The Pokedex Backend User Documentation

Application Overview

The purpose of the Pokedex application is to allow Pokemon fans to track Pokemon that they have acquired. Information about each Pokemon will be recorded, and the user will be able to add, update, or delete a Pokemon.

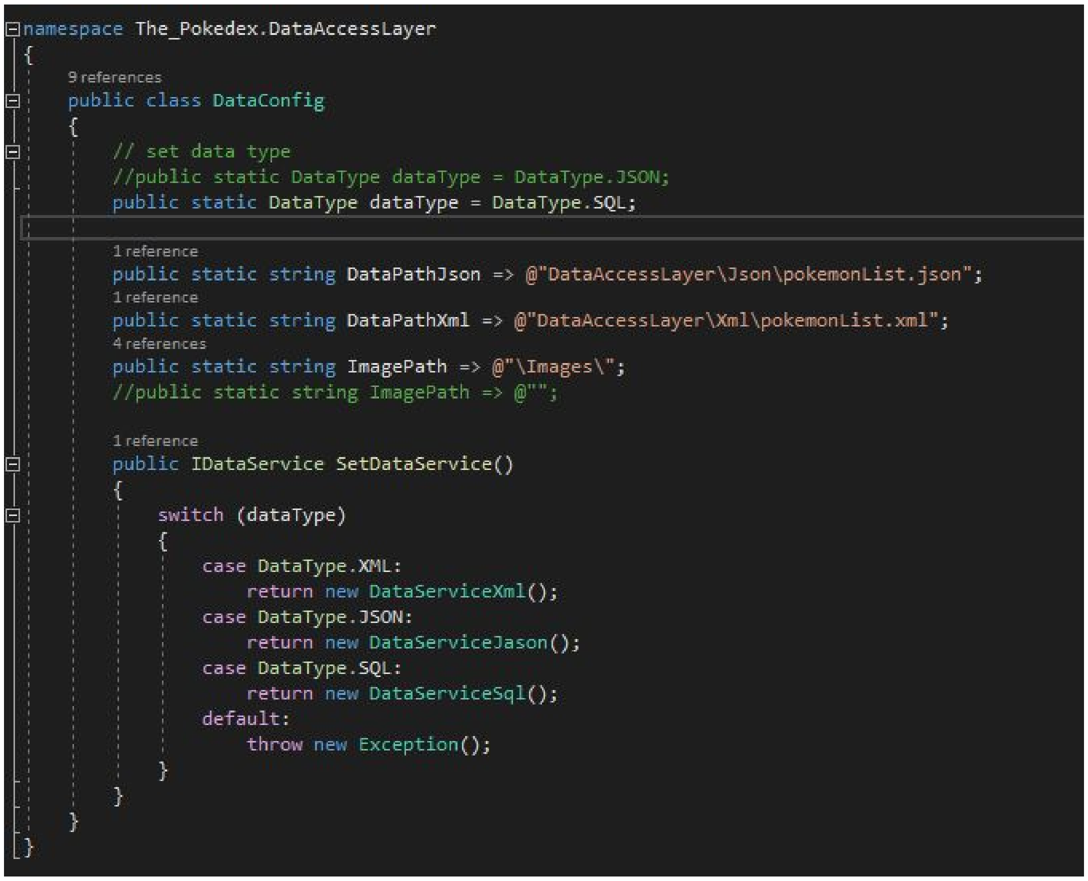
Design Pattern

This application was developed following the N-Tier architecture. The Pokedex application contains the following layers:

- BusinessLayer: Contains business logic for the application.
- DataAccessLayer: Contains classes that will allow you to interact with a SQL database, XML file, or JSON file. Also contains seed data for the application.
- Models: Consists of the Pokemon model, PokemonOperation (used in adding and editing), and FileIoMessage (for error handling).
- UtilityClass: Contains Observable object for binding and RelayCommand for the use of ICommand.
- ViewModels: Contains each team member's individual ViewModel and the Viewmodel for the add, edit, and main window.
- Views: This is where team members individual views (UI) can be found as well as the add, edit, and main window views.

