Building an Automobile Management Application using WPF application and Entity Framework Core

Introduction

Imagine you're an employee of a car retailer named **Automobile Store**. Your manager has asked you to develop a WPF application for automobile management (CarID, CarName, Manufacturer, Price, and ReleasedYear). The application has to support adding, viewing, modifying, and removing products—a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

This lab explores creating an application using WPF with .NET Core, and C#. An **SQL Server Database** will be created to persist the car's data that will be used for reading and managing automobile data by **Entity Framework Core**

Lab Objectives

In this lab, you will:

- Use the Visual Studio.NET to create WPF application and Class Library (.dll) project.
- Create a SQL Server database named MyStock that has a Cars table.
- Develop a DataProvider class to perform CRUD actions using Entity Framework Core.
- Apply Dependency injection (DI) in WPF application.
- Apply Repository pattern and Singleton pattern in a project.
- Add CRUD action methods to WPF application.
- Run the project and test the WPF application actions.

MyStock Database

T470S.MyStock - dbo.Cars

Column Name	Data Type	Allow Nulls
CarID	int	☐
CarName	varchar(50)	☐
Manufacturer	varchar(50)	☐
Price	money	☐
ReleasedYear	int	☐
		☐

Column Properties

Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	No

T470S.MyStock - dbo.Cars

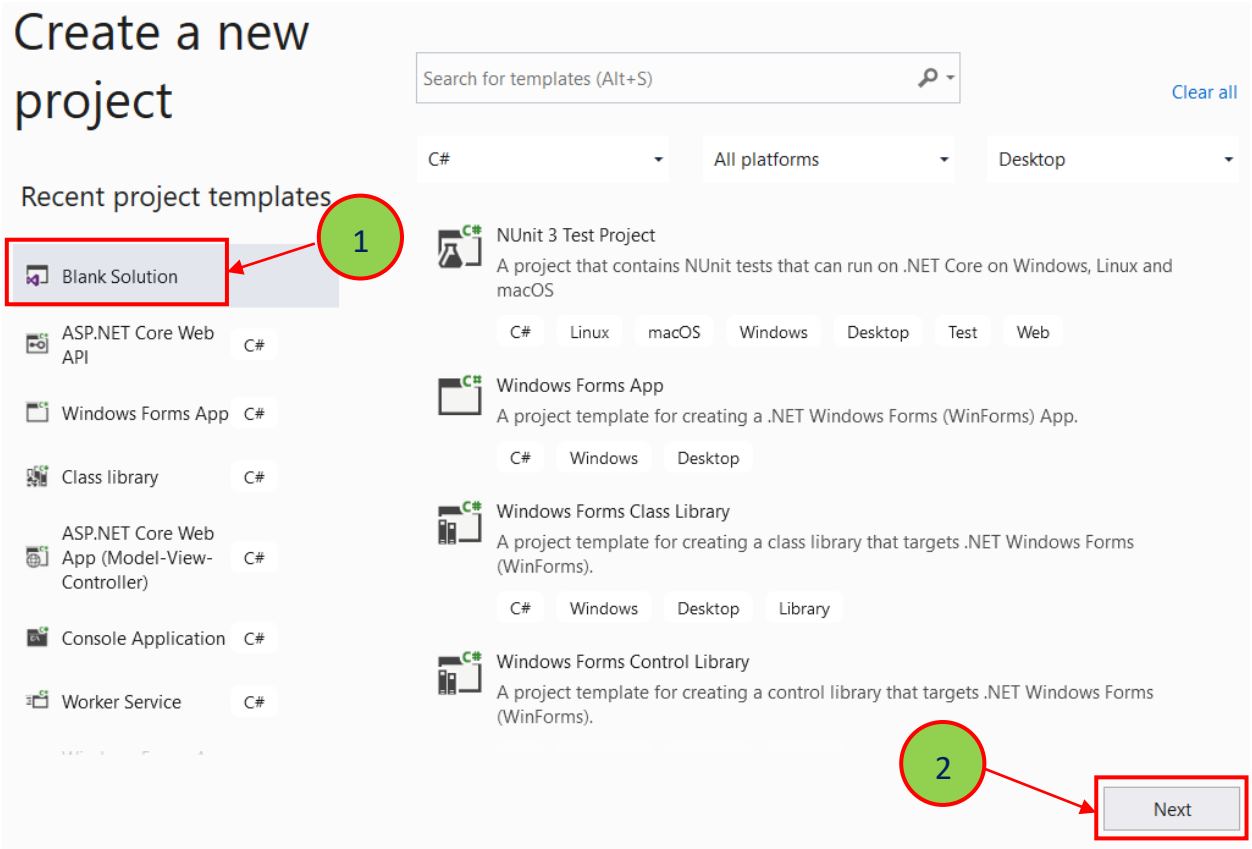
CarID	CarName	Manufacturer	Price	ReleasedYear
1	Accord	Honda Motor Company	24970.0000	2021
2	BMW 8 Series	BMW	85000.0000	2021
3	Clarity	Honda Motor Company	33400.0000	2021
4	Audi A6	Audi	14000.0000	2021
5	Everest Titanium 2.0L AT 4WD	Ford	60000.0000	2021
6	Ranger Wildtrak 2.0L AT 4×4	Ford	40000.0000	2021
7	Lexus IS	Lexus	100000.0000	2021
8	Lexus IS 300h	Lexus	220000.0000	2021

