**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

# 

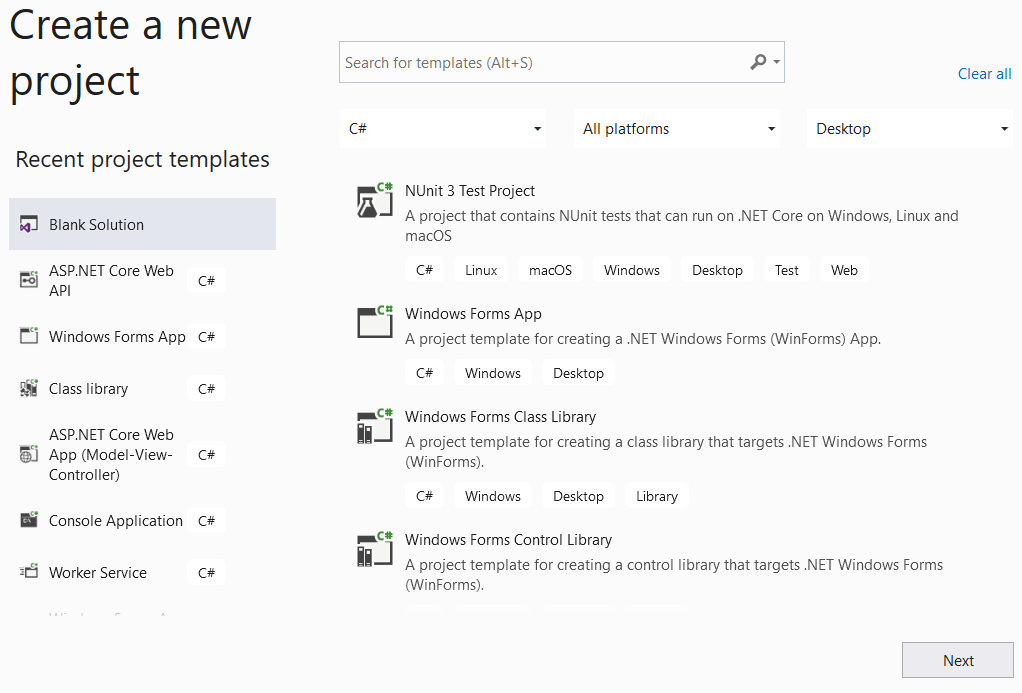
# 

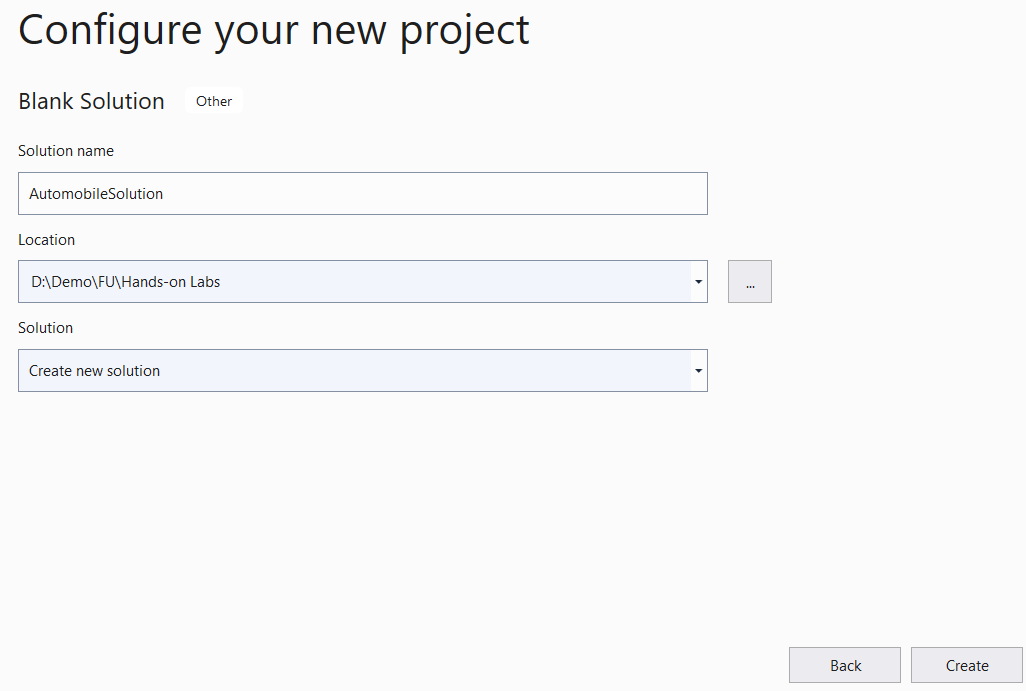
# Activity 01: Build a solution by Visual Studio.NET

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.

