Lab 01. Building an Product Management Application using LINQ and WPF

1. Introduction

Imagine you're an employee of a store named **ProductStore**. Your manager has asked you to develop a WPF application for product management. 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 "in-memory database" will be created to persist the car's data, so a collection is called List will be used for reading and managing product data.

2. Lab Objectives

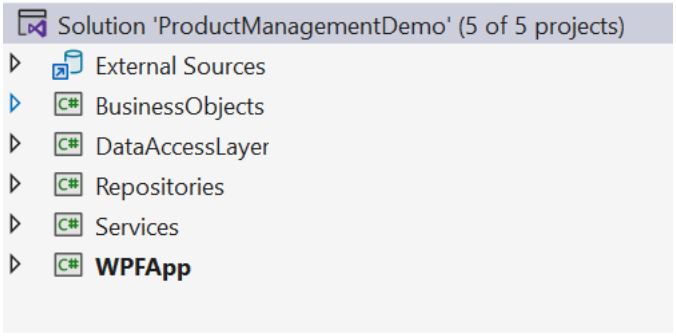
In this lab, you will:

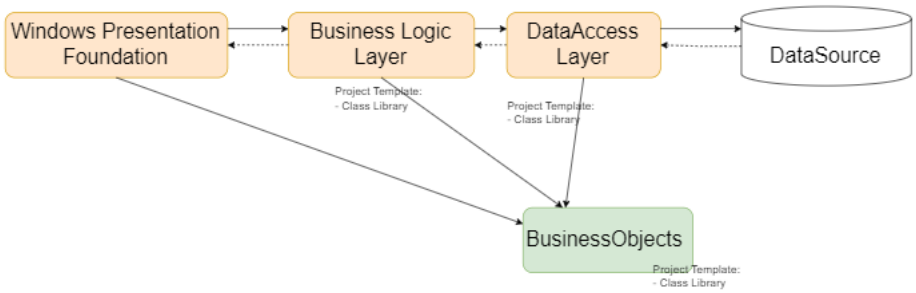
- Use the Visual Studio.NET to create WPF application and Class Library (.dll) project.
- Use the Visual Studio.NET to create Windows Forms and Class Library (.dll) project.
- Create a List of persisting products using LINQ to Object to find items.
- Apply passing data in WPF application
- Apply Repository pattern in a project.
- Add CRUD action methods to WPF application.
- Run the project and test the WPF application actions.

Activity 01: Build a solution by Visual Studio.NET

Create a Blank Solution named **ProductManagementDemo** then add new a **Class Library** project named **BusinessObjects, DataAccessObjects, Repositories, Services** and a WPF project named **ProductManagement**

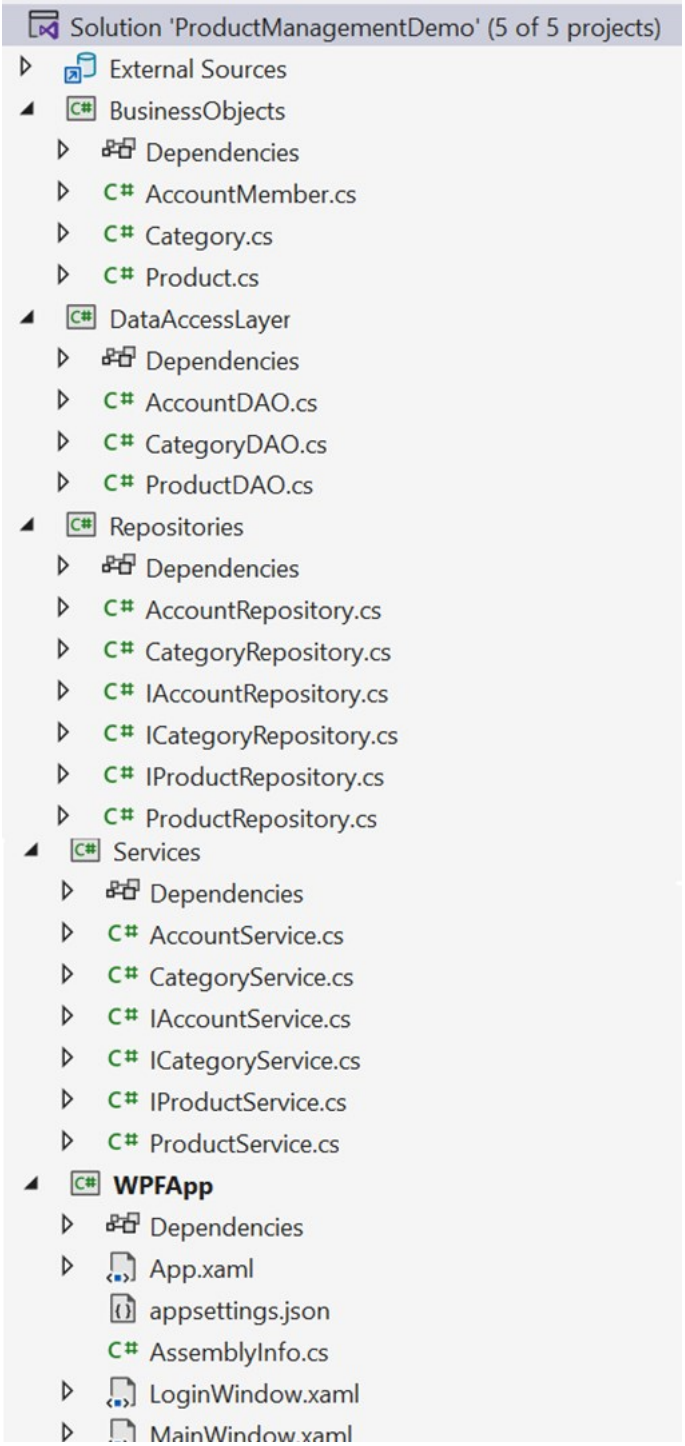
Step 01. Create a Blank solution.
Step 02. Create 4 **Class Library** projects.
Step 03. Create a WPF project.