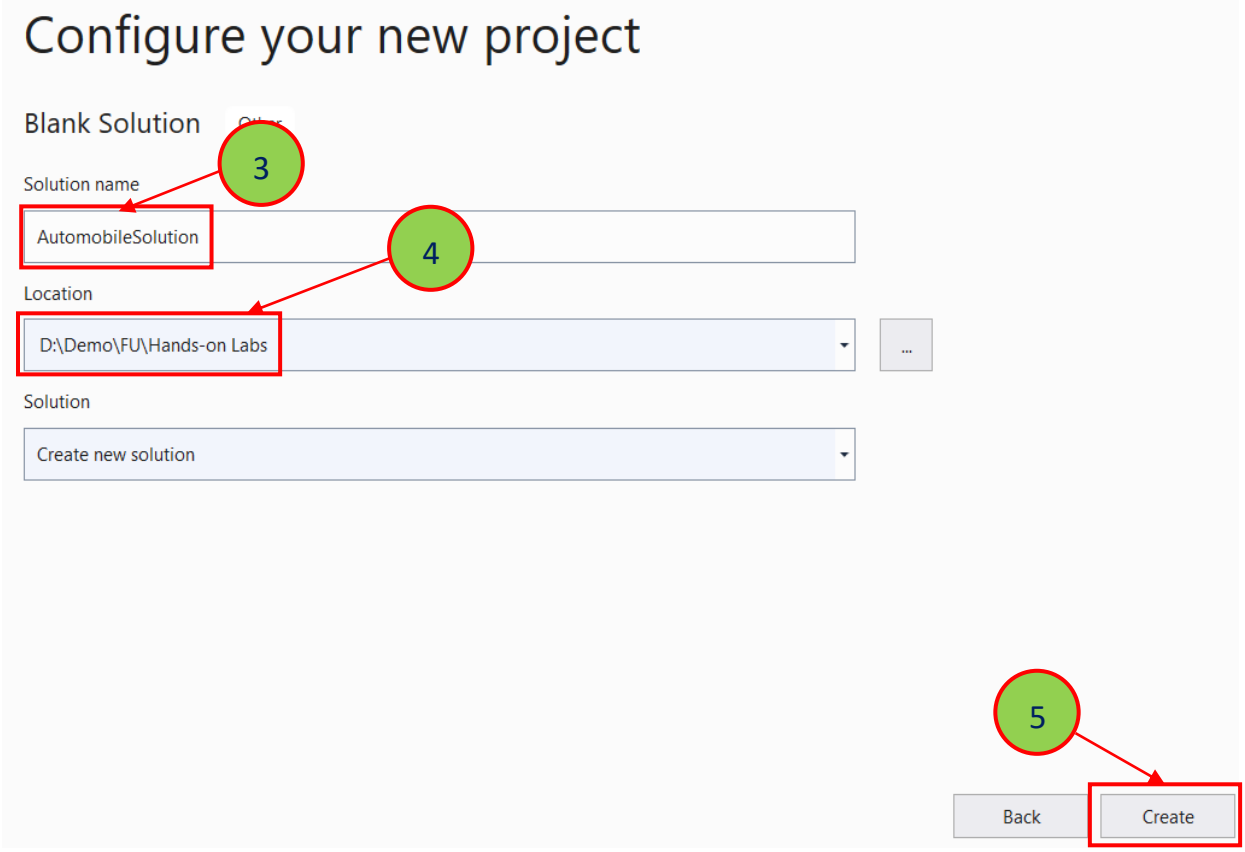
Activity 01: Build a solution by Visual Studio.NET

Create a Blank Solution named **AutomobileSolution** then add new a **Class Library** project named **AutomobileLibrary** and a WPF project named **AutomobileWPFApp**

Step 01. Create a Blank solution.

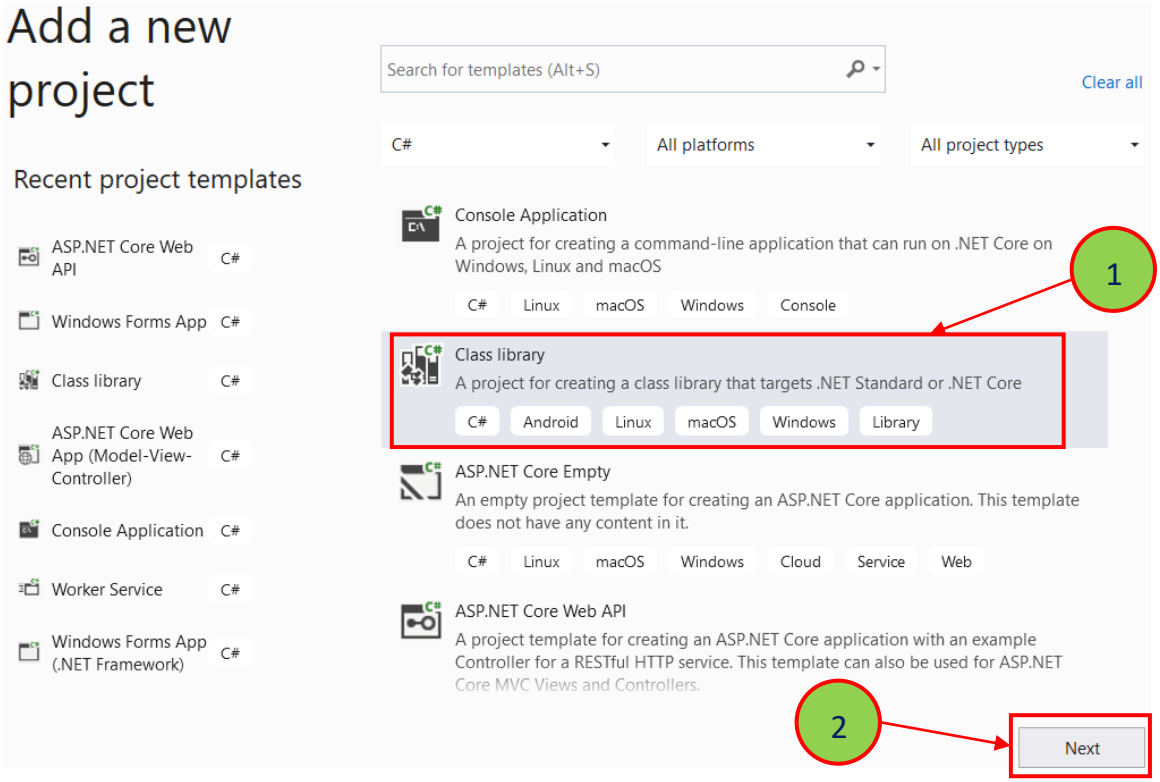
- Open the Visual Studio .NET application and performs steps as follows:

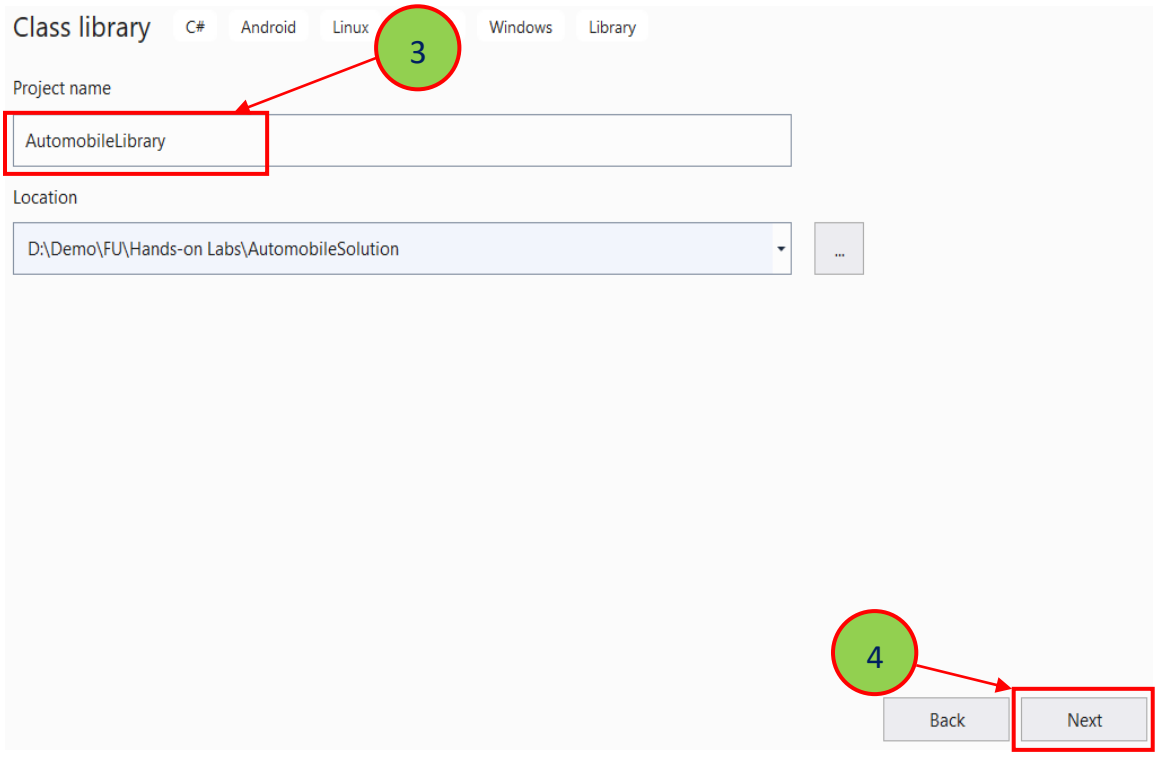

Create a new project
Search for templates (Alt+S)
Clear all
C#
All platforms
Desktop
Recent project templates
1
Blank Solution
ASP.NET Core Web API C#
Windows Forms App C#
Class library C#
ASP.NET Core Web App (Model-View-Controller) C#
Console Application C#
Worker Service C#
NUnit 3 Test Project
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Desktop Test Web
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop
Windows Forms Class Library
A project template for creating a class library that targets .NET Windows Forms (WinForms).
C# Windows Desktop Library
Windows Forms Control Library
A project template for creating a control library that targets .NET Windows Forms (WinForms).
2
Next

