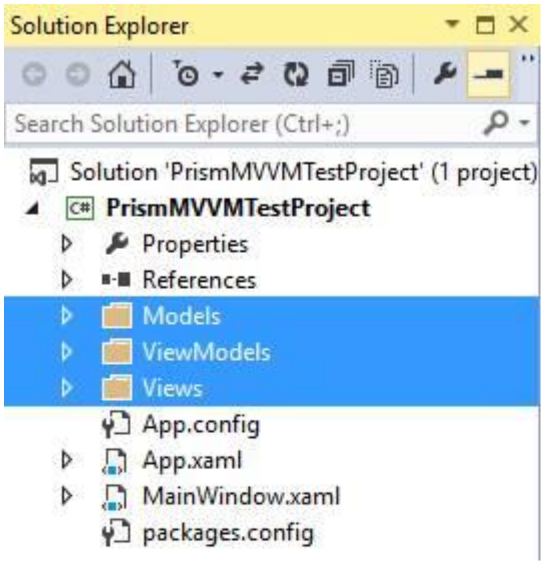
Open A Child Window From View Model In MVVM In WPF

Using Visual Studio 2013 and ‘Prism.Unity’ via nugget Packages.

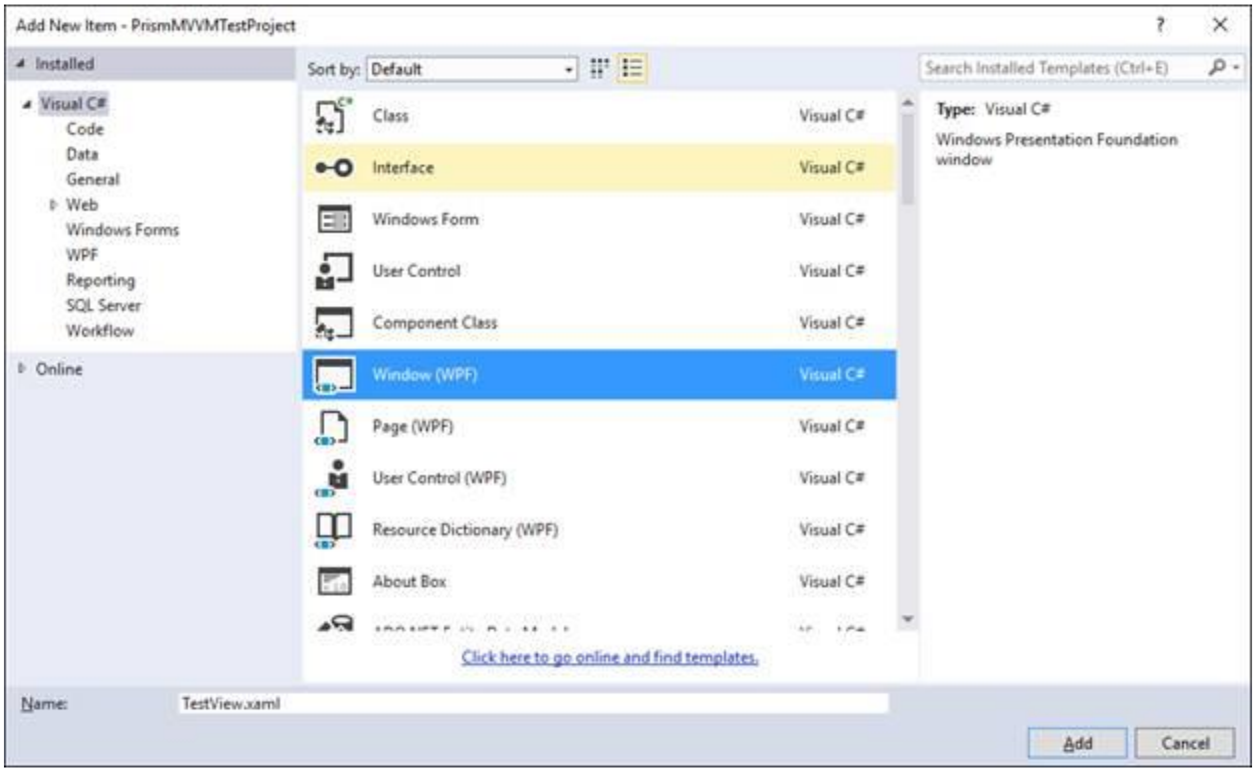
Step 1: Create a project named ‘PrismMVVMTestProject’ of WPF application.

Step 2: It’s a better approach to create the three different folders in the project for Model, View and View model respectively.



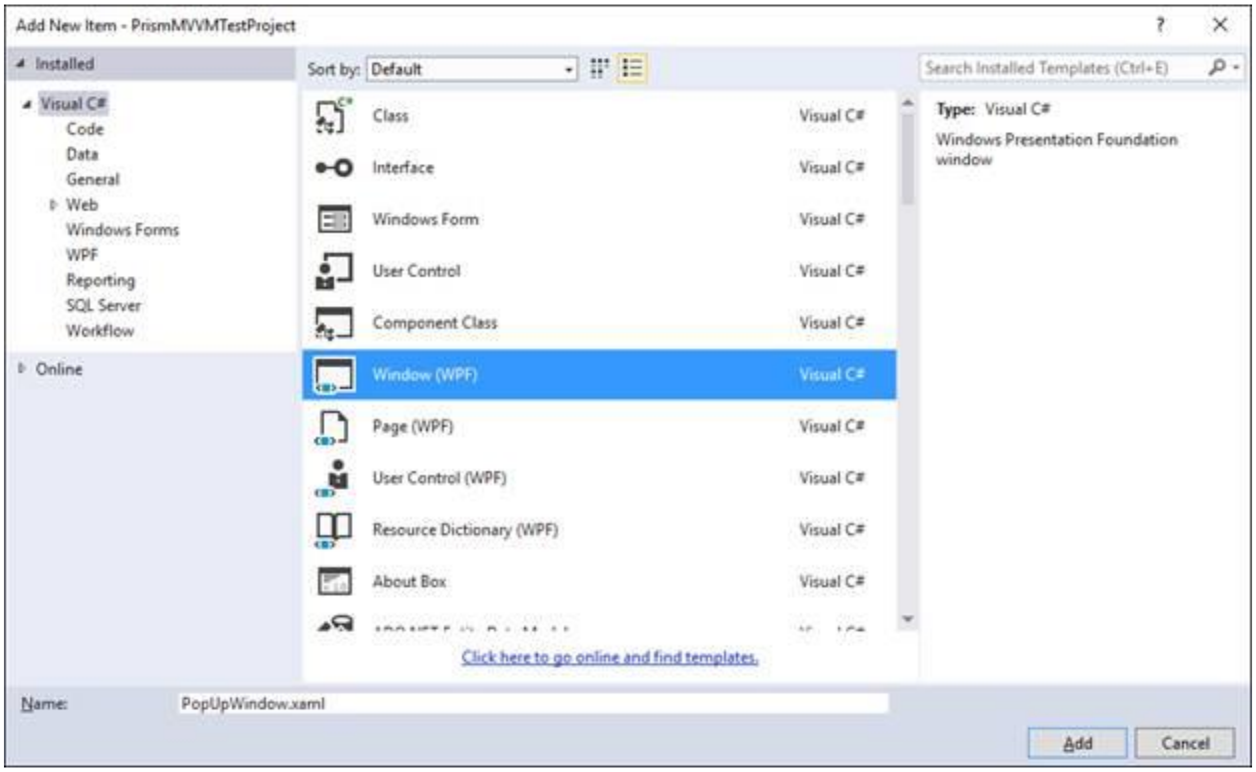
Step 3: Create pages in all folders

i. Create a View Named ‘TestView.xaml’ as a parent page in the Views folder.



