    {   //SET DATA TO CARPART CLASS
        CarPart carPart = new CarPart();
        carPart.Name = txtName.Text;
        Car selectedCar = (Car)ddlCarType.SelectedItem;
        carPart.CarId = selectedCar.CarId;
        carPart.ManufactureDate = Convert.ToDateTime(dateManuDate.Value.ToString("yyyy-MM-dd"));
        carPart.Price = Convert.ToDecimal(txtPrice.Text);
        carPart.Description = txtDesc.Text;
        CarColor selectedColor = (CarColor)ddlColors.SelectedItem;
        carPart.ColorId = selectedColor.ColorId.Value;

        carPart.EntryDate = DateTime.Now.ToString("yyyy-MM-dd");
        carPart.EntryUser = GlobalSession.UserId;

        carPart.Picture = CarPartPicture;

        if (btnSave.Text == "Save")
        {
            try
            {   //INSERT QUERY
                con.Open();
                string insertCar = "INSERT INTO CarPart (carpart_name, carpart_car_id, carpart_manufacture_date, carpart_price, carpart_description, " +
                    "carpart_color_id, carpart_picture, entry_user, entry_date) " +
                        "VALUES (@carPartName, @carId, @carPartManufactureDate, @carPartPrice, @carPartDesc, @carPartColor, @carPartPicture, @entryUser, " +
                        "@entryDate)";
```

```
                using (SqlCommand cmd = new SqlCommand(insertCar, con))
                {
                    // Add parameters to the command
                    cmd.Parameters.AddWithValue("@carPartName", carPart.Name);
                    cmd.Parameters.AddWithValue("@carId", carPart.CarId);
                    cmd.Parameters.AddWithValue("@carPartManufactureDate", carPart.ManufactureDate);
                    cmd.Parameters.AddWithValue("@carPartPrice", carPart.Price);
                    cmd.Parameters.AddWithValue("@carPartDesc", carPart.Description);
                    cmd.Parameters.AddWithValue("@carPartColor", carPart.ColorId);

                    // Use byte array for the image data
                    cmd.Parameters.AddWithValue("@carPartPicture", CarPartPicture ?? (object)DBNull.Value);

                    cmd.Parameters.AddWithValue("@entryUser", carPart.EntryUser);
                    cmd.Parameters.AddWithValue("@entryDate", carPart.EntryDate);
                    cmd.ExecuteNonQuery();
                }
                con.Close();
                MessageBox.Show("Car Part details saved successfully.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Error: " + ex.Message);
            }

        }
        else if (btnSave.Text == "Update")
        {   //UPDATE QUERY
            carPart.CarPartId = Convert.ToInt32(txtCarPartId.Text);
            CarPartController carPartController = new CarPartController();
            carPartController.UpdateCarParts(carPart);
        }
```

```
        }
        else if (btnSave.Text == "Delete")
        {   //DELETE QUERY
            carPart.CarPartId = Convert.ToInt32(txtCarPartId.Text);
            CarPartController carPartController = new CarPartController();
            carPartController.DeleteCarParts(carPart);
        }
        else if (btnSave.Text == "Order")
        {   //INSERT TO ORDER TABLE
            carPart.CarPartId = Convert.ToInt32(txtCarPartId.Text);
            CarPartController carPartController = new CarPartController();
            carPartController.OrderCarParts(carPart);
        }
    }
}
```

```
1 reference
private bool ValidateCarPartForm()
{
    ErrorProvider errorProvider = new ErrorProvider();
    errorProvider.Clear();

    // Validate CarPart Name
    if (string.IsNullOrWhiteSpace(txtName.Text))
    {
        errorProvider.SetError(txtName, "Car Part Name is required.");
        txtName.Focus();
        return false;
    }