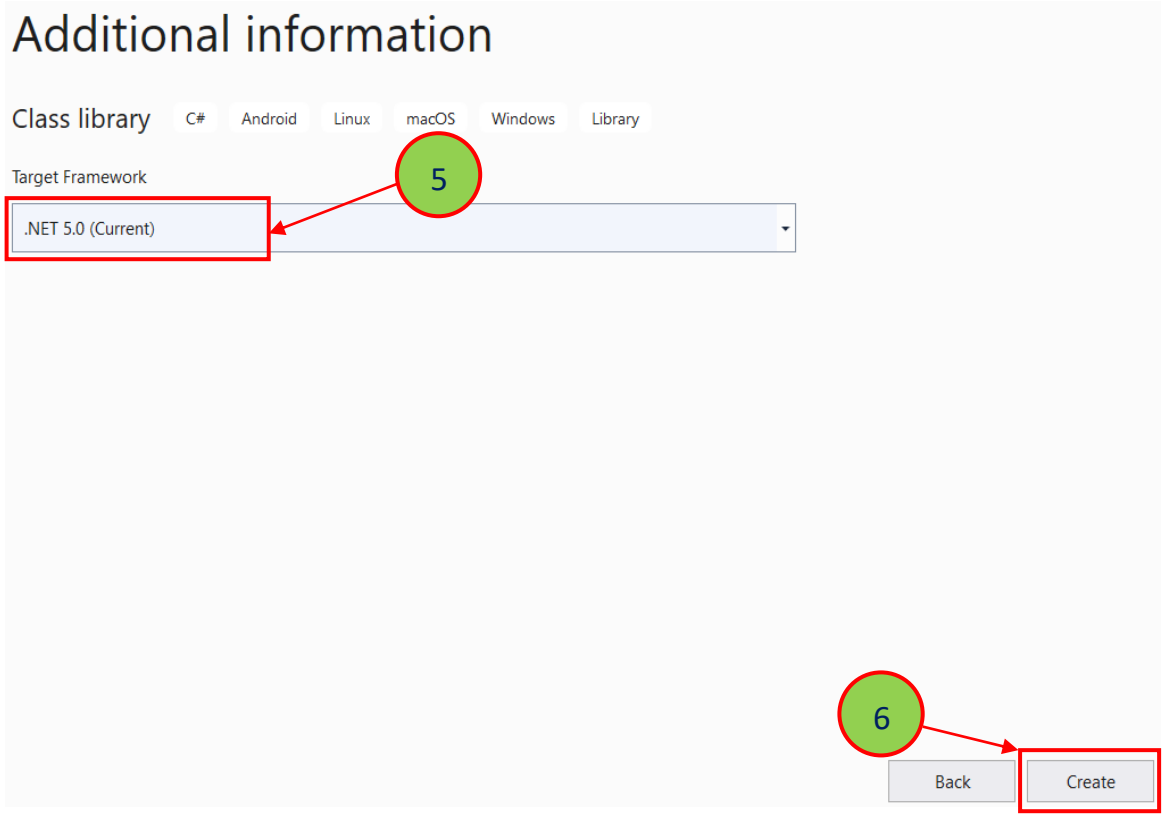

Configure your new project
Blank Solution
3
Solution name
AutomobileSolution
4
Location
D:\Demo\FU\Hands-on Labs
...
Solution
Create new solution
5
Back
Create

<u>Step 02</u>. Create a **Class Library** project.

- From the File menu | Add | New Project, on the Add New Project dialog, select “Class Library” and performs steps as follows:



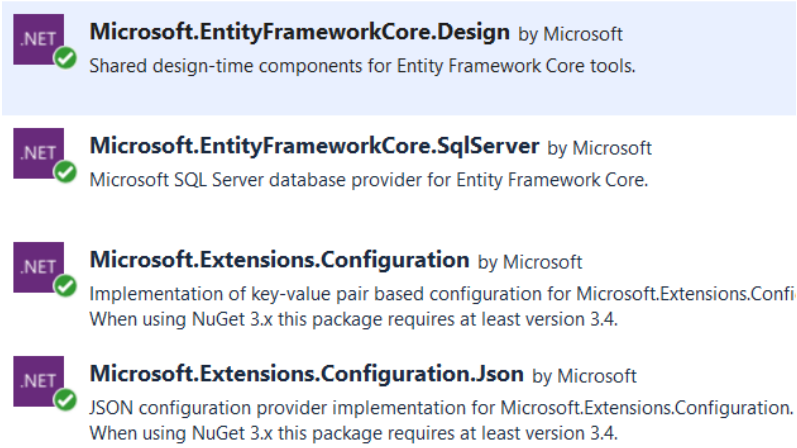




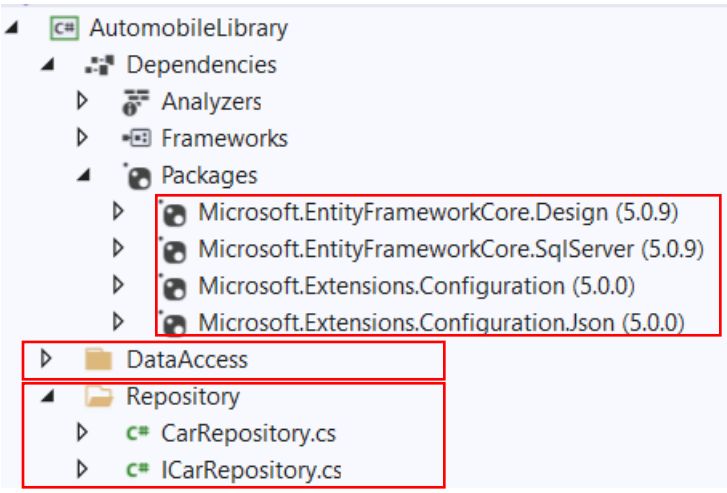
Step 03. Repeat **Step 02** to create a WPF project.

Activity 02: Write codes for the AutomobileLibrary project

Step 01. Install the following packages from NuGet:

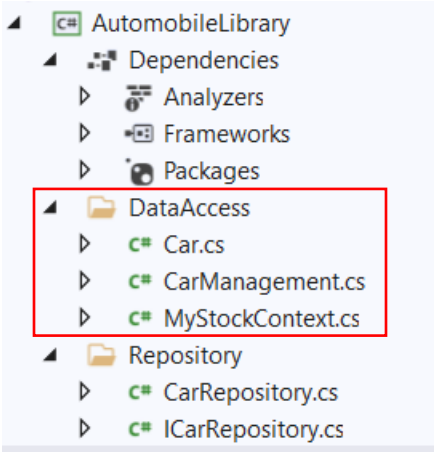


Step 02. Create folders and add classes to the project as follows:



Step 03. Right-click on project , select **Open In Terminal.** On **Developer PowerShell** dialog execute the following commands to generate model:

```
dotnet ef dbcontext scaffold "server =(local); database = MyStock;uid=sa;pwd=123;"
Microsoft.EntityFrameworkCore.SqlServer --output-dir DataAccess
```



Step 04.