Choose Persistence

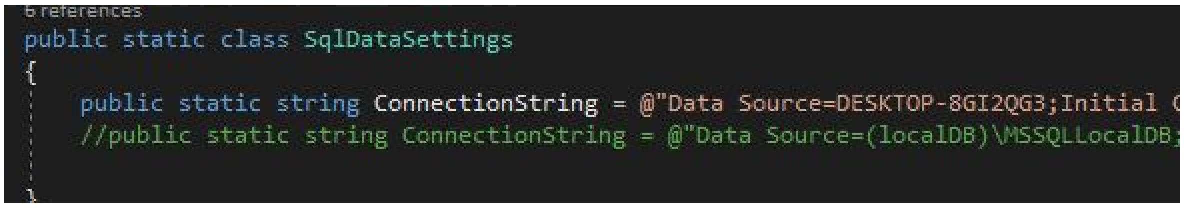
1. Navigate to DataConfig.cs in the DataAccessLayer.
2. Set the dataType to whichever persistence you wish to use.



```
namespace The_Pokedex.DataAccessLayer
{
    9 references
    public class DataConfig
    {
        // set data type
        //public static DataType dataType = DataType.JSON;
        public static DataType dataType = DataType.SQL;

        1 reference
        public static string DataPathJson => @"DataAccessLayer\Json\pokemonList.json";
        1 reference
        public static string DataPathXml => @"DataAccessLayer\Xml\pokemonList.xml";
        4 references
        public static string ImagePath => @"\Images\";
        //public static string ImagePath => @"";

        1 reference
        public IDataService SetDataService()
        {
            switch (dataType)
            {
                case DataType.XML:
                    return new DataServiceXml();
                case DataType.JSON:
                    return new DataServiceJason();
                case DataType.SQL:
                    return new DataServiceSql();
                default:
                    throw new Exception();
            }
        }
    }
}
```

Connection String (SQL)

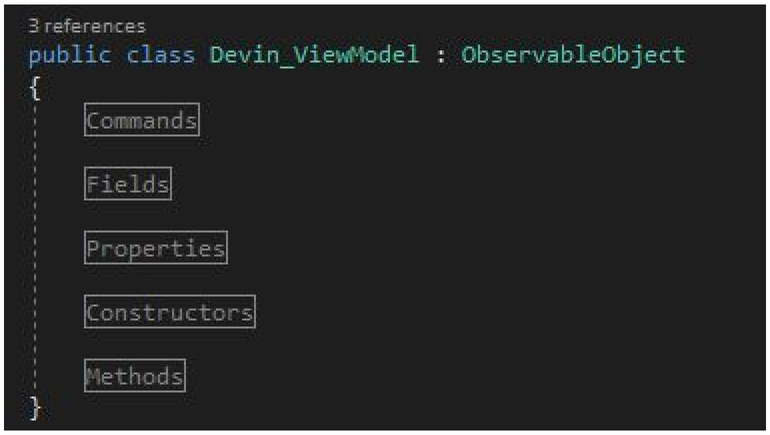
1. Navigate to SqlDataSettings.cs.
2. Set the connection string for your SQL database.



```
6 references
public static class SqlDataSettings
{
    public static string ConnectionString = @"Data Source=DESKTOP-8GI2QG3;Initial C
    //public static string ConnectionString = @"Data Source=(localDB)\MSSQLLocalDB;

}
```