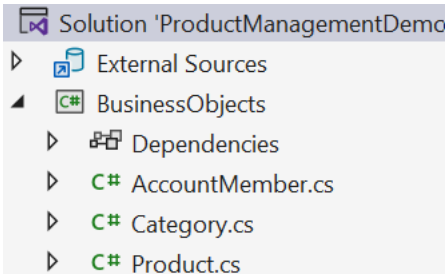


Note:

- **Data Source** in this case is the "**In memory Database"** (using List collection)
- **Services Project** – This project represents a layer or component responsible for implementing the business logic of an application.
- **Repository Project** – This project provides an abstraction layer between the application's business logic and the underlying data source.
- **Data Access Layer Project** – This project used to abstract and encapsulate the logic for accessing data from a data source, such as a database.


Solution 'ProductManagementDemo' (5 of 5 projects)
External Sources
BusinessObjects
Dependencies
AccountMember.cs
Category.cs
Product.cs
DataAccessLayer
Dependencies
AccountDAO.cs
CategoryDAO.cs
ProductDAO.cs
Repositories
Dependencies
AccountRepository.cs
CategoryRepository.cs
IAccountRepository.cs
ICategoryRepository.cs
IProductRepository.cs
ProductRepository.cs
Services
Dependencies
AccountService.cs
CategoryService.cs
IAccountService.cs
ICategoryService.cs
IProductService.cs
ProductService.cs
WPFApp
Dependencies
App.xaml
appsettings.json
AssemblyInfo.cs
LoginWindow.xaml
MainWindow.xaml

Activity 02: Write codes for the BusinessObjects project



Step 01. On the **BusinessObjects** project, add a class named **Category.cs** and write codes as follows:

```
1   using System.Collections.Generic;
2
3   namespace BusinessObjects
4   {
        25 references
5       public partial class Category
6       {
            0 references
7           public Category()
8           {
9               Products = new HashSet<Product>();
10          }
11
            8 references
12          public Category(int catID, string catName)
13          {
14              this.CategoryId = catID;
15              this.CategoryName = catName;
16          }
17
            1 reference
18          public int CategoryId { get; set; }
            1 reference
19          public string CategoryName { get; set; }
20
            1 reference
21          public virtual ICollection<Product> Products { get; set; }
22      }
23  }
```

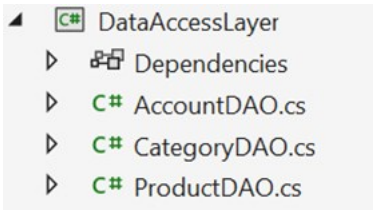
Step 02. On the **BusinessObjects** project, add a class named **Product.cs** and write codes as follows:

```
namespace BusinessObjects
{
    46 references
    public partial class Product
    {
        6 references
        public Product(int id,string name,int catId,
            short unitInStock,decimal price) {
            this.ProductId= id;
            this.ProductName = name;
            this.CategoryId= catId;
            this.UnitsInStock = unitInStock;
            this.UnitPrice = price;
        }
        12 references
        public int ProductId { get; set; }
        7 references
        public string ProductName { get; set; }
        7 references
        public int? CategoryId { get; set; }
        7 references
        public short? UnitsInStock { get; set; }
        7 references
        public decimal? UnitPrice { get; set; }
        0 references
        public virtual Category Category { get; set; }
    }
}
```

Step 03. On the **BusinessObjects** project, add a class named **AccountMember.cs** and write codes as follows:

```
namespace BusinessObjects
{
    8 references
    public partial class AccountMember
    {
        1 reference
        public string MemberId { get; set; } = null!;
        2 references
        public string MemberPassword { get; set; } = null!;
        0 references
        public string FullName { get; set; } = null!;
        0 references
        public string? EmailAddress { get; set; }
        2 references
        public int? MemberRole { get; set; }
    }
}
```

Activity 03: Write codes for the DataAccessLayer project



Step 01. On the **DataAccessLayer** project, add a class named **CategoryDAO.cs** and write codes as follows:

```
using System.Collections.Generic;
using BusinessObjects;

namespace DataAccessLayer
{
    1 reference
    public class CategoryDAO
    {
        1 reference
        public static List<Category> GetCategories()
        {
            Category beverages = new Category(1, "Beverages");
            Category condiments = new Category(2, "Condiments");
            Category confections = new Category(3, "Confections");
            Category dairy = new Category(4, "Dairy Products");
            Category grains = new Category(5, "Grains/Cereals");
            Category meat = new Category(6, "Meat/Poultry");
            Category produce = new Category(7, "Produce");
            Category seafood = new Category(8, "Seafood");

            var listCategories = new List<Category>();
            try
            {
                listCategories.Add(beverages);
                listCategories.Add(condiments);
                listCategories.Add(confections);
                listCategories.Add(dairy);
                listCategories.Add(grains);
                listCategories.Add(meat);
                listCategories.Add(produce);
                listCategories.Add(seafood);
            }
            catch (Exception e)
            {
                throw new Exception(e.Message);
            }
            return listCategories;
        }
    }
}
```

Step 02. On the **DataAccessLayer** project, add a class named **ProductDAO.cs** and write codes as follows:

The details of functions in ProductDAO.cs:

```
using System.Collections.Generic;
using System.Linq;
using BusinessObjects;

namespace DataAccessLayer
{
    3 references
    public class ProductDAO
    {
        private static List<Product> listProducts;

        1 reference
        public ProductDAO()
        {

            Product chai = new Product(1, "Chai", 3, 12, 18);
            Product chang = new Product(2, "Chang", 1, 23, 19);
            Product aniseed = new Product(3, "Aniseed Syrup", 2, 23, 10);
            listProducts = new List<Product> { chai, chang, aniseed };
            //Product chef = new Product(4, "Chef Anton's Cajun Seasoning", 2, 34, 22);
            //Product chefMix = new Product(5, "Chef Anton's Gumbo Mix", 2, 45, 34);
            //Product grandma = new Product(6, "Grandma's Boysenberry Spread", 2, 21, 25);
            //Product uncle = new Product(7, "Uncle Bob's Organic Dried Pears", 7, 22, 30);
            //Product northwoods = new Product(8, "Northwoods Cranberry Sauce", 2, 10, 40);
            //Product mishi = new Product(9, "Mishi Kobe Niku", 6, 12, 97);
            //Product ikura = new Product(10, "Ikura", 8, 13, 32);
           // listProducts = new List<Product> { chai, chang, aniseed, chef, chefMix, grandma
        }
        1 reference
        public List<Product> GetProducts()
        {
            return listProducts;
        }
```

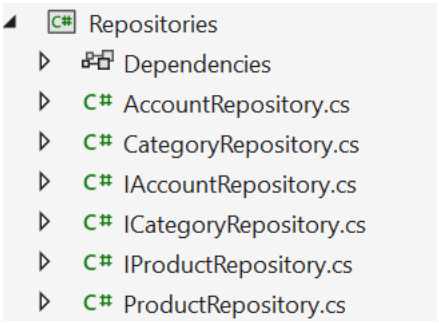
```
        public static List<Product> GetProducts()
        {
            var listProducts = new List<Product>();
            try
            {
                using var db = new MyStoreContext();
                listProducts = db.Products.ToList();
            }
            catch (Exception e) { }

            return listProducts;
        }
```

```
32          public void SaveProduct(Product p)
33          {
34              listProducts.Add(p);
35          }
36
            1 reference
37          public void UpdateProduct(Product product)
38          {
39              foreach (Product p in listProducts.ToList())
40              {
41                  if (p.ProductId == product.ProductId)
42                  {
43                      p.ProductId = product.ProductId;
44                      p.ProductName = product.ProductName;
45                      p.UnitPrice = product.UnitPrice;
46                      p.UnitsInStock = product.UnitsInStock;
47                      p.CategoryId = product.CategoryId;
48                  }
49              }
50
51          }
52
53          public void DeleteProduct(Product product)
54          {
55              foreach (Product p in listProducts.ToList())
56              {
57                  if (p.ProductId == product.ProductId)
58                  {
59                      listProducts.Remove(p);
60                  }
61              }
62          }
63
            1 reference
64          public Product GetProductById(int id)
65          {
66              foreach (Product p in listProducts.ToList())
67              {
68                  if (p.ProductId == id)
69                  {
70                      return p;
71                  }
72              }
73              return null;
74
75          }
76      }
```

Step 03. Write codes for **AccountDAO.cs** as follows:

```
using BusinessObjects;

namespace DataAccessLayer
{
    1 reference
    public class AccountDAO
    {
        1 reference
        public static AccountMember GetAccountById(string accountID)
        {
            AccountMember accountMember = new AccountMember();
            if (accountID.Equals("PS0001")) // just for demonstration
            {
                accountMember.MemberId = accountID;
                accountMember.MemberPassword = "@1";
                accountMember.MemberRole = 1;
            }
            return accountMember;
        }
    }
}
```

Activity 04: Write codes for the Repositories project



Step 01. On the **Repositories** project, add an interface named **ICatergoryRepository.cs** and write codes as follows:

```
using System.Collections.Generic;
using BusinessObjects;

namespace Repositories
{
    2 references
    public interface ICategoryRepository
    {
        2 references
        List<Category> GetCategories();
    }
}
```

Step 02. On the **Repositories** project, add an interface named **IProductRepository.cs** and write codes as follows:

```
using System.Collections.Generic;
using BusinessObjects;

namespace Repositories
{
    2 references
    public interface IProductRepository
    {
        2 references
        void SaveProduct(Product p);
        2 references
        void DeleteProduct(Product p);
        2 references
        void UpdateProduct(Product p);
        2 references
        List<Product> GetProducts();
        2 references
        Product GetProductById(int id);
    }
}
```

Step 03. On the **Repositories** project, add an interface named **IAccountRepository.cs** and write codes as follows:

```
using BusinessObjects;

namespace Repositories
{
    2 references
    public interface IAccountRepository
    {
        2 references
        AccountMember GetAccountById(string accountID);
    }
}
```

Step 04. Write codes for class **CategoryRepository.cs** as follows:

```
using System.Collections.Generic;
using BusinessObjects;
using DataAccessLayer;

namespace Repositories
{
    1 reference
    public class CategoryRepository : ICategoryRepository
    {
        2 references
        public List<Category> GetCategories() => CategoryDAO.GetCategories();
    }
}
```

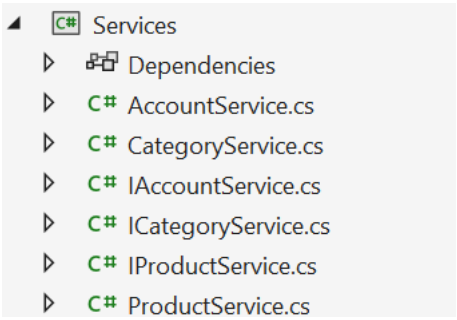
Step 05. Write codes for class **ProductRepository.cs** as follows:

```
using BusinessObjects;
using System.Collections.Generic;
using DataAccessLayer;

namespace Repositories
{
    1 reference
    public class ProductRepository : IProductRepository
    {
        2 references
        public void DeleteProduct(Product p) => ProductDAO.DeleteProduct(p);
        2 references
        public void SaveProduct(Product p) => ProductDAO.SaveProduct(p);
        2 references
        public void UpdateProduct(Product p) => ProductDAO.UpdateProduct(p);
        2 references
        public List<Product> GetProducts() => ProductDAO.GetProducts();
        2 references
        public Product GetProductById(int id)=>ProductDAO.GetProductById(id);


    }
}
```