    Car selectedCar = (Car)ddlCarType.SelectedItem;
    if (selectedCar == null)
    {
        errorProvider.SetError(ddlCarType, "Please select a car.");
        ddlCarType.Focus();
        return false;
    }

    if (dateManuDate.Value > DateTime.Now)
    {
        errorProvider.SetError(dateManuDate, "Invalid Date, Manufacture date cannot be in the future.");
        dateManuDate.Focus();
        return false;
    }

    if (string.IsNullOrWhiteSpace(txtPrice.Text))
    {
        errorProvider.SetError(txtPrice, "Valid price is required.");
        txtPrice.Focus();
        return false;
    }
```

```
        if (string.IsNullOrWhiteSpace(txtDesc.Text))
        {
            errorProvider.SetError(txtDesc, "Description is required.");
            txtDesc.Focus();
            return false;
        }

        CarColor selectedColor = (CarColor)ddlColors.SelectedItem;
        if (selectedColor == null || !selectedColor.ColorId.HasValue)
        {
            errorProvider.SetError(ddlColors, "Please select a color.");
            ddlColors.Focus();
            return false;
        }

        if (picCarPart.Image == null)
        {
            errorProvider.SetError(picCarPart, "CarPart picture is required.");
            picCarPart.Focus();
            return false;
        }

        return true;
    }

    1 reference
    private void btnBack_Click(object sender, EventArgs e)
    {
        CarPartsUI carPartsUI = new CarPartsUI();
        carPartsUI.Show();
        this.Hide();
    }
}
```

Controller

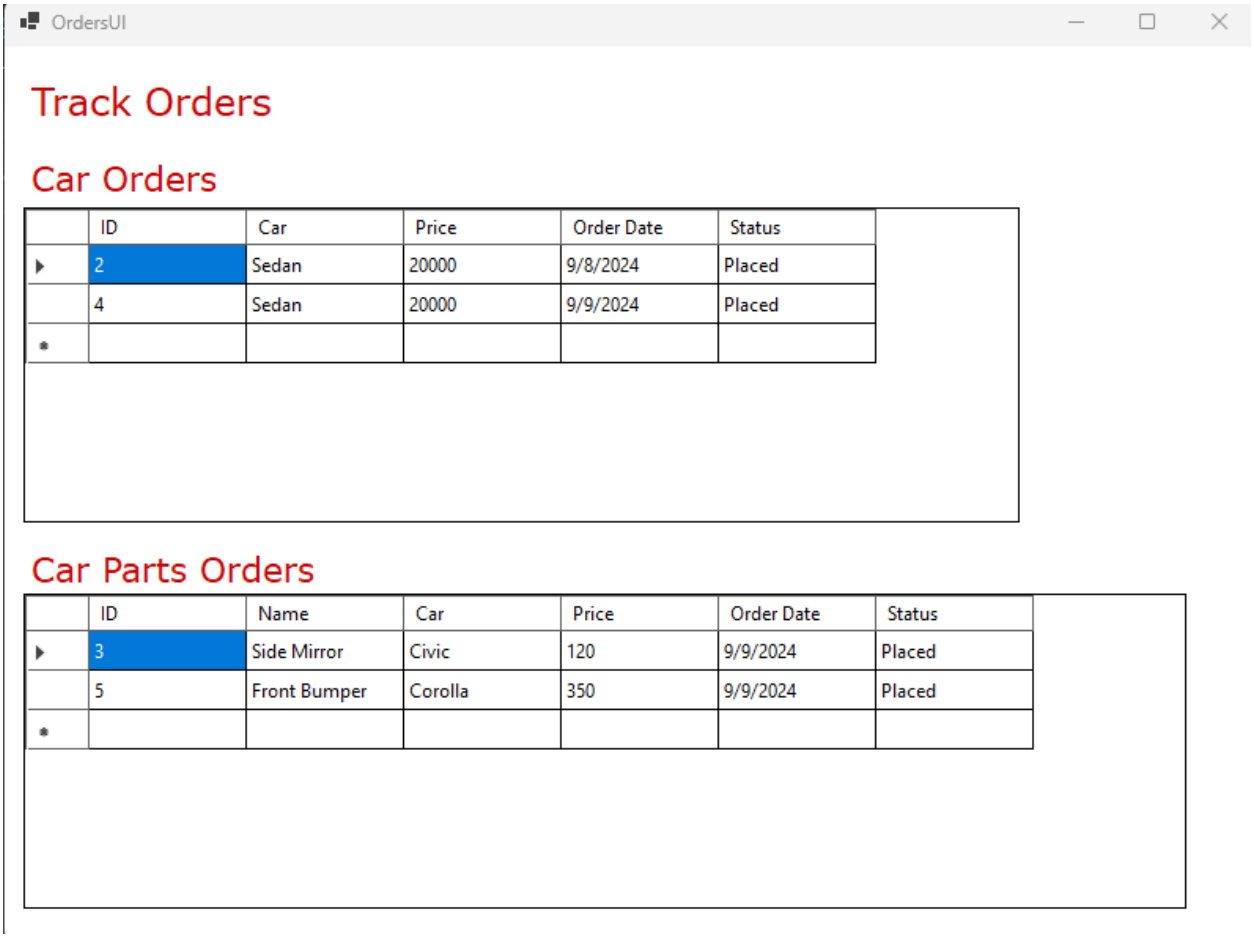
```
1 reference
public CarPart GetCarPartById(int carpartId) //GET DATA TO FILL CARPART FORM IN EDIT, DELETE AND ORDER MODE
{
    string selectCarPartById = "Select carpart_id, carpart_name, carpart_car_id, carpart_manufacture_date, carpart_price, carpart_description, " +
        "carpart_color_id, carpart_picture from CarPart where carpart_id = " + carpartId;
    CarPart carpart = new CarPart();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarPartById, con))
        {
            using (SqlDataReader reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    carpart.CarPartId = Convert.ToInt32(reader["carpart_id"]);
                    carpart.Name = reader["carpart_name"].ToString();
                    carpart.CarId = Convert.ToInt32(reader["carpart_car_id"]);
                    carpart.ManufactureDate = Convert.ToDateTime(reader["carpart_manufacture_date"]);
                    carpart.Price = Convert.ToDecimal(reader["carpart_price"]);
                    carpart.Description = reader["carpart_description"].ToString();
                    carpart.ColorId = Convert.ToInt32(reader["carpart_color_id"]);