Add Your ViewModel

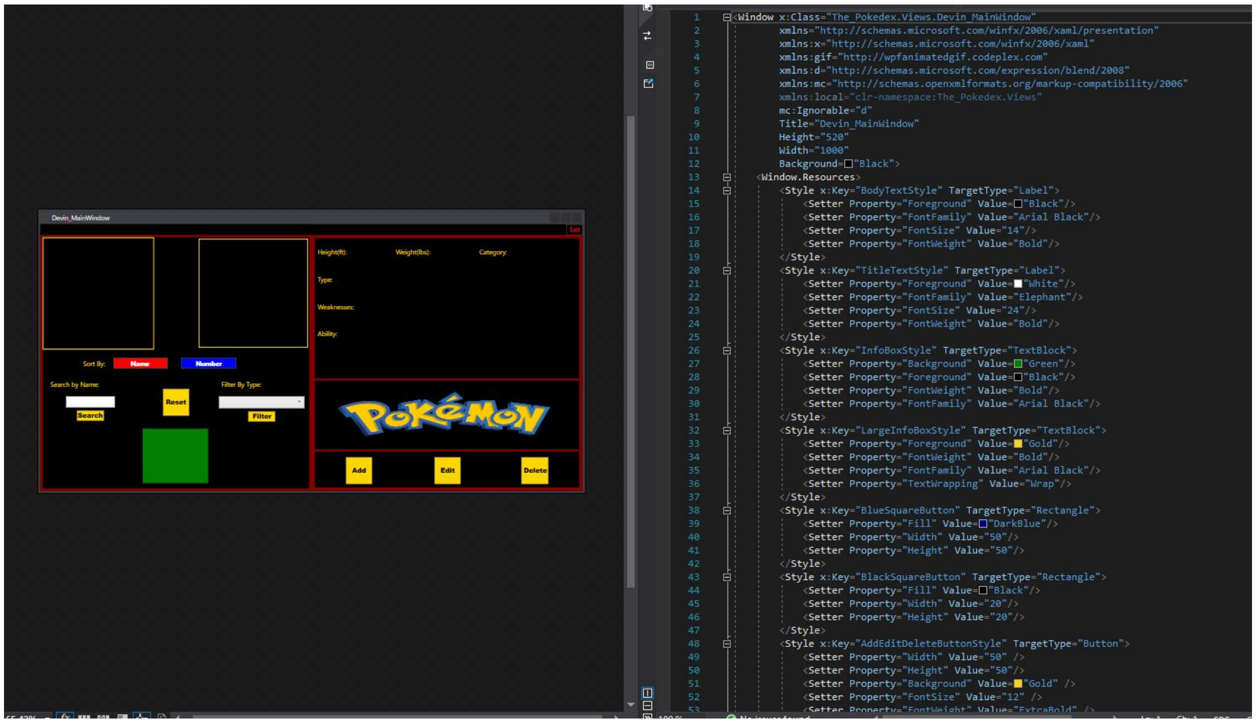
1. Add your ViewModel.cs to the Viewmodels folder.
2. Ensure that you inherit from ObservableObject.



```
3 references
public class Devin_ViewModel : ObservableObject
{
    Commands

    Fields

    Properties

    Constructors

    Methods
}
```