Step 06. Write codes for class **AccountRepository.cs** as follows:

```
using BusinessObjects;
using DataAccessLayer;

namespace Repositories
{
    1 reference
    public class AccountRepository : IAccountRepository
    {
        2 references
        public AccountMember GetAccountById(string accountID)
            =>AccountDAO.GetAccountById(accountID);

    }
}
```

Activity 05: Write codes for the Services project



Step 01. On the **Services** project, add an interface named **ICatergoryService.cs** and write codes as follows:

```
using BusinessObjects;
using System.Collections.Generic;
namespace Services
{
    2 references
    public interface ICategoryService
    {
        2 references
        List<Category> GetCategories();
    }
}
```

Step 02. On the **Services** project, add an interface named **IProductService.cs** and write codes as follows:

```
using BusinessObjects;
using System.Collections.Generic;
namespace Services
{
    2 references
    public interface IProductService
    {
        2 references
        void SaveProduct(Product p);
        2 references
        void DeleteProduct(Product p);
        2 references
        void UpdateProduct(Product p);
        2 references
        List<Product> GetProducts();
        2 references
        Product GetProductById(int id);
    }
}
```

Step 03. On the **Services** project, add an interface named **IAccountService.cs** and write codes as follows:

```
using BusinessObjects;

namespace Services
{
    2 references
    public interface IAccountService
    {
        2 references
        AccountMember GetAccountById(string accountID);
    }
}
```

Step 04. Write codes for class **CategoryService.cs** as follows:

```
using System.Collections.Generic;
namespace Services
{
    2 references
    public class CategoryService : ICategoryService
    {
        private readonly ICategoryRepository iCategoryRepository;

        1 reference
        public CategoryService()
        {
            iCategoryRepository = new CategoryRepository();
        }

        2 references
        public List<Category> GetCategories()
        {
            return iCategoryRepository.GetCategories();
        }
    }
}
```

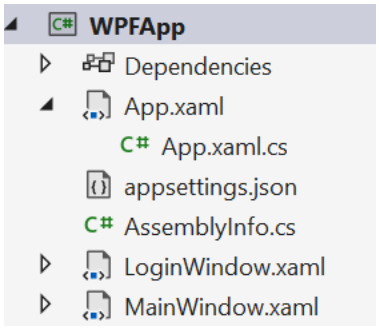
Step 05. Write codes for class **ProductService.cs** as follows:

```
using BusinessObjects;
using Repositories;
using System.Collections.Generic;

namespace Services
{
    2 references
    public class ProductService : IProductService
    {
        private readonly IProductRepository iProductRepository;

        1 reference
        public ProductService()
        {
            iProductRepository = new ProductRepository();
        }
        2 references
        public void DeleteProduct(Product p)
        {
            iProductRepository.DeleteProduct(p);
        }

        2 references
        public Product GetProductById(int id)
        {
            return iProductRepository.GetProductById(id);
        }

        2 references
        public List<Product> GetProducts()
        {
            return iProductRepository.GetProducts();
        }

        2 references
        public void SaveProduct(Product p)
        {
            iProductRepository.SaveProduct(p);
        }

        2 references
        public void UpdateProduct(Product p)
        {
            iProductRepository.UpdateProduct(p);
        }
    }
}
```