On the **DataAccess** folder, add a class named **CarManagement.cs** and write codes as follows:

```
//...
using Microsoft.EntityFrameworkCore;
namespace AutomobileLibrary.DataAccess{
    public class CarManagement {
        //---------------------------------------------
        //Using Singleton Pattern
        private static CarManagement instance = null;
        private static readonly object instanceLock = new object();
        private CarManagement() { }
        public static CarManagement Instance{
            get{
                lock (instanceLock){
                    if (instance == null){
                        instance = new CarManagement();
                    }
                    return instance;
                }
            }
        }
        //---------------------------------------------
        public IEnumerable<Car> GetCarList() {
            List<Car> cars;
            try {
                var myStockDB = new MyStockContext();
                cars = myStockDB.Cars.ToList();
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }

            return cars;
        }
        //---------------------------------------------
        public Car GetCarByID(int carID){
            Car car = null;
            try
            {
                var myStockDB = new MyStockContext();
                car = myStockDB.Cars.SingleOrDefault(car => car.CarId == carID);
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }
            return car;
        }
```

```
        //-------------------------------------------------------------
        public void AddNew(Car car){
            try{
                Car _car = GetCarByID(car.CarId);
                if (_car == null){
                    var myStockDB = new MyStockContext();
                    myStockDB.Cars.Add(car);
                    myStockDB.SaveChanges();
                }
                else{
                    throw new Exception("The car is already exist.");
                }
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }
        }

        //-------------------------------------------------------------
        public void Update(Car car){
            try{
                Car c = GetCarByID(car.CarId);
                if (c != null){
                    var myStockDB = new MyStockContext();
                    myStockDB.Entry<Car>(car).State = EntityState.Modified;
                    myStockDB.SaveChanges();
                }
                else {
                    throw new Exception("The car does not already exist.");
                }
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }
        }
        //-------------------------------------------------------------
        public void Remove(Car car) {
            try{
                Car _car = GetCarByID(car.CarId);
                if (_car != null){
                    var myStockDB = new MyStockContext();
                    myStockDB.Cars.Remove(car);
                    myStockDB.SaveChanges();
                }
                else{
                    throw new Exception("The car does not already exist.");
                }
            }
            catch (Exception ex){
                throw new Exception(ex.Message);
            }
        }//end Remove
    }//end class
}//end namespace
```

Step 05. Write codes for **ICarRepository.cs** as follows:

```
using System.Collections.Generic;
using AutomobileLibrary.DataAccess;
namespace AutomobileLibrary.Repository{
    public interface ICarRepository {
        IEnumerable<Car> GetCars();
        Car GetCarByID(int carId);
        void InsertCar(Car car);
        void DeleteCar(Car car);
        void UpdateCar(Car car);
    }
}
```

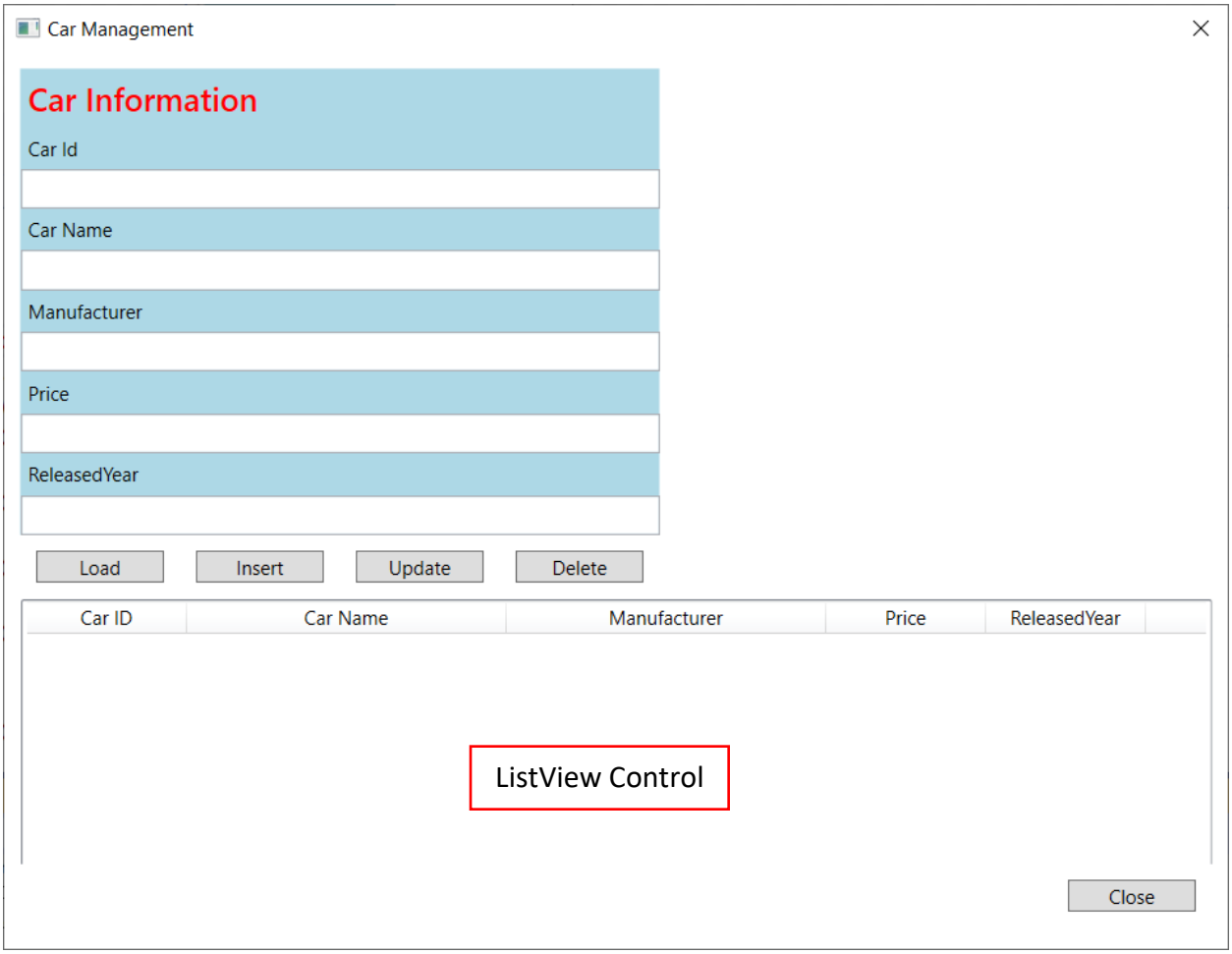
Step 06. Write codes for **CarRepository.cs** as follows:

```
using System.Collections.Generic;
using AutomobileLibrary.DataAccess;
namespace AutomobileLibrary.Repository{
    public class CarRepository : ICarRepository {

        public Car GetCarByID(int carId) => CarManagement.Instance.GetCarByID(carId);

        public IEnumerable<Car> GetCars() => CarManagement.Instance.GetCarList();

        public void InsertCar(Car car) => CarManagement.Instance.AddNew(car);

        public void DeleteCar(Car car) => CarManagement.Instance.Remove(car);

        public void UpdateCar(Car car) => CarManagement.Instance.Update(car);
    }
}
```

Activity 03: Design UI and write codes for AutomobileWPFApp project

Step 01.