                    byte[] carPartPictureBytes = (byte[])reader["carpart_picture"];
                    carpart.Picture = carPartPictureBytes;
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
```

```
1 reference
public void UpdateCarParts(CarPart carPart)
{
    try
    {
        // Define the SQL UPDATE query using parameters
        string updateCarPart = "UPDATE CarPart SET carpart_name = @carPartName, carpart_car_id = @carId, carpart_manufacture_date = @carPartManufactureDate, " +
            "carpart_price = @carPartPrice, carpart_description = @carPartDesc, carpart_color_id = @carPartColor, carpart_picture = @carPartPicture, " +
            "entry_user = @entryUser, entry_date = @entryDate WHERE carpart_id = @carPartId";

        using (SqlCommand cmd = new SqlCommand(updateCarPart, con))
        {
            // Add parameters to the command to prevent SQL injection and handle types correctly
            cmd.Parameters.AddWithValue("@carPartName", carPart.Name);
            cmd.Parameters.AddWithValue("@carId", carPart.CarId);
            cmd.Parameters.AddWithValue("@carPartManufactureDate", carPart.ManufactureDate);
            cmd.Parameters.AddWithValue("@carPartPrice", carPart.Price);
            cmd.Parameters.AddWithValue("@carPartDesc", carPart.Description);
            cmd.Parameters.AddWithValue("@carPartColor", carPart.ColorId);
            cmd.Parameters.AddWithValue("@carPartPicture", carPart.Picture);
            cmd.Parameters.AddWithValue("@entryUser", carPart.EntryUser);
            cmd.Parameters.AddWithValue("@entryDate", carPart.EntryDate);
            cmd.Parameters.AddWithValue("@carPartId", carPart.CarPartId); // Use carPart ID to identify the record to update