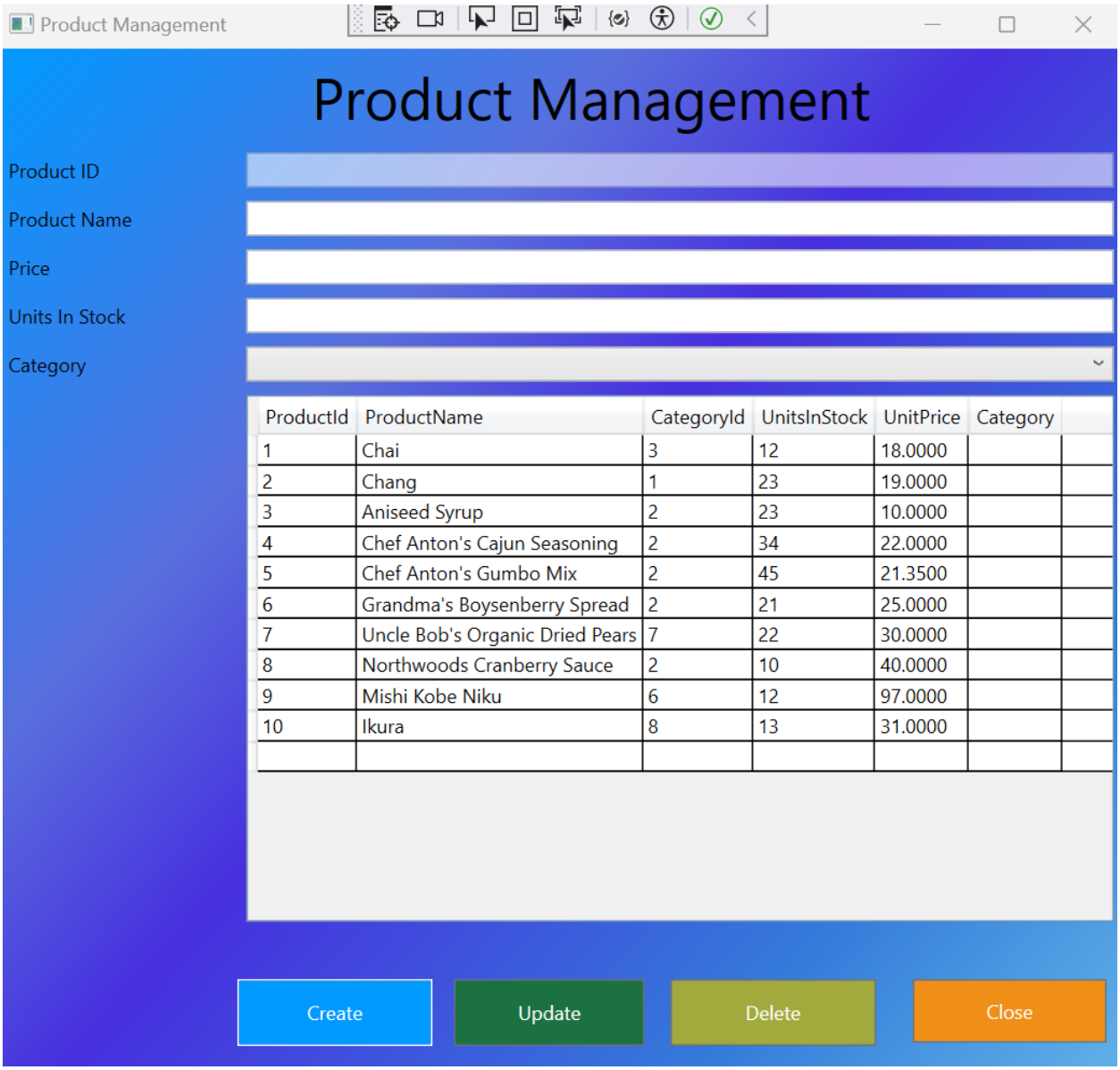
Step 06. Write codes for class **AccountService.cs** as follows:

```
1   using BusinessObjects;
2   using Repositories;
3
4   namespace Services
5   {
        2 references
6       public class AccountService : IAccountService
7       {
8           private readonly IAccountRepository iAccountRepository;
            1 reference
9           public AccountService() {
10              iAccountRepository = new AccountRepository();
11          }
            2 references
12          public AccountMember GetAccountById(string accountID)
13          {
14              return iAccountRepository.GetAccountById(accountID);
15          }
16      }
17  }
```

Activity 04: Design UI and write codes for WPF project



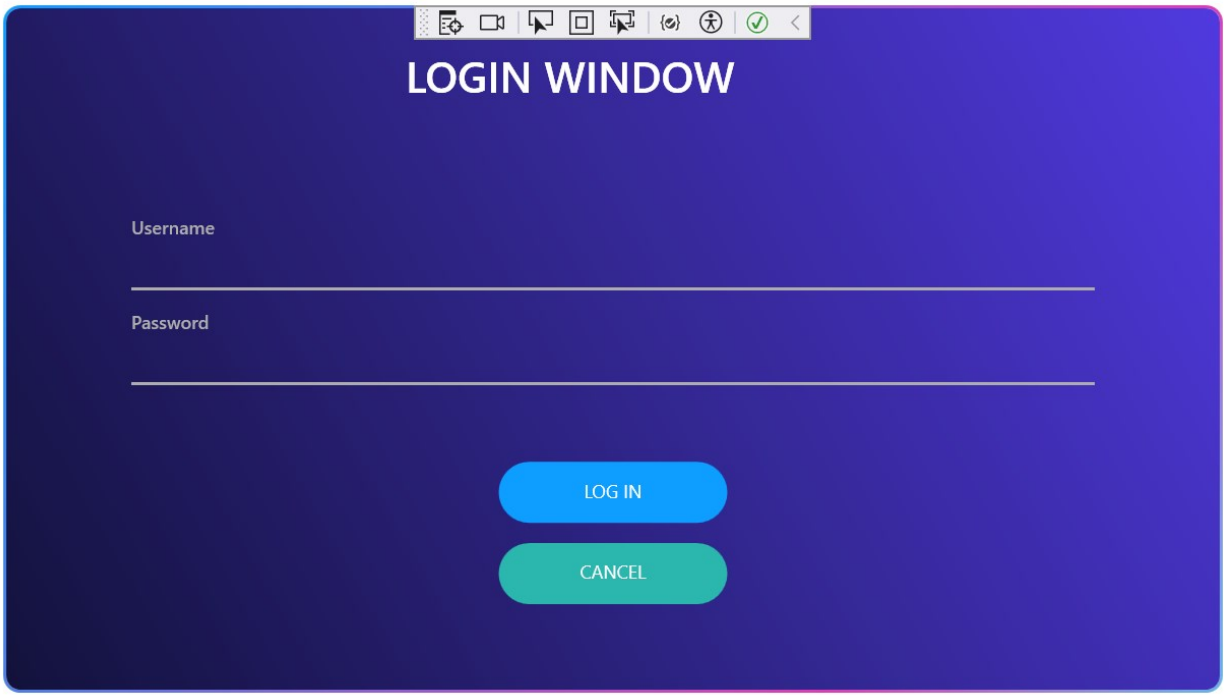
Step 01. On the **WPF** project, design UI as follows**:**