On the **AutomobileWPFApp** project, rename **MainWindow.xaml** to **WindowCarManagement.xaml** and then design UI as follows**:**



❖ **XAML** code for WindowCarManagement.xaml

```
<Window x:Class="AutomobileWPFApp.WindowCarManagement"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:AutomobileWPFApp"
        mc:Ignorable="d"
        Title="Car Management" Width="780"
        WindowStartupLocation="CenterScreen" ResizeMode="NoResize" >
    <Grid>
        <DockPanel VerticalAlignment="Top" Margin="10">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto"/>
                    <RowDefinition Height="*"/>
                    <RowDefinition Height="4*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>
                <!--StackPanel for Label and TextBox controls-->
                <StackPanel Background="LightBlue"  Orientation ="Vertical"
                            HorizontalAlignment="Left" Width="400">
```

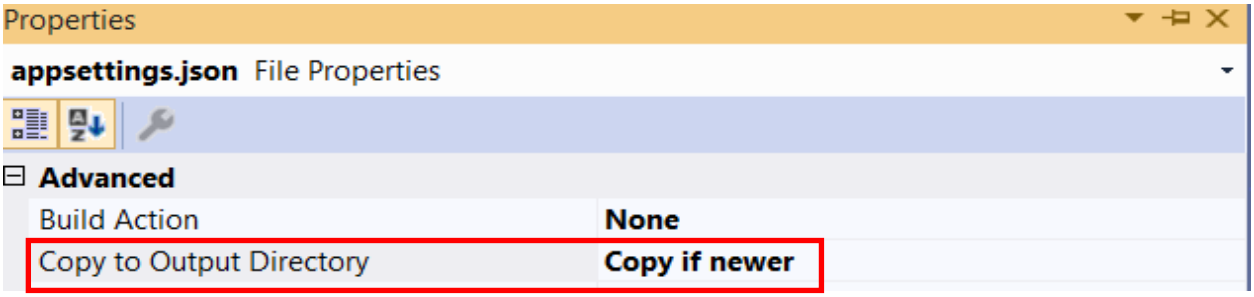
```
        <Label Name="lbTitle" Foreground="Red" FontWeight="DemiBold"
               FontSize="20" Content="Car Information"  />

        <Label Name="lbCarId" Content="Car Id"/>
        <TextBox Name="txtCarId" HorizontalAlignment="Stretch"
                 Height="25"
                 Text="{Binding Path=CarId, Mode=OneWay}"
                  DataContext="{Binding ElementName=lvCars,
                  Path=SelectedItem}" />

        <Label Name="lbCarName" Content="Car Name" />
        <TextBox Name="txtCarName" HorizontalAlignment="Stretch"
                 Height="25"
                 Text="{Binding Path=CarName, Mode=OneWay}"
                 DataContext="{Binding ElementName=lvCars,
                 Path=SelectedItem}" />

        <Label Name="lbManufacturer" Content="Manufacturer" />
        <TextBox Name="txtManufacturer" HorizontalAlignment="Stretch"
                 Height="25"
                 Text="{Binding Path=Manufacturer, Mode=OneWay}"
                 DataContext="{Binding ElementName=lvCars,
                 Path=SelectedItem}" />

        <Label Name="lbPrice" Content="Price" />
        <TextBox Name="txtPrice" HorizontalAlignment="Stretch"
                 Height="25"
                 Text="{Binding
                 Path=Price,StringFormat={}{0:N3}, Mode=OneWay}"
                 DataContext="{Binding ElementName=lvCars,
                 Path=SelectedItem}" />

        <Label Name="lbReleasedYear" Content="ReleasedYear" />
        <TextBox Name="txtReleasedYear" HorizontalAlignment="Stretch"
                 Height="25"
                 Text="{Binding Path=ReleasedYear, Mode=OneWay}"
                 DataContext="{Binding ElementName=lvCars,
                 Path=SelectedItem}" />
    </StackPanel>
    <!--StackPanel for Button controls-->
    <StackPanel Grid.Row="1" Orientation="Horizontal"
                HorizontalAlignment="Left">
        <Button x:Name="btnLoad" Margin="10"  Width="80" Content="Load"
                Click="btnLoad_Click"/>
        <Button x:Name="btnInsert" Margin="10"  Width="80" Content="Insert"
                Click="btnInsert_Click"/>
        <Button x:Name="btnUpdate"  Margin="10" Width="80"  Content="Update"
                Click="btnUpdate_Click"/>
        <Button x:Name="btnDelete" Margin="10"  Width="80" Content="Delete"
                Click="btnDelete_Click"/>
    </StackPanel>
    <!ListView control-->
    <ListView Grid.Row="2" Name="lvCars" Width="Auto" Height="Auto" >
        <ListView.View>
            <GridView>
                <GridViewColumn Header="Car ID"  Width="100"
                        DisplayMemberBinding="{Binding Path=CarId }"/>
                <GridViewColumn Header="Car Name" Width="200"
```

```
                                DisplayMemberBinding="{Binding Path=CarName}"/>
                        <GridViewColumn Header="Manufacturer"  Width="200"
                                DisplayMemberBinding="{Binding Path=Manufacturer }"/>
                        <GridViewColumn Header="Price" Width="100"
                                DisplayMemberBinding="{Binding Path=Price,
                                StringFormat={}{0:N3}}"/>
                        <GridViewColumn Header="ReleasedYear"  Width="100"
                                DisplayMemberBinding="{Binding Path=ReleasedYear}"/>
                    </GridView>
                </ListView.View>
            </ListView>
            <! Button control-->
            <Button Grid.Row="3" x:Name="btnClose" Margin="10"
                    HorizontalAlignment="Right" VerticalAlignment="Bottom"
                    Width="80" Content="Close" Click="btnClose_Click"  />
        </Grid>
    </DockPanel>
  </Grid>
</Window>
```