            con.Open(); // Open the connection

            // Execute the UPDATE query
            cmd.ExecuteNonQuery();
        }

        MessageBox.Show("Update Successful");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
```

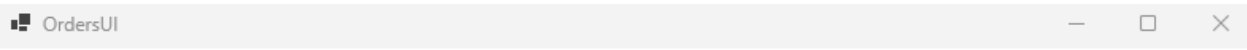
```
public void DeleteCarParts(CarPart carPart)
{
    con.Open();
    try
    {
        string deleteCarPart = "Delete from CarPart where carpart_id = " + carPart.CarPartId;
        using (SqlCommand cmd = new SqlCommand(deleteCarPart, con))
        {
            cmd.ExecuteNonQuery();
        }
        MessageBox.Show("Delete Successful");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
}

1 reference
public void OrderCarParts(CarPart carPart)
{
    con.Open();
    try
    {
        string orderCar = "Insert into Orders (user_id, order_date, order_status) VALUES (" + carPart.EntryUser + ", '" + carPart.EntryDate + "', " +
            "'Placed');" +
            "select CAST(scope_identity() as int)";
        SqlCommand cmd = new SqlCommand(orderCar, con);
        int order_id = (int)cmd.ExecuteScalar(); //Get order_id";
        string orderItem = "Insert into OrderProduct (order_id, car_part_id) VALUES (" + order_id + ", " + carPart.CarPartId + ")";
        SqlCommand cmdorder = new SqlCommand(orderItem, con);
        cmdorder.ExecuteNonQuery();
        MessageBox.Show("Order Successfull");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
```

Track Orders

Customer will be able to see their Orders



Admin will be shown the same table except with an option to edit the Order Status




```
using ABCCarTraders.Controller;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ABCCarTraders
{
    7 references
    public partial class OrdersUI : Form
    {
        3 references
        public OrdersUI()
        {
            InitializeComponent();
        }

        1 reference
        private void OrdersUI_Load(object sender, EventArgs e)
        {
            GetOrderData();
        }
```

```
1 reference
private void GetOrderData()
{
    int roleid = GlobalSession.RoleId;
    int userId = GlobalSession.UserId;
    OrderController orderController = new OrderController();
    if (roleid == 1) //FOR CUSTOMERS
    {
        grdCarOrders.DataSource = orderController.GetCarOrdersById(userId);
        grdCarOrders.Columns["order_id"].HeaderText = "ID";
        grdCarOrders.Columns["car_type"].HeaderText = "Car";  // This is from the Car table for carpart_car_id
        grdCarOrders.Columns["car_price"].HeaderText = "Price";
        grdCarOrders.Columns["order_date"].HeaderText = "Order Date";
        grdCarOrders.Columns["order_status"].HeaderText = "Status";

        grdCarPartOrders.DataSource = orderController.GetCarPartOrdersById(userId);
        grdCarPartOrders.Columns["order_id"].HeaderText = "ID";
        grdCarPartOrders.Columns["carpart_name"].HeaderText = "Name";
        grdCarPartOrders.Columns["car_type"].HeaderText = "Car";
        grdCarPartOrders.Columns["carpart_price"].HeaderText = "Price";
        grdCarPartOrders.Columns["order_date"].HeaderText = "Order Date";
        grdCarPartOrders.Columns["order_status"].HeaderText = "Status";

    }
```

```
    else if (roleid == 2) // FOR ADMINS
    {
        grdCarOrders.DataSource = orderController.GetOrders();
        grdCarOrders.Columns["order_id"].HeaderText = "ID";
        grdCarOrders.Columns["car_type"].HeaderText = "Car";  // This is from the Car table for carpart_car_id
        grdCarOrders.Columns["car_price"].HeaderText = "Price";
        grdCarOrders.Columns["order_date"].HeaderText = "Order Date";
        grdCarOrders.Columns["order_status"].HeaderText = "Status";