- **XAML** code for MainWindow.xaml

```
<Window x:Class="WPFApp.MainWindow"
        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:WPFApp"
        mc:Ignorable="d"
        Loaded="Window_Loaded"
        WindowStartupLocation="CenterScreen"
        Title="Product Management" Height="670" Width="710">
    <Grid>
        <Grid>
            <Grid.Background>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="#0099FF" Offset="0"/>
                    <GradientStop Color="#FF347BDA" Offset="0.794"/>
                    <GradientStop Color="#FF60B1E7" Offset="1"/>
                    <GradientStop Color="#FF596FDD" Offset="0.309"/>
                    <GradientStop Color="#FF472FDE" Offset="0.484"/>
                </LinearGradientBrush>
            </Grid.Background>
            <Grid.RowDefinitions>
                <RowDefinition Height="60"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="30"/>
                <RowDefinition Height="*"/>
                <RowDefinition Height="30"/>
            </Grid.RowDefinitions>

            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="119.415"/>
                <ColumnDefinition Width="30.585"/>
                <ColumnDefinition Width="47*"/>
                <ColumnDefinition Width="513*"/>
            </Grid.ColumnDefinitions>

            <Label x:Name="label" Content="Product Management"
Grid.Column="2" Grid.Row="0" FontSize="36" Grid.ColumnSpan="2"
HorizontalAlignment="Center" Width="466"/>

            <Label x:Name="label1" Margin ="2,2,2,2"  Content="Product ID"
Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2"/>
            <TextBox x:Name="txtProductID" Margin ="4,4,4,4" Grid.Column="2"
Grid.Row="1" Text="" TextWrapping="Wrap" Grid.ColumnSpan="2"
IsEnabled="False" />
```

```
            <Label x:Name="label2" Margin ="2,2,2,2" Grid.Column="0"
Grid.Row="2" Content="Product Name" Grid.ColumnSpan="2" />
            <TextBox x:Name="txtProductName" Margin ="4,4,4,4"
Grid.Column="2" Grid.Row="2" Text="" TextWrapping="Wrap" Grid.ColumnSpan="2"
/>

            <Label x:Name="label8" Margin ="2,2,2,2" Content="Price"
Grid.Column="0" Grid.Row="3" Grid.ColumnSpan="2"/>
            <TextBox x:Name="txtPrice"  Margin ="4,4,4,4" Grid.Column="2"
Grid.Row="3" Text="" TextWrapping="Wrap" Grid.ColumnSpan="2" />

            <Label x:Name="label3" Margin ="2,2,2,2" Content="Units In
Stock" Grid.Column="0" Grid.Row="4" Grid.ColumnSpan="2"/>
            <TextBox x:Name="txtUnitsInStock" Margin ="4,4,4,4"
Grid.Column="2" Grid.Row="4"  Text="" TextWrapping="Wrap"
Grid.ColumnSpan="2" />

            <Label x:Name="label4" Margin ="2,2,2,2" Content="Category"
Grid.Column="0" Grid.Row="5" Grid.ColumnSpan="2"/>
            <ComboBox x:Name="cboCategory" Margin ="4,4,4,4" Grid.Column="2"
Grid.Row="5" Grid.ColumnSpan="2" />


            <DataGrid x:Name="dgData" Margin ="4,4,4,63" Grid.Column="2"
Grid.Row="6" Grid.ColumnSpan="2" SelectionChanged="dgData_SelectionChanged"
/>
            <Button  x:Name="btnCreate" Content="Create"
HorizontalAlignment="Left" Margin="29,365,0,16" Grid.Row="6"
Grid.RowSpan="2" Width="121"  Background="#FF0099FF" BorderBrush="White"
Foreground="White" Grid.ColumnSpan="3" Grid.Column="1"
Click="btnCreate_Click"/>
            <Button x:Name="btnUpdate" Content="Update" Grid.Column="3"
HorizontalAlignment="Left" Margin="87,365,0,16" Grid.Row="6" Width="118"
Background="#FF1B7140" Foreground="White" Grid.RowSpan="2"
RenderTransformOrigin="0.37,0.2" Click="btnUpdate_Click"/>
            <Button x:Name="btnDelete" Content="Delete" Grid.Column="3"
HorizontalAlignment="Left" Margin="221,365,0,16" Grid.Row="6"
Grid.RowSpan="2" Width="127" Foreground="White" Background="#FFA2AA3D"
Click="btnDelete_Click"/>

            <Button x:Name="btnClose" Content="Close" Grid.Column="3"
HorizontalAlignment="Left" Margin="371,365,0,18" Grid.Row="6"
Grid.RowSpan="2" Width="120" Background="#FFEF8F18" Foreground="White"
Click="btnClose_Click"/>

        </Grid>
    </Grid>
</Window>
```




```
<Window x:Class="WPFApp.LoginWindow"
        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:WPFApp"
        mc:Ignorable="d"
        Title="LoginWindow" Height="450" Width="800"
        WindowStartupLocation="CenterScreen"
        WindowStyle="None"
        Background="Transparent"
        AllowsTransparency="True">
    <Grid>
        <Border CornerRadius="10"
        BorderThickness="2"
        Opacity="0.95">
            <Border.BorderBrush>
                <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
                    <GradientStop Color="#0099FF" Offset="0"/>
                    <GradientStop Color="#DA34AE" Offset="0.75"/>
                    <GradientStop Color="#FF60B1E7" Offset="1"/>
                    <GradientStop Color="#FF596FDD" Offset="0.309"/>
                    <GradientStop Color="#FF8C57CA" Offset="0.484"/>
                </LinearGradientBrush>
            </Border.BorderBrush>
            <Border.Background>
                <LinearGradientBrush StartPoint="0,1" EndPoint="1,0">
                    <GradientStop Color="#060531" Offset="0"/>
                    <GradientStop Color="#FF472FDE" Offset="1"/>
```

```
                </LinearGradientBrush>
            </Border.Background>

            <Grid>
                <StackPanel Orientation="Horizontal"