Add Your View

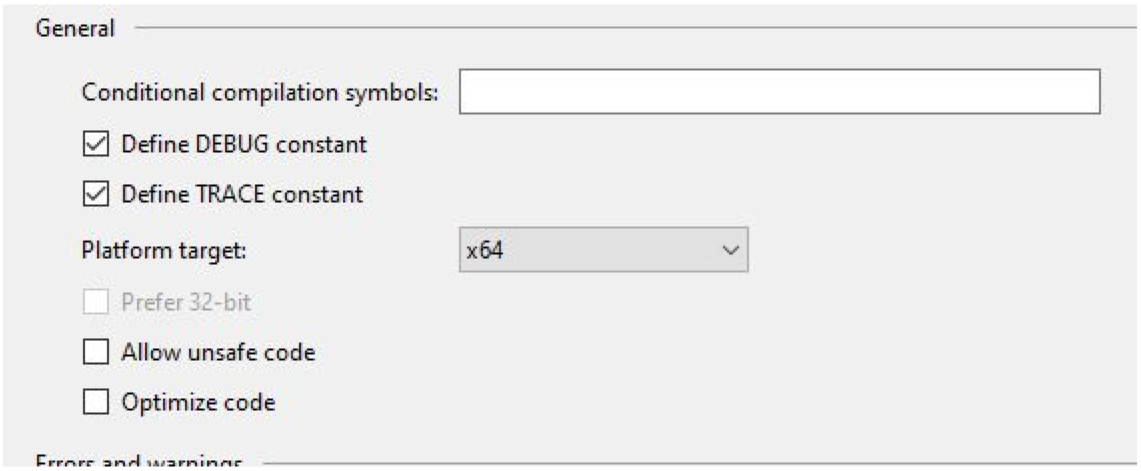
1. Add your View.xaml to the views folder.
2. Buttons are bound to ICommand.
3. Bind user inputs and displayed information to properties in the ViewModel.



```
<Window x:Class="The_Pokedex.Views.Devin_MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:gif="http://wpfanimatedgif.codeplex.com"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:The_Pokedex.Views"
        mc:Ignorable="d"
        Title="Devin_MainWindow"
        Height="520"
        Width="1000"
        Background="Black">
    <Window.Resources>
        <Style x:Key="BodyTextStyle" TargetType="Label">
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="FontFamily" Value="Arial Black"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="FontWeight" Value="Bold"/>
        </Style>
        <Style x:Key="TitleTextStyle" TargetType="Label">
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontFamily" Value="Elephant"/>
            <Setter Property="FontSize" Value="24"/>
            <Setter Property="FontWeight" Value="Bold"/>
        </Style>
        <Style x:Key="InfoBoxStyle" TargetType="TextBlock">
            <Setter Property="Background" Value="Green"/>
            <Setter Property="Foreground" Value="Black"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="FontFamily" Value="Arial Black"/>
        </Style>
        <Style x:Key="LargeInfoBoxStyle" TargetType="TextBlock">
            <Setter Property="Foreground" Value="Gold"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="FontFamily" Value="Arial Black"/>
            <Setter Property="TextWrapping" Value="Wrap"/>
        </Style>
        <Style x:Key="BlueSquareButton" TargetType="Rectangle">
            <Setter Property="Fill" Value="DarkBlue"/>
            <Setter Property="Width" Value="50"/>
            <Setter Property="Height" Value="50"/>
        </Style>
        <Style x:Key="BlackSquareButton" TargetType="Rectangle">
            <Setter Property="Fill" Value="Black"/>
            <Setter Property="Width" Value="20"/>
            <Setter Property="Height" Value="20"/>
        </Style>
        <Style x:Key="AddEditDeleteButtonStyle" TargetType="Button">
            <Setter Property="Width" Value="50" />
            <Setter Property="Height" Value="50"/>
            <Setter Property="Background" Value="Gold" />
            <Setter Property="FontSize" Value="12" />
            <Setter Property="FontWeight" Value="ExtraBold" />
```

Change Build Settings

Change the platform target to x64 in the build section of the project properties.



General

Conditional compilation symbols:

[✓] Define DEBUG constant

[✓] Define TRACE constant

Platform target: x64

[ ] Prefer 32-bit

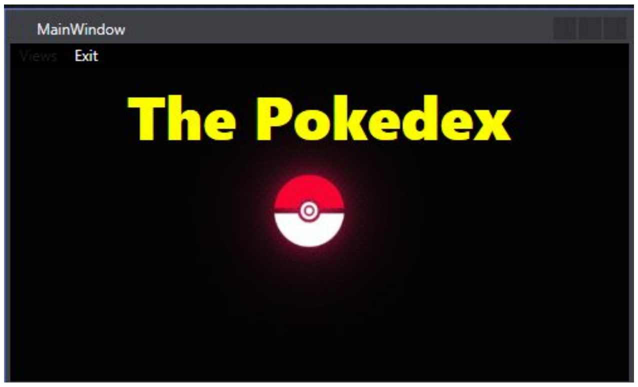
[ ] Allow unsafe code

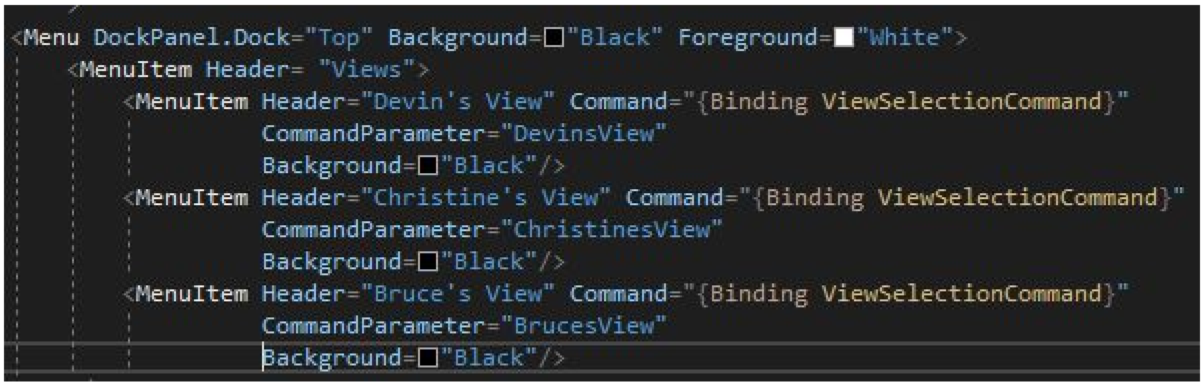
[ ] Optimize code

Errors and warnings

Update MainWindow.xaml

1. In MainWindow.xaml, add your view to the menu.
2. Ensure that you add your view name as a command parameter.

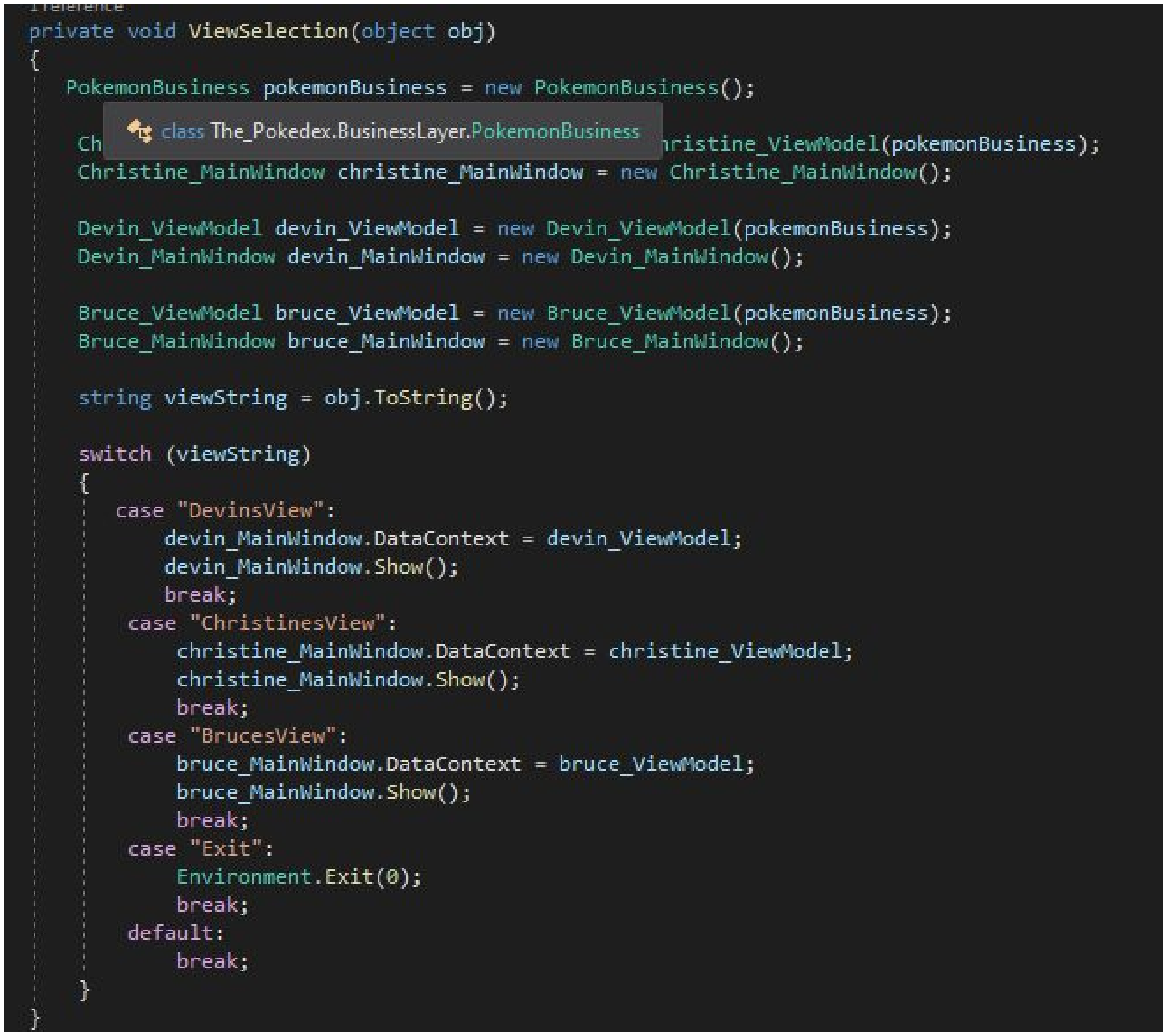




```
<Menu DockPanel.Dock="Top" Background="Black" Foreground="White">
    <MenuItem Header= "Views">
        <MenuItem Header="Devin's View" Command="{Binding ViewSelectionCommand}"
                  CommandParameter="DevinsView"
                  Background="Black"/>
        <MenuItem Header="Christine's View" Command="{Binding ViewSelectionCommand}"
                  CommandParameter="ChristinesView"
                  Background="Black"/>
        <MenuItem Header="Bruce's View" Command="{Binding ViewSelectionCommand}"
                  CommandParameter="BrucesView"
                  Background="Black"/>
```