        DataGridViewButtonColumn editButtonColumn = new DataGridViewButtonColumn
        {
            HeaderText = "Edit",
            Text = "Edit",
            UseColumnTextForButtonValue = true,
            Name = "EditColumn"
        };
        grdCarOrders.Columns.Add(editButtonColumn);

        grdCarPartOrders.DataSource = orderController.GetCarParts();
        grdCarPartOrders.Columns["order_id"].HeaderText = "ID";
        grdCarPartOrders.Columns["carpart_name"].HeaderText = "Name";
        grdCarPartOrders.Columns["car_type"].HeaderText = "Car";
        grdCarPartOrders.Columns["carpart_price"].HeaderText = "Price";
        grdCarPartOrders.Columns["order_date"].HeaderText = "Order Date";
        grdCarPartOrders.Columns["order_status"].HeaderText = "Status";

        DataGridViewButtonColumn editButtonColumnParts = new DataGridViewButtonColumn
        {
            HeaderText = "Edit",
            Text = "Edit",
            UseColumnTextForButtonValue = true,
            Name = "EditColumn"
        };
        grdCarPartOrders.Columns.Add(editButtonColumnParts);
    }
```

Controller

```
1 reference
public DataTable GetCarOrdersById(int userid) //GET CARS ORDERED BY A PARTICULAR USER
{
    string selectCarOrdersById = "SELECT o.order_id, c.car_type, c.car_price, o.order_date, o.order_status FROM Orders o " +
        "JOIN OrderProduct op ON o.order_id = op.order_id JOIN Car c ON c.car_id = op.car_id WHERE o.user_id = " + userid;
    DataTable dt = new DataTable();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarOrdersById, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dt); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dt;
}
```

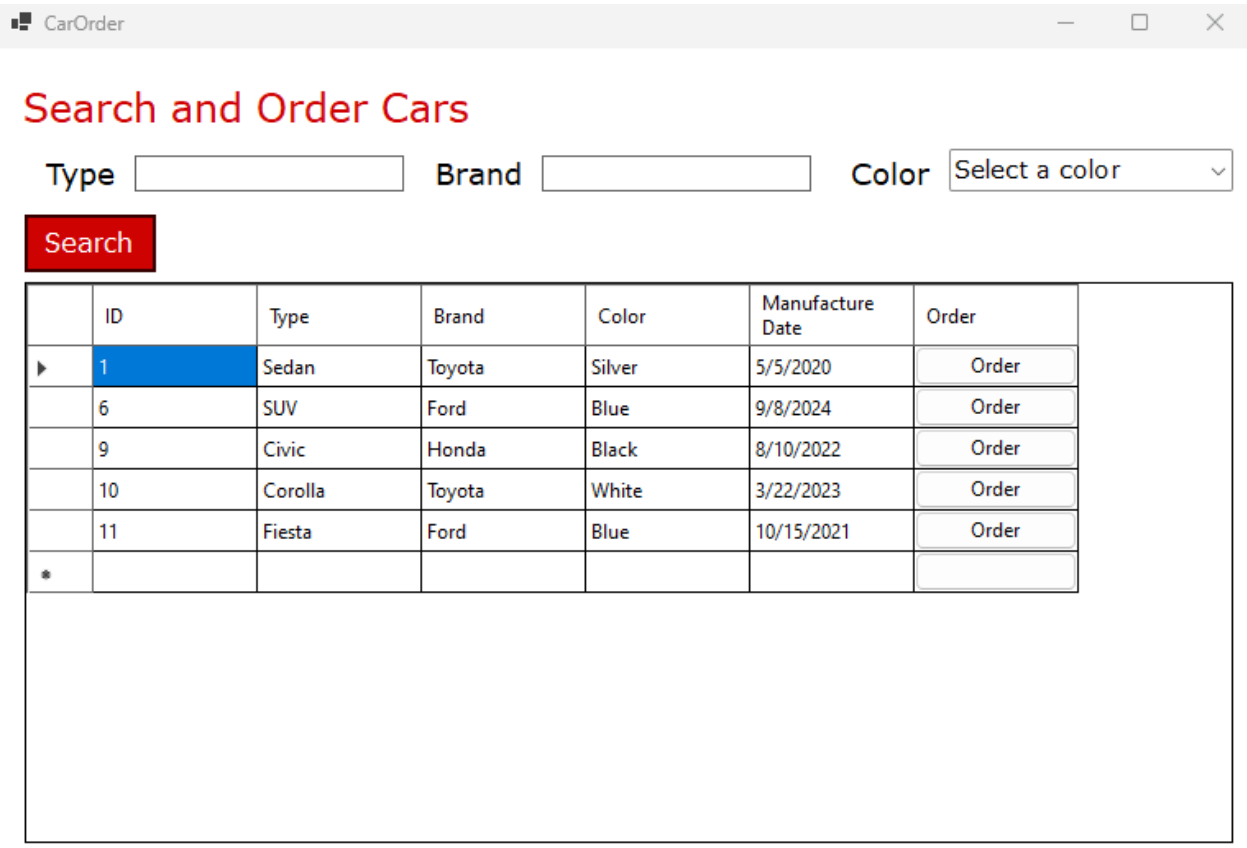
```
1 reference
public DataTable GetCarPartOrdersById(int userId)  //GET CAR PARTS ORDERED BY A PARTICULAR USER
{
    string selectCarPartsOrderById = "SELECT o.order_id, cp.carpart_name, c.car_type, cp.carpart_price, o.order_date, o.order_status FROM Orders o " +
        "JOIN OrderProduct op ON o.order_id = op.order_id JOIN CarPart cp ON op.car_part_id = cp.carpart_id " +
        "JOIN Car c ON cp.carpart_car_id = c.car_id WHERE o.user_id = " + userId;
    DataTable dt = new DataTable();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarPartsOrderById, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dt); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dt;
}
```