Step 02. Right-click on the project | Add | New Item, select **JavaScript JSON Configuration File** then rename to **appsettings.json**, click Add and write contents as follows:

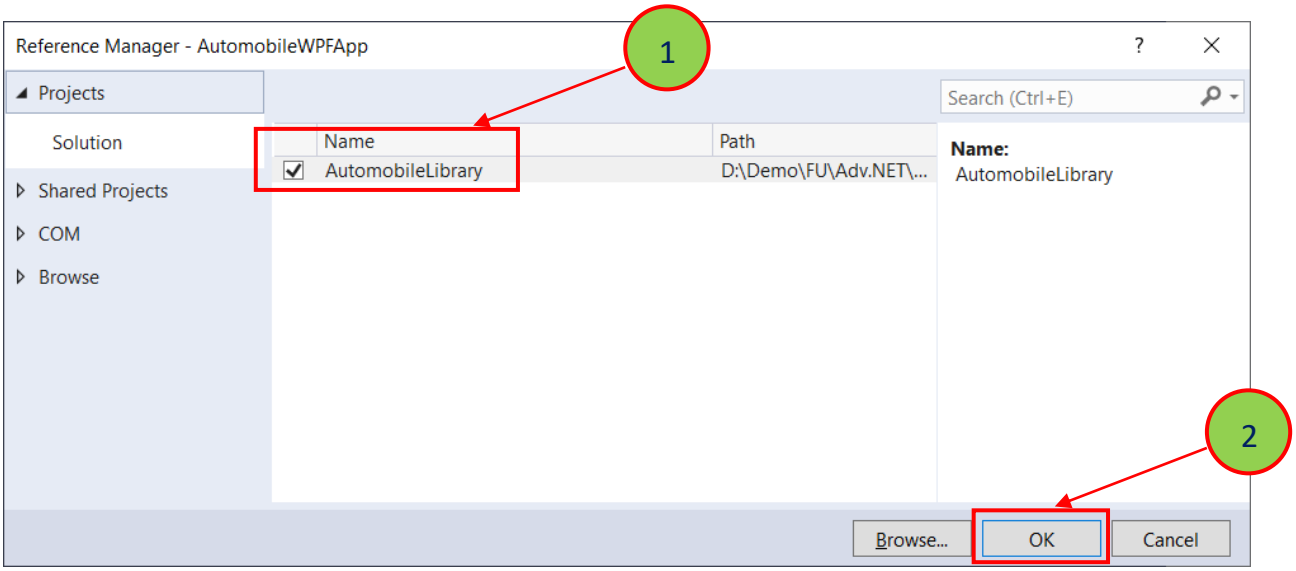
```
{
  "ConnectionStrings": {
    "MyStockDB": "Server=(local);uid=sa;pwd=123;database=MyStock"
  }
}
```

Step 03. Next, right-click on **appsettings.json** | Properties, select ***Copy if newer***

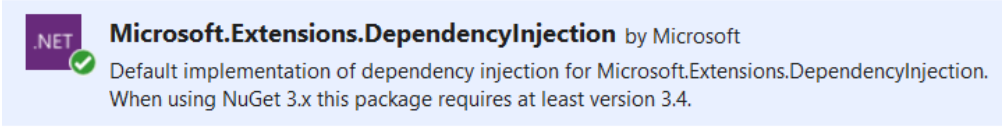


Step 04. Add a reference to the **AutomobileLibrary** project

Right-click on **AutomobileWPFApp** project, select Add | Project Reference, and perform as the below figure:



Step 05. Install the following package from NuGet:



Step 06. Write codes for **App.xaml.cs**:

```
//---
using Microsoft.Extensions.DependencyInjection;
using AutomobileLibrary.Repository;
namespace AutomobileWPFApp{
    public partial class App : Application {
        private ServiceProvider serviceProvider;
        public App(){
            //Config for DependencyInjection (DI)
            ServiceCollection services = new ServiceCollection();
            ConfigureServices(services);
            serviceProvider = services.BuildServiceProvider();
        }
        //-----------------------------------------------------
        private void ConfigureServices(ServiceCollection services){
            services.AddSingleton(typeof(ICarRepository), typeof(CarRepository));
            services.AddSingleton<WindowCarManagement>();
        }
        //-----------------------------------------------------
        private void OnStartup(object sender, StartupEventArgs e){
            var windowCarManagement = serviceProvider.GetService<WindowCarManagement>();
            windowCarManagement.Show();
        }
    }//end class
}
```

Step 07. Open **App.xaml** and then update XAML code as follows:

```
<Application x:Class="AutomobileWPFApp.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:AutomobileWPFApp"
             Startup="OnStartup">
    <Application.Resources>

    </Application.Resources>
</Application>
```