  HorizontalAlignment="Center"
   Height="82" VerticalAlignment="Top" Width="632">
                    <TextBlock Text="LOGIN WINDOW"
 Foreground="White"
 FontSize="28"
 FontWeight="Medium"
 FontFamily="Montserrat"
 Cursor="Hand"
 Margin="180,30,0,0" Width="377"
/>
                </StackPanel>
                <StackPanel
                        Orientation="Vertical"
                        Margin="82,102,82,68">
                    <TextBlock Text="Username"
                           Foreground="DarkGray"
                           FontSize="12"
                           FontWeight="Medium"
                           FontFamily="Montserrat"
                           Margin="0,35,0,0"/>
                    <TextBox x:Name="txtUser"
                         FontSize="13"
                         FontWeight="Medium"
                         FontFamily="Montserrat"
                         Foreground="White"
                         CaretBrush="LightGray"
                         BorderBrush="DarkGray"
                         BorderThickness="0,0,0,2"
                         Height="28"
                         VerticalContentAlignment="Center"
                         Margin="0,5,0,0"
                         >
                        <TextBox.Background>
                            <LinearGradientBrush></LinearGradientBrush>
                        </TextBox.Background>
                    </TextBox>

                    <TextBlock Text="Password"
                           Foreground="DarkGray"
                           FontSize="12"
                           FontWeight="Medium"
                           FontFamily="Montserrat"
                           Margin="0,15,0,0"/>
                    <PasswordBox x:Name="txtPass"
                         FontSize="13"
                         FontWeight="Medium"
                         FontFamily="Montserrat"
                         Foreground="White"
```

```
                        CaretBrush="LightGray"
                        BorderBrush="DarkGray"
                        BorderThickness="0,0,0,2"
                        Height="28"
                        VerticalContentAlignment="Center"
                        Margin="0,5,0,0">
                    <PasswordBox.Background>
                        <LinearGradientBrush></LinearGradientBrush>
                    </PasswordBox.Background>
                </PasswordBox>
                <Button x:Name="btnLogin"
                    BorderThickness="0"
                    Content="LOG IN"
                    Foreground="White"
                    FontSize="12"
                    FontFamily="Montserrat"
                    Cursor="Hand"
                    Margin="0,50,0,0"
                    Click="btnLogin_Click">
                    <Button.Style>
                        <Style TargetType="Button">
                            <Setter Property="Background"
Value="#0099FF"/>
                            <Style.Triggers>
                                <Trigger Property="IsMouseOver"
Value="True">
                                    <Setter Property="Background"
Value="#28AEED"/>
                                </Trigger>
                            </Style.Triggers>
                        </Style>
                    </Button.Style>
                    <Button.Template>
                        <ControlTemplate TargetType="Button">
                            <Border Width="150" Height="40"
                                CornerRadius="20"
                                Background="{TemplateBinding
Background}">
                                <ContentPresenter
VerticalAlignment="Center"
HorizontalAlignment="Center"/>
                            </Border>
                        </ControlTemplate>
                    </Button.Template>


                </Button>


            </StackPanel>
            <StackPanel>
                <Button x:Name="btnCancel"
```

```
BorderThickness="0"
Content="CANCEL"
Foreground="White"
FontSize="12"
FontFamily="Montserrat"
Cursor="Hand"
Margin="20,350,20,0"
Click="btnCancel_Click">
                        <Button.Style>
                            <Style TargetType="Button">
                                <Setter Property="Background"
Value="LightSeaGreen"/>
                                <Style.Triggers>
                                    <Trigger Property="IsMouseOver"
Value="True">
                                        <Setter Property="Background"
Value="SeaGreen"/>
                                    </Trigger>
                                </Style.Triggers>
                            </Style>
                        </Button.Style>
                        <Button.Template>
                            <ControlTemplate TargetType="Button">
                                <Border Width="150" Height="40"
            CornerRadius="20"
            Background="{TemplateBinding Background}">
                                    <ContentPresenter
VerticalAlignment="Center"
                          HorizontalAlignment="Center"/>
                                </Border>
                            </ControlTemplate>
                        </Button.Template>


                    </Button>

                </StackPanel>

            </Grid>

        </Border>
    </Grid>
</Window>
```