Update the MainWindowViewModel

1. Instantiate your view and viewmodel in the ViewSelection method.
2. Add your case to the switch/case structure.



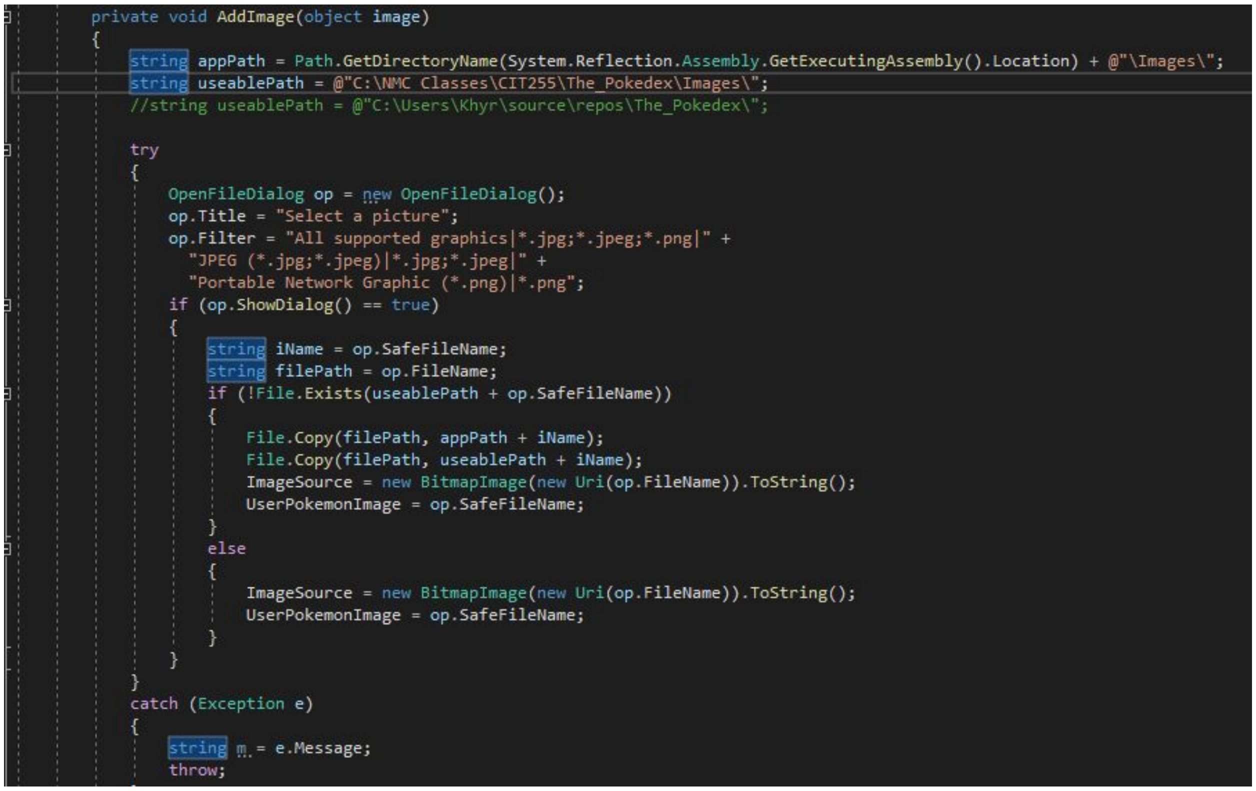
```
1 reference
private void ViewSelection(object obj)
{
    PokemonBusiness pokemonBusiness = new PokemonBusiness();
         class The_Pokedex.BusinessLayer.PokemonBusiness
     Ch                                          ristine_ViewModel(pokemonBusiness);
     Christine_MainWindow christine_MainWindow = new Christine_MainWindow();

     Devin_ViewModel devin_ViewModel = new Devin_ViewModel(pokemonBusiness);
     Devin_MainWindow devin_MainWindow = new Devin_MainWindow();

     Bruce_ViewModel bruce_ViewModel = new Bruce_ViewModel(pokemonBusiness);
     Bruce_MainWindow bruce_MainWindow = new Bruce_MainWindow();

     string viewString = obj.ToString();

     switch (viewString)
     {
        case "DevinsView":
            devin_MainWindow.DataContext = devin_ViewModel;
            devin_MainWindow.Show();
            break;
         case "ChristinesView":
             christine_MainWindow.DataContext = christine_ViewModel;
             christine_MainWindow.Show();
             break;
         case "BrucesView":
             bruce_MainWindow.DataContext = bruce_ViewModel;
             bruce_MainWindow.Show();
             break;
         case "Exit":
             Environment.Exit(0);
             break;
         default:
             break;
     }
}
```

Bugs

Images are added as a BitmapImage from the local computer, so the URI path will need to be changed to the user's computer in order for the images to load properly.