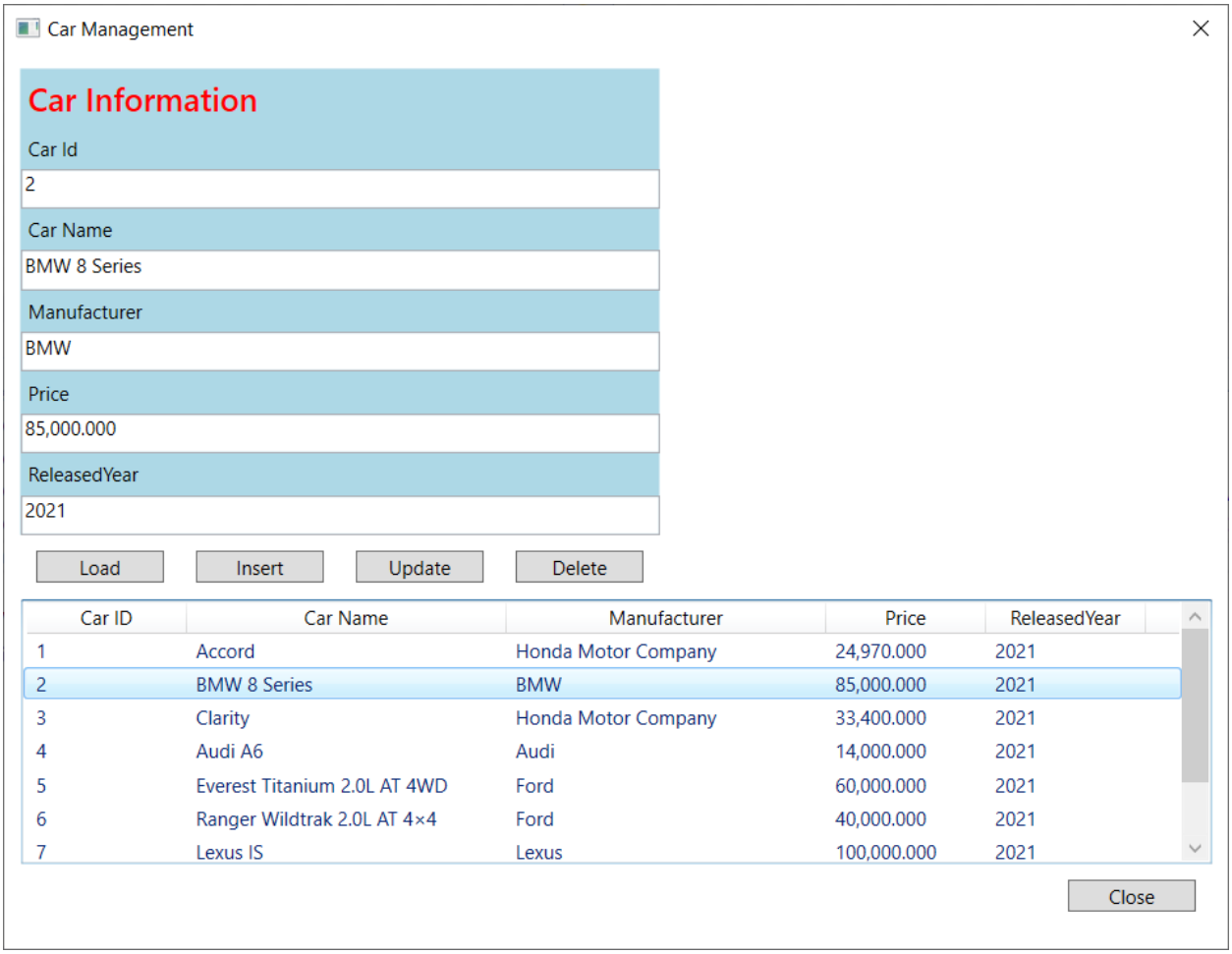
Step 08. Write codes for **WindowCarManagement.xaml.cs**:

```
//...
using AutomobileLibrary.Repository;
using AutomobileLibrary.DataAccess;

public partial class WindowCarManagement : Window{...
    ICarRepository carRepository;...
    public WindowCarManagement(ICarRepository repository){
        InitializeComponent();
        carRepository = repository;
    }
    //-------------------------------------------------------------
    private Car GetCarObject(){
        Car car = null;
        try...
        {
            car = new Car{
                CarId = int.Parse(txtCarId.Text),
                CarName = txtCarName.Text,
                Manufacturer = txtManufacturer.Text,
                Price = decimal.Parse(txtPrice.Text),
                ReleasedYear = int.Parse(txtReleasedYear.Text)
            };
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Get car");
        }
        return car;
    }//end GetCarObject

    //-------------------------------------------------------------
    public void LoadCarList() {
        lvCars.ItemsSource = carRepository.GetCars();
    }...
    //-------------------------------------------------------------
    private void btnLoad_Click(object sender, RoutedEventArgs e) {
        try {...
            LoadCarList();
        }
        catch (Exception ex){...
            MessageBox.Show(ex.Message, "Load car list");
        }
    }
```

```
    //-------------------------------------------------------
    private void btnInsert_Click(object sender, RoutedEventArgs e){
        try{
            Car car = GetCarObject();
            carRepository.InsertCar(car);
            LoadCarList();
            MessageBox.Show($"{car.CarName} inserted successfully ", "Insert car");
        }
        catch (Exception ex){
            MessageBox.Show(ex.Message, "Insert car");
        }
    }
    //-------------------------------------------------------
    private void btnUpdate_Click(object sender, RoutedEventArgs e){
        try{
            Car car = GetCarObject();
            carRepository.UpdateCar(car);
            LoadCarList();
            MessageBox.Show($"{car.CarName} updated successfully ", "Update car");
        }
        catch (Exception ex){
            MessageBox.Show(ex.Message, "Update car");
        }
    }

    //-------------------------------------------------------
    private void btnDelete_Click(object sender, RoutedEventArgs e){
        try{
            Car car = GetCarObject();
            carRepository.DeleteCar(car);
            LoadCarList();
            MessageBox.Show($"{car.CarName} deleted successfully ", "Delete car");
        }
        catch (Exception ex){
            MessageBox.Show(ex.Message, "Delete car");
        }
    }
    //-------------------------------------------------------
    private void btnClose_Click(object sender, RoutedEventArgs e)=> Close();
}//end class
```

Activity 05: Run the AutomobileWPFApp project and test all actions

Step 01. Click the **Load** button and display the result as the below figure.



Step 02. Enter the values on TextBoxes then click the **Insert** button to add a new car.

Step 03. Select a row on the ListView then click the **Delete** button to remove a Car.

Step 04. Click a row on the ListView and edit the values on TextBoxes, then click the **Update** button to update the car information.