```
1 reference
public DataTable GetOrders() //GET ALL CAR ORDERS
{
    string selectCarOrdersById = "SELECT o.order_id, c.car_type, c.car_price, o.order_date, o.order_status FROM Orders o " +
        "JOIN OrderProduct op ON o.order_id = op.order_id JOIN Car c ON c.car_id = op.car_id";
    DataTable dt = new DataTable();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarOrdersById, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dt); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dt;
}
```

```
1 reference
public DataTable GetCarParts() //GET ALL CAR PART ORDERS
{
    string selectCarPartsOrderById = "SELECT o.order_id, cp.carpart_name, c.car_type, cp.carpart_price, o.order_date, o.order_status FROM Orders o " +
        "JOIN OrderProduct op ON o.order_id = op.order_id JOIN CarPart cp ON op.car_part_id = cp.carpart_id " +
        "JOIN Car c ON cp.carpart_car_id = c.car_id";
    DataTable dt = new DataTable();
    try
    {
        con.Open();
        using (SqlCommand cmd = new SqlCommand(selectCarPartsOrderById, con))
        {
            using (SqlDataAdapter da = new SqlDataAdapter(cmd))
            {
                da.Fill(dt); // Fills the DataTable with the result from the database
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: " + ex.Message);
    }
    finally
    {
        con.Close();
    }
    return dt;
}
```

4. Use of Search Algorithms

User Search for Cars based on Criteria