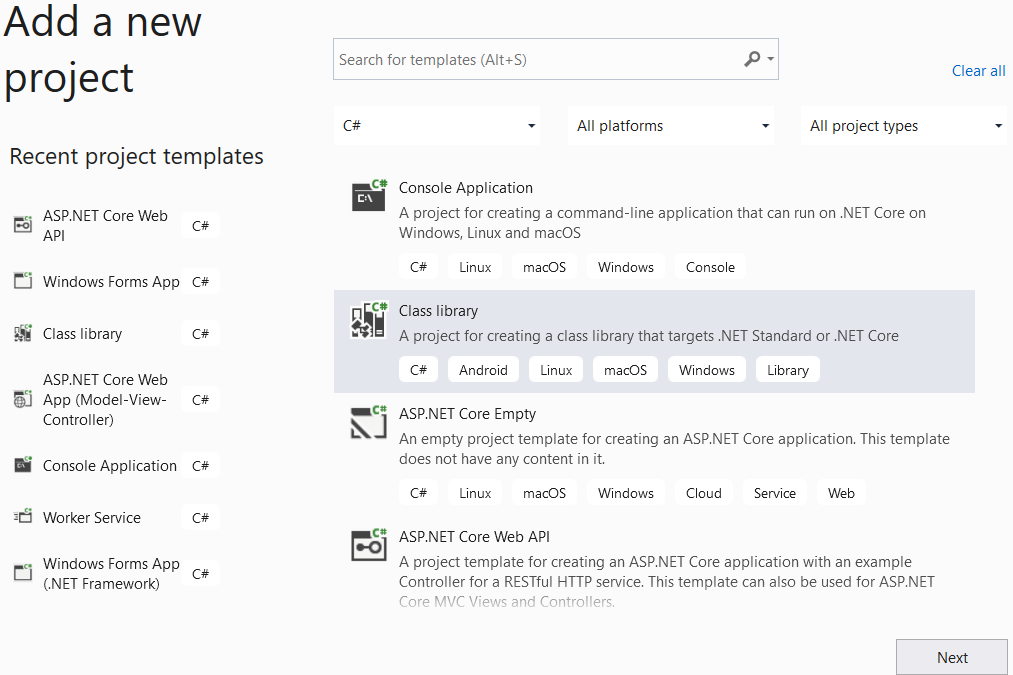
* Open the Visual Studio .NET application and performs steps as follows:

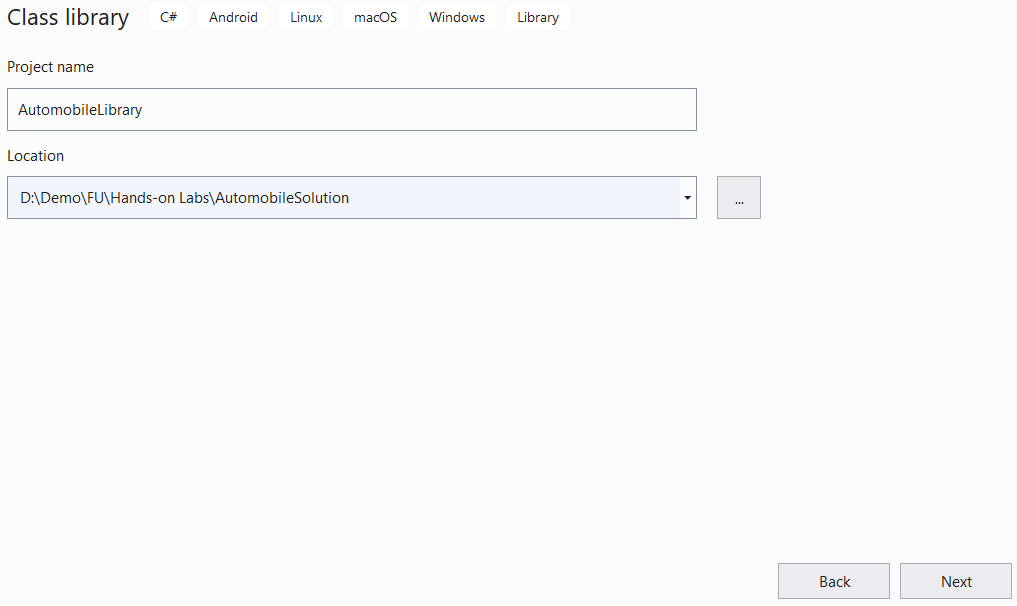




**Step 02.** Create a **Class Library** project.

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



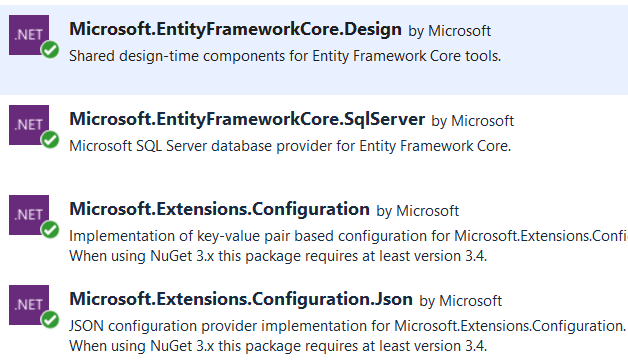


****

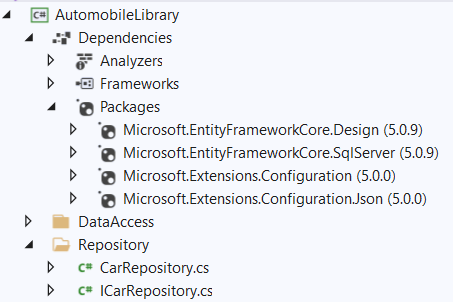
**Step 03.** Repeat **Step 02** to create a WPF project.

# Activity 02: Write codes for the AutomobileLibrary project

**Step 01**. Install the following packages from NuGet:

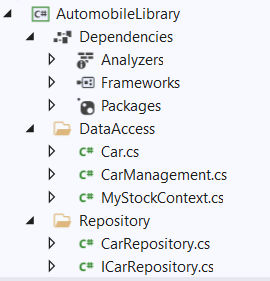
****

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



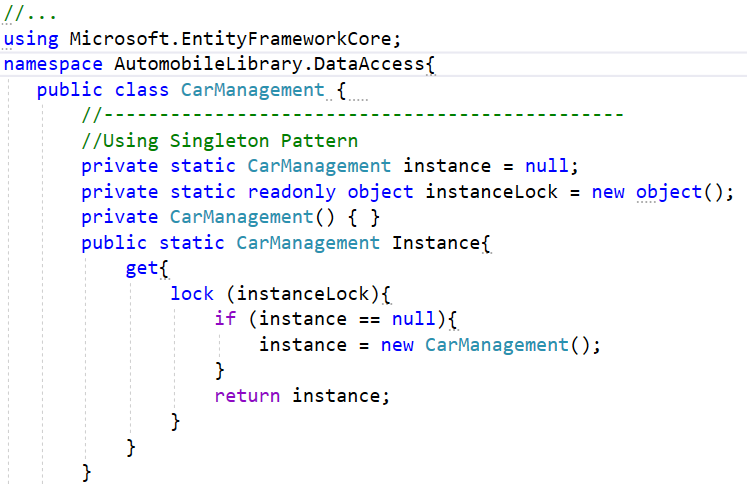
**Step 03**. Right-click on project , select **Open In Terminal.** On **Developer PowerShell** dialog execute the following commands to generate model:

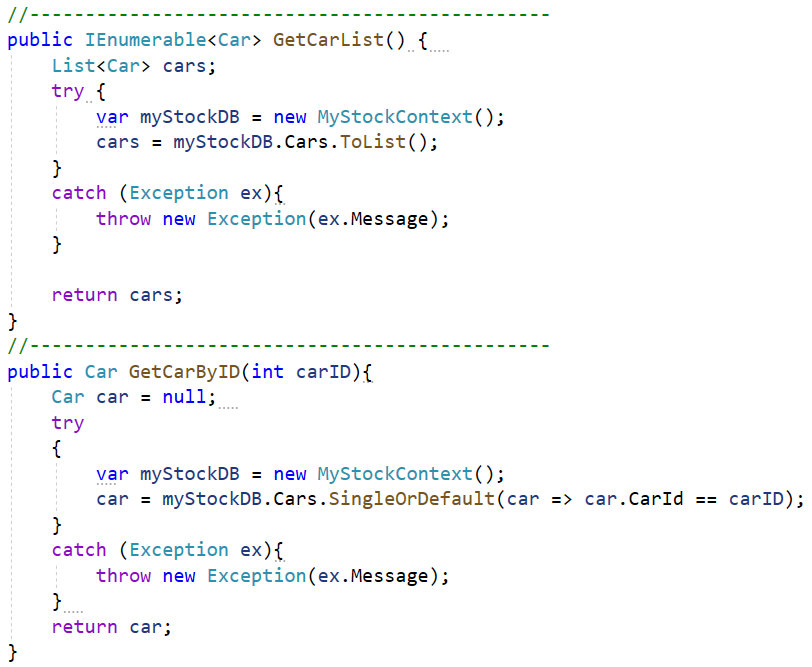
dotnet ef dbcontext scaffold "server =(local); database = **MyStock**;*uid=sa;pwd=123;*" Microsoft.EntityFrameworkCore.SqlServer --output-dir **DataAccess**

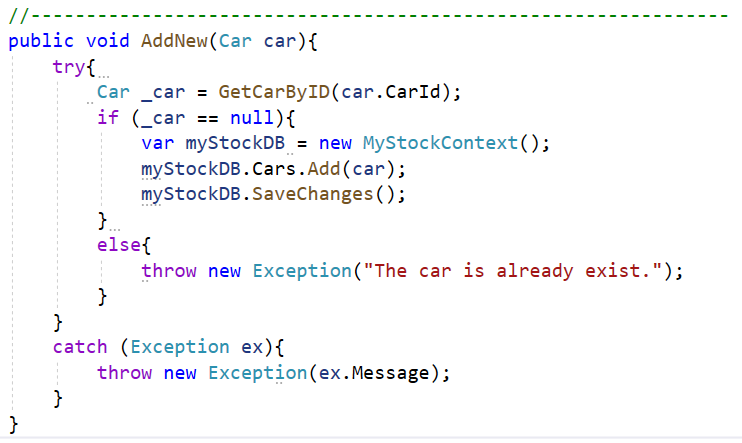


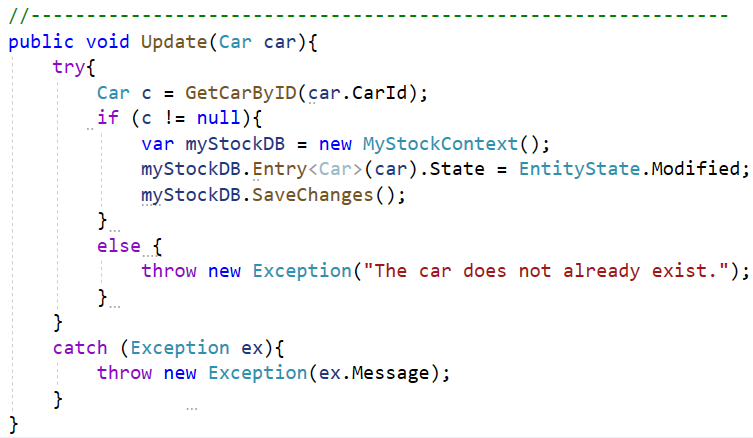
**Step 04**.