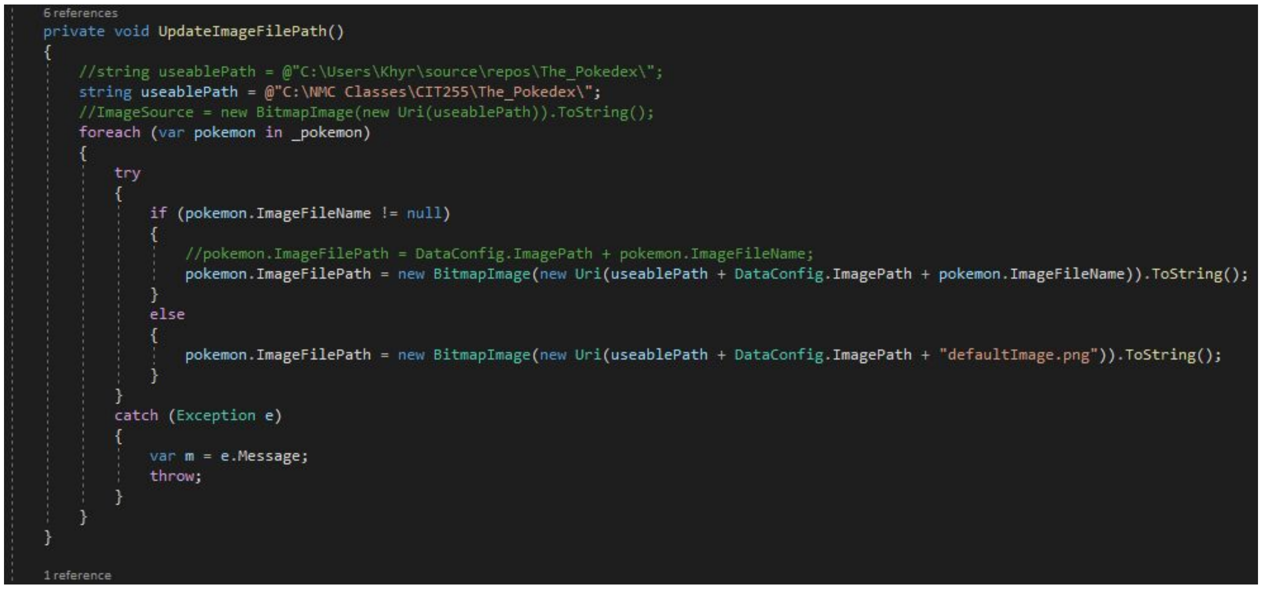
Add your own path to the following lines of code:

Change the useablePath variable In the AddImage method located in both the AddViewmodel and EditViewmodel to your local path.



```
private void AddImage(object image)
{
    string appPath = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + @"\Images\";
    string useablePath = @"C:\NMC Classes\CIT255\The_Pokedex\Images\";
    //string useablePath = @"C:\Users\Khyr\source\repos\The_Pokedex\";

    try
    {
        OpenFileDialog op = new OpenFileDialog();
        op.Title = "Select a picture";
        op.Filter = "All supported graphics|*.jpg;*.jpeg;*.png|" +
          "JPEG (*.jpg;*.jpeg)|*.jpg;*.jpeg|" +
          "Portable Network Graphic (*.png)|*.png";
        if (op.ShowDialog() == true)
        {
            string iName = op.SafeFileName;
            string filePath = op.FileName;
            if (!File.Exists(useablePath + op.SafeFileName))
            {
                File.Copy(filePath, appPath + iName);
                File.Copy(filePath, useablePath + iName);
                ImageSource = new BitmapImage(new Uri(op.FileName)).ToString();
                UserPokemonImage = op.SafeFileName;
            }
            else
            {
                ImageSource = new BitmapImage(new Uri(op.FileName)).ToString();
                UserPokemonImage = op.SafeFileName;
            }
        }
    }
    catch (Exception e)
    {
        string m = e.Message;
        throw;
```

Change the useablePath variable in the UpdateImageFilePath method in your ViewModel.



```
6 references
private void UpdateImageFilePath()
{
    //string useablePath = @"C:\Users\Khyr\source\repos\The_Pokedex\";
    string useablePath = @"C:\NMC Classes\CIT255\The_Pokedex\";
    //ImageSource = new BitmapImage(new Uri(useablePath)).ToString();
    foreach (var pokemon in _pokemon)
    {
        try
        {
            if (pokemon.ImageFileName != null)
            {
                //pokemon.ImageFilePath = DataConfig.ImagePath + pokemon.ImageFileName;
                pokemon.ImageFilePath = new BitmapImage(new Uri(useablePath + DataConfig.ImagePath + pokemon.ImageFileName)).ToString();
            }
            else
            {
                pokemon.ImageFilePath = new BitmapImage(new Uri(useablePath + DataConfig.ImagePath + "defaultImage.png")).ToString();
            }
        }
        catch (Exception e)
        {
            var m = e.Message;
            throw;
        }
    }
}

1 reference
```