```
public DataTable SearchCar(string type, string brand, int color)
{
    con.Open();
    string searchQuery = "SELECT c.car_id, c.car_type, c.car_brand, cc.color_name, c.car_manufacture_date FROM Car c JOIN CarColor cc " +
        "ON c.car_color = cc.color_id WHERE (c.car_type = '" + type + "' OR '' = '" + type + "') " +
        "AND (c.car_brand = '" + brand + "' OR '' = '" + brand + "') AND (c.car_color = " + color + " OR 0 = " + color + ")";
    DataTable dtCars = new DataTable();
```

The car search functionality allows customers to search for vehicles based on several criteria: car type, car brand, and car color. The SQL query filters the results by these criteria using a JOIN between the Car and CarColor tables, ensuring that both car details and color names are retrieved.

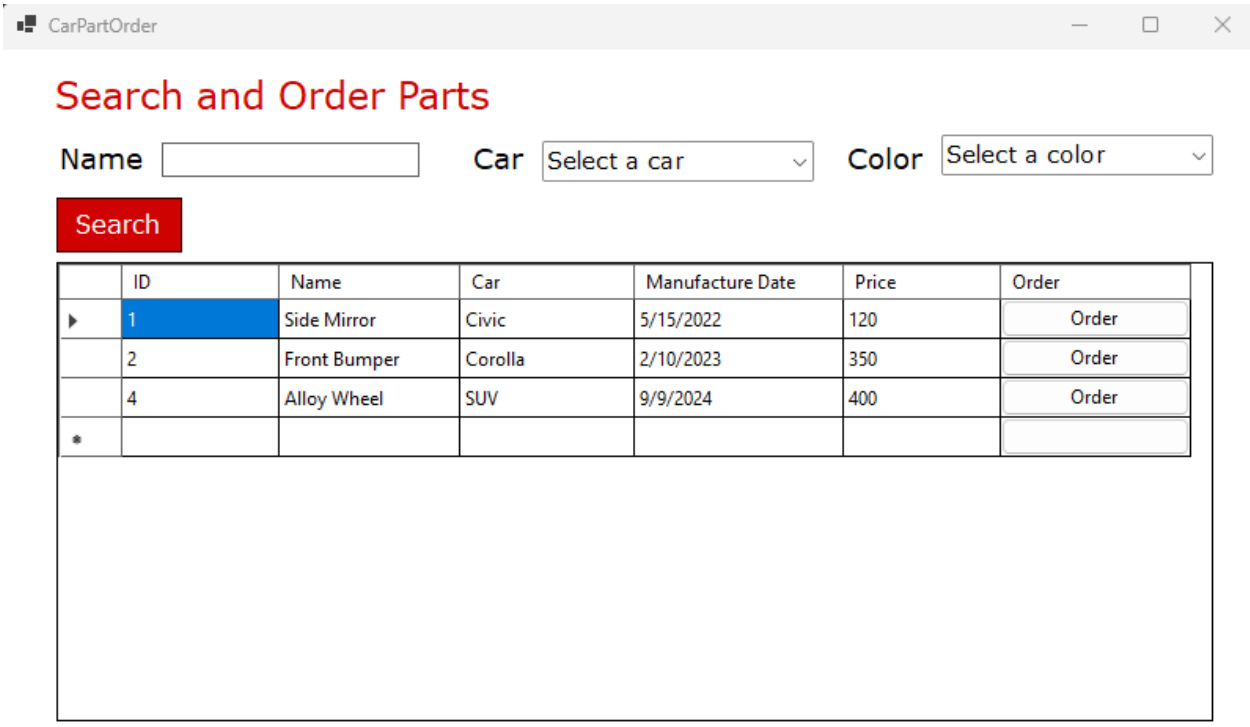
The search logic checks if a specific criterion is provided by the customer, and if it is not, it returns all results for that criterion. For example:

- If the car type is not specified, the query will compare an empty string ('') to return all types.
- If the car brand is not selected, it returns all brands.

- The car color can either be filtered by its ID or, if not selected, it defaults to 0, returning all colors.

This query allows for partial or complete searches while still delivering relevant results.

User Search for Car Parts based on Criteria




```
1 reference
public DataTable SearchCarPart(string name, int carType, int color)
{
    con.Open();
    string searchQuery = "SELECT cp.carpart_id, cp.carpart_name, c.car_type, cp.carpart_manufacture_date, cp.carpart_price FROM CarPart cp " +
        "JOIN Car c ON cp.carpart_car_id = c.car_id WHERE (cp.carpart_name = '" + name + "' OR '' = '" + name + "') " +
        "AND (cp.carpart_car_id =" + carType + " OR -1 = " + carType + ") AND (cp.carpart_color_id = " + color + " OR 0 = " + color + ")";

    DataTable dtCarParts = new DataTable();
```

This SQL query facilitates a search for car parts by allowing customers to filter results based on specific criteria. It retrieves information from the CarPart table and joins it with the Car table to include related car details. The search parameters include the car part's name, the type of car it is associated with, and the color of the car part. If any of these parameters are not specified (e.g., an empty string for name or a default value of -1 for car type), the query omits that criterion from the search, thereby returning a broader set of results. This approach ensures that users can flexibly search for car parts that meet one or more of their criteria.

5. Database

DESKTOP-I8O0T49\S...radersDB - dbo.Car

Column Name	Data Type	Allow Nulls
car_id	int	☐
car_type	varchar(50)	☐
car_brand	varchar(50)	☐
car_color	int	☐
car_fueleconomy	float	☐
car_engine_power	float	☐
car_manufacture_date	date	☐
car_price	float	☐
car_picture	varbinary(MAX)	☐
entry_user	int	☐
entry_date	date	☐

DESKTOP-I8O0T49\S...DB - dbo.CarColor | DESKTOP-I8O0T49\S...rad

Column Name	Data Type	Allow Nulls
color_id	int	☐
color_name	varchar(50)	☐
		☐

DESKTOP-I8O0T49\S...rsDB - dbo.CarPart | DESKTOP-I8O0T49\S...D

Column Name	Data Type	Allow Nulls
carpart_id	int	☐
carpart_name	varchar(50)	☐
carpart_car_id	int	☐
carpart_manufacture_date	date	☐
carpart_price	float	☐
carpart_description	varchar(200)	☐
carpart_color_id	int	☐
carpart_picture	image	☐
entry_user	int	☐
entry_date	date	☐

DESKTOP-I8O0T49\S...ersDB - dbo.Orders | DESKTOP-I8O0T49\S...rs

Column Name	Data Type	Allow Nulls
order_id	int	☐
user_id	int	☐
order_date	date	☐
order_status	varchar(50)	☐

DESKTOP-I8O0T49\S...dbo.OrderProduct | DESKTOP-I8O0T49\S...er

Column Name	Data Type	Allow Nulls
order_id	int	☐
car_id	int	☑
car_part_id	int	☑

DESKTOP-I8O0T49\S...B - dbo.UserLogin | DESKTOP-I8O0T49\S...db

Column Name	Data Type	Allow Nulls
login_id	int	☐
login_username	varchar(50)	☐
login_password	varchar(50)	☐
login_user_id	int	☐
login_role_id	int	☐

DESKTOP-I8O0T49\S...DB - dbo.UserRole | SQLQuery34.sql -...8O0T

Column Name	Data Type	Allow Nulls
user_role_id	int	☐
user_role_description	varchar(50)	☐

DESKTOP-I8O0T49\S...dersDB - dbo.Users | DESKTOP-I8O0T49\S...D

Column Name	Data Type	Allow Nulls
user_id	int	☐
user_firstname	varchar(50)	☑
user_lastname	varchar(50)	☑
user_fullname	varchar(50)	☐
user_email	varchar(50)	☐
user_contactno	varchar(10)	☑
user_address	varchar(50)	☑

6. Reflection on Development Process

Throughout the development process, I encountered some challenges, primarily with handling images and configuring forms to open in specific modes. Despite these difficulties, the overall experience was positive. The database integration was smooth, and the class structures were effective in managing the application's functionality. The solutions I implemented for image handling and form management enhanced my problem-solving skills and contributed to a deeper understanding of C# and database interactions. Overall, the project was a valuable learning experience, and I am pleased with the successful outcomes.