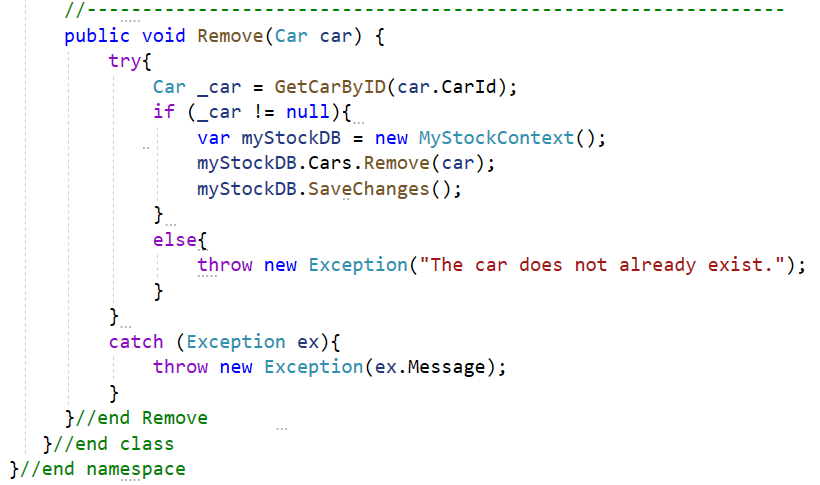
On the **DataAccess** folder, add a class named **CarManagement.cs** and write codes as follows:



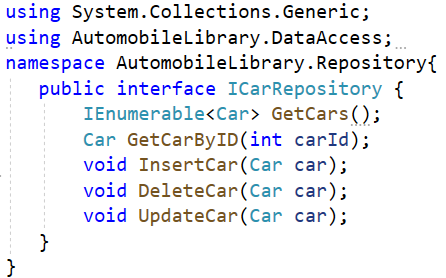




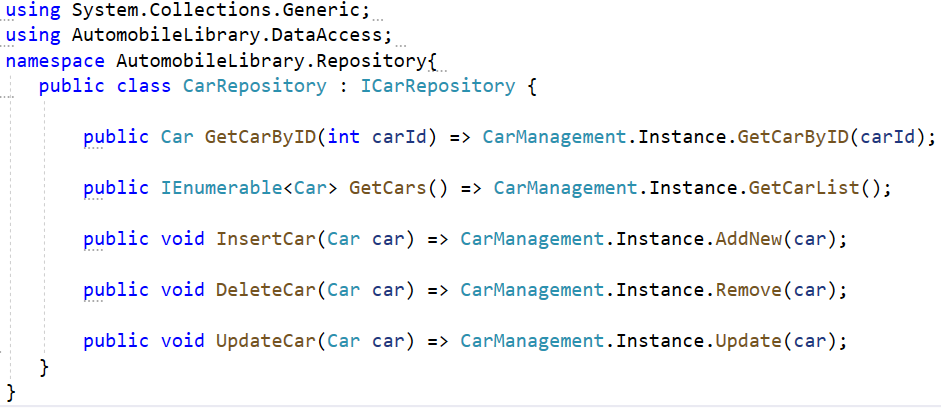




**Step 05**. Write codes for **ICarRepository.cs** as follows:



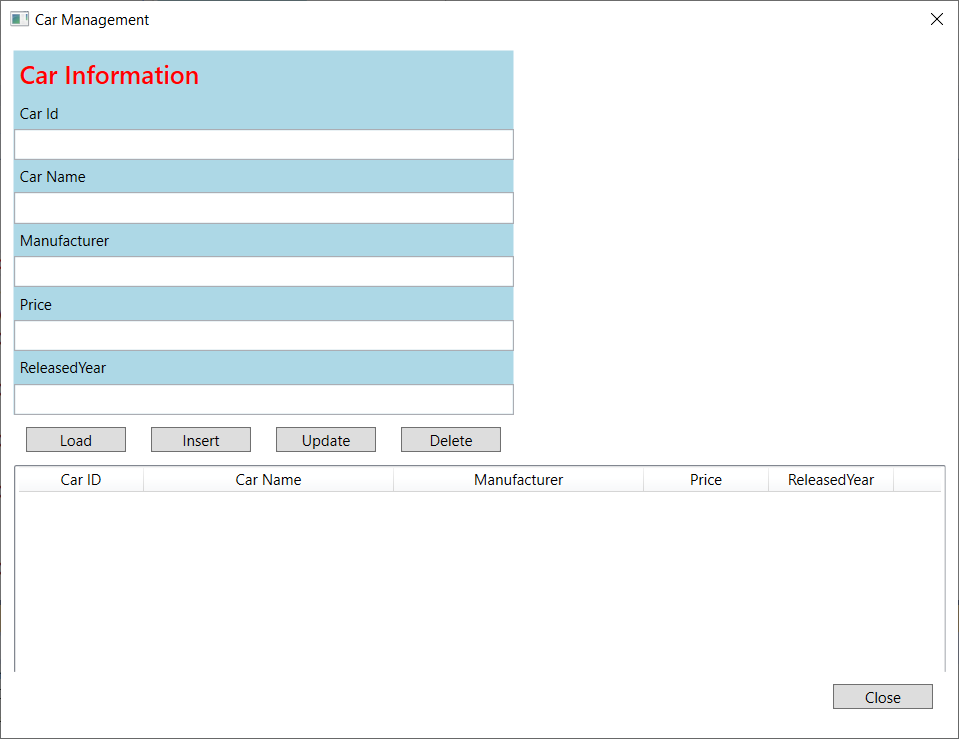
**Step 06**. Write codes for **CarRepository.cs** as follows:



# 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**:**



ListView Control

* **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***" />

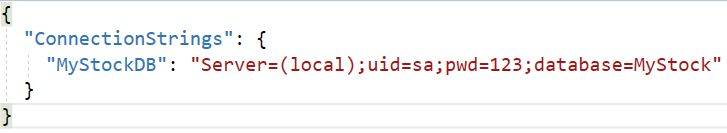
</Grid>

</DockPanel>

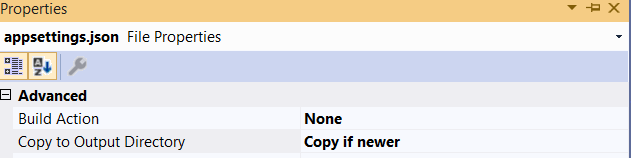
</Grid>

</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:



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



**Step 04**. Write codes for **frmCarDetails.cs**:

# 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:

# 

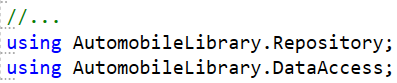
# Step 06. Write codes for App.xaml.cs:

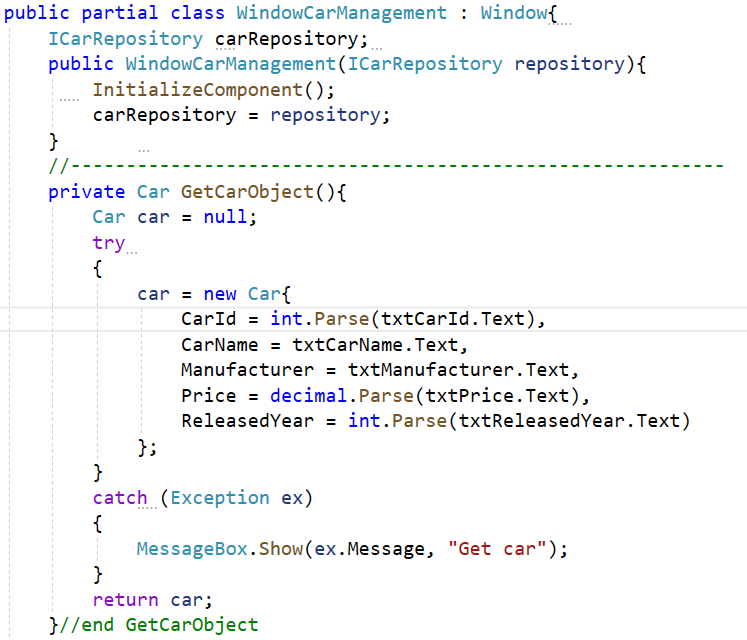
# 

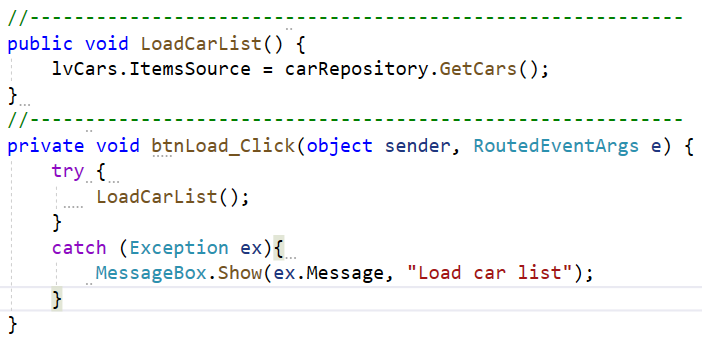
# Step 07. Open App.xaml and then update XAML code as follows:

# 

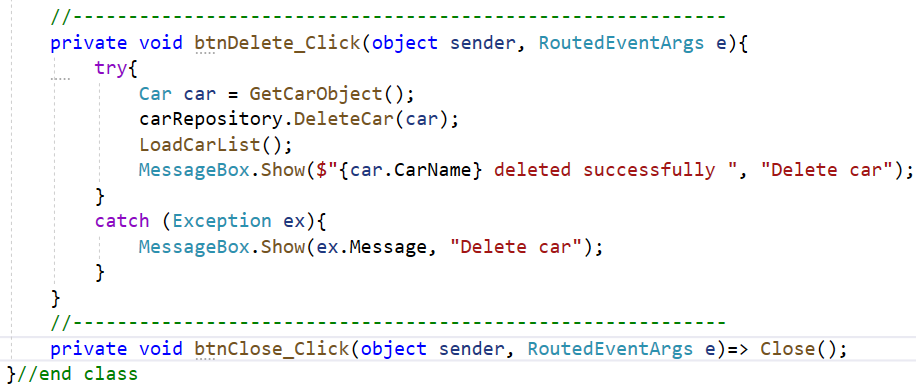
# Step 08. Write codes for WindowCarManagement.xaml.cs:





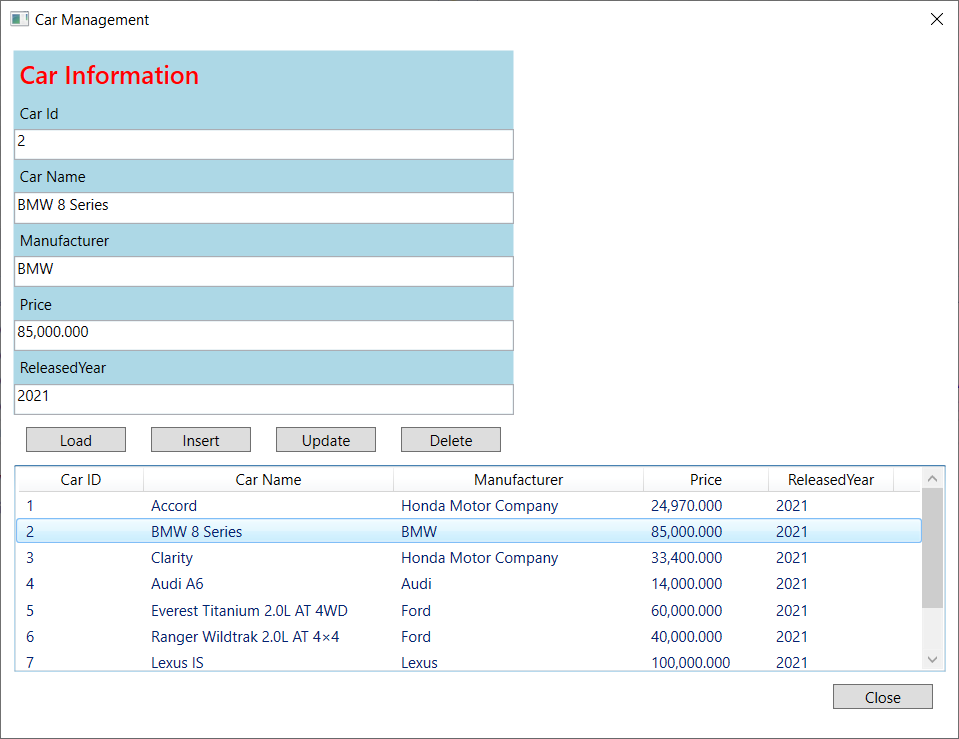






# Activity 05: Run the AutomobileWPFApp project and test all actions

# 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** buttonto update the car information.