Step 03. Add a reference to the WPF project to Services Project

Step 04. Write codes for LoginWindow.xaml.cs:

```
using BusinessObjects;
using Services;
using System.Windows;

namespace WPFApp
{
    /// <summary> Interaction logic for LoginWindow.xaml
    2 references
    public partial class LoginWindow : Window
    {
        private readonly IAccountService iAccountService;
        0 references
        public LoginWindow()
        {
            InitializeComponent();
            iAccountService = new AccountService();
        }

        1 reference
        private void btnLogin_Click(object sender, RoutedEventArgs e)...

        1 reference
        private void btnCancel_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}
```

The details for btnLogin_Click() function:

```
private void btnLogin_Click(object sender, RoutedEventArgs e)
{
    AccountMember account = iAccountService.GetAccountById(txtUser.Text);
    if (account != null && account.MemberPassword.Equals(txtPass.Password)
        && account.MemberRole == 1)
    {
        this.Hide();
        MainWindow mainWindow = new MainWindow();
        mainWindow.Show();

    }
    else
    {
        MessageBox.Show("You are not permission !");
    }
}
```

Step 05. Write codes for MainWindow.xaml.cs:

```
using System;
using System.Windows;
using System.Windows.Controls;
using BusinessObjects;
using Services;

namespace WPFApp
{
    /// <summary> Interaction logic for MainWindow.xaml
    5 references
    public partial class MainWindow : Window
    {
        private readonly IProductService iProductService;
        private readonly ICategoryService iCategoryService;
        1 reference
        public MainWindow()...
        1 reference
        public void LoadCategoryList()...
        4 references
        public void LoadProductList()...
        1 reference
        private void Window_Loaded(object sender, RoutedEventArgs e)...
        1 reference
        private void btnCreate_Click(object sender, RoutedEventArgs e)...
        1 reference
        private void dgData_SelectionChanged(object sender, SelectionChangedEventArgs e)...
        1 reference
        private void btnClose_Click(object sender, RoutedEventArgs e)...
        1 reference
        private void btnUpdate_Click(object sender, RoutedEventArgs e)...
        1 reference
        private void btnDelete_Click(object sender, RoutedEventArgs e)...
        1 reference
        private void resetInput()...
    }
}
```

The functions in details:

```
16      public MainWindow()
17      {
18          InitializeComponent();
19          iProductService = new ProductService();
20          iCategoryService = new CategoryService();
21      }
        1 reference
22      public void LoadCategoryList()
23      {
24          try
25          {
26              var catList = iCategoryService.GetCategories();
27              cboCategory.ItemsSource = catList;
28              cboCategory.DisplayMemberPath = "CategoryName";
29              cboCategory.SelectedValuePath = "CategoryId";
30
31          }
32          catch (Exception ex)
33          {
34              MessageBox.Show(ex.Message, "Error on load list of categories");
35          }
36      }

37      public void LoadProductList()
38      {
39          try
40          {
41              var productList = iProductService.GetProducts();
42              dgData.ItemsSource = productList;
43          }
44          catch (Exception ex)
45          {
46              // MessageBox.Show(ex.Message, "Error on load list of products");
47          }
48          finally
49          {
50              resetInput();
51          }
52      }
        1 reference
53      private void Window_Loaded(object sender, RoutedEventArgs e)
54      {
55          LoadCategoryList();
56          LoadProductList();
57      }
```

```
58      private void btnCreate_Click(object sender, RoutedEventArgs e)
59      {
60          try
61          {
62              Product product = new Product();
63              product.ProductName = txtProductName.Text;
64              product.UnitPrice = Decimal.Parse(txtPrice.Text);
65              product.UnitsInStock = short.Parse(txtUnitsInStock.Text);
66              product.CategoryId = Int32.Parse(cboCategory.SelectedValue.ToString());
67              iProductService.SaveProduct(product);
68          }
69          catch (Exception ex)
70          {
71              MessageBox.Show(ex.Message);
72          }
73          finally
74          {
75              LoadProductList();
76          }
77      }

78      private void dgData_SelectionChanged(object sender, SelectionChangedEventArgs e)
79      {
80          DataGrid dataGrid = sender as DataGrid;
81          DataGridRow row =
82              (DataGridRow)dataGrid.ItemContainerGenerator
83              .ContainerFromIndex(dataGrid.SelectedIndex);
84          DataGridCell RowColumn =
85              dataGrid.Columns[0].GetCellContent(row).Parent as DataGridCell;
86          string id = ((TextBlock)RowColumn.Content).Text;
87          Product product = iProductService.GetProductById(Int32.Parse(id));
88          txtProductID.Text = product.ProductId.ToString();
89          txtProductName.Text = product.ProductName;
90          txtPrice.Text = product.UnitPrice.ToString();
91          txtUnitsInStock.Text = product.UnitsInStock.ToString();
92          cboCategory.SelectedValue = product.CategoryId;
93      }
        1 reference
94      private void btnClose_Click(object sender, RoutedEventArgs e)
95      {
96          this.Close();
97      }
```

```
98      private void btnUpdate_Click(object sender, RoutedEventArgs e)
99      {
100         try
101         {
102             if (txtProductID.Text.Length > 0)
103             {
104                 Product product = new Product();
105                 product.ProductId = Int32.Parse(txtProductID.Text);
106                 product.ProductName = txtProductName.Text;
107                 product.UnitPrice = Decimal.Parse(txtPrice.Text);
108                 product.UnitsInStock = short.Parse(txtUnitsInStock.Text);
109                 product.CategoryId = Int32.Parse(cboCategory.SelectedValue.ToString());
110                 iProductService.UpdateProduct(product);
111
112             }
113             else
114             {
115                 MessageBox.Show("You must select a Product !");
116             }
117         }
118         catch (Exception ex)
119         {
120             MessageBox.Show(ex.Message);
121         }
122         finally
123         {
124             LoadProductList();
125         }
126     }

127     private void btnDelete_Click(object sender, RoutedEventArgs e)
128     {
129         try
130         {
131             if (txtProductID.Text.Length > 0)
132             {
133                 Product product = new Product();
134                 product.ProductId = Int32.Parse(txtProductID.Text);
135                 product.ProductName = txtProductName.Text;
136                 product.UnitPrice = Decimal.Parse(txtPrice.Text);
137                 product.UnitsInStock = short.Parse(txtUnitsInStock.Text);
138                 product.CategoryId = Int32.Parse(cboCategory.SelectedValue.ToString());
139                 iProductService.DeleteProduct(product);
140
141             }
142             else
143             {
144                 MessageBox.Show("You must select a Product !");
145             }
146         }
147         catch (Exception ex)
148         {
149
150         }
151         finally
152         {
153
154             LoadProductList();
155         }
156     }
```

```
157         private void resetInput()
158         {
159             txtProductID.Text = "";
160             txtProductName.Text = "";
161             txtPrice.Text = "";
162             txtUnitsInStock.Text = "";
163             cboCategory.SelectedValue = 0;
164         }
165     }
```

Step 06. Open App.xaml and then update XAML code as follows:

```
<Application x:Class="WPFApp.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:WPFApp"
             StartupUri="LoginWindow.xaml">
    <Application.Resources>

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

Activity 05: Run the WPFApp project and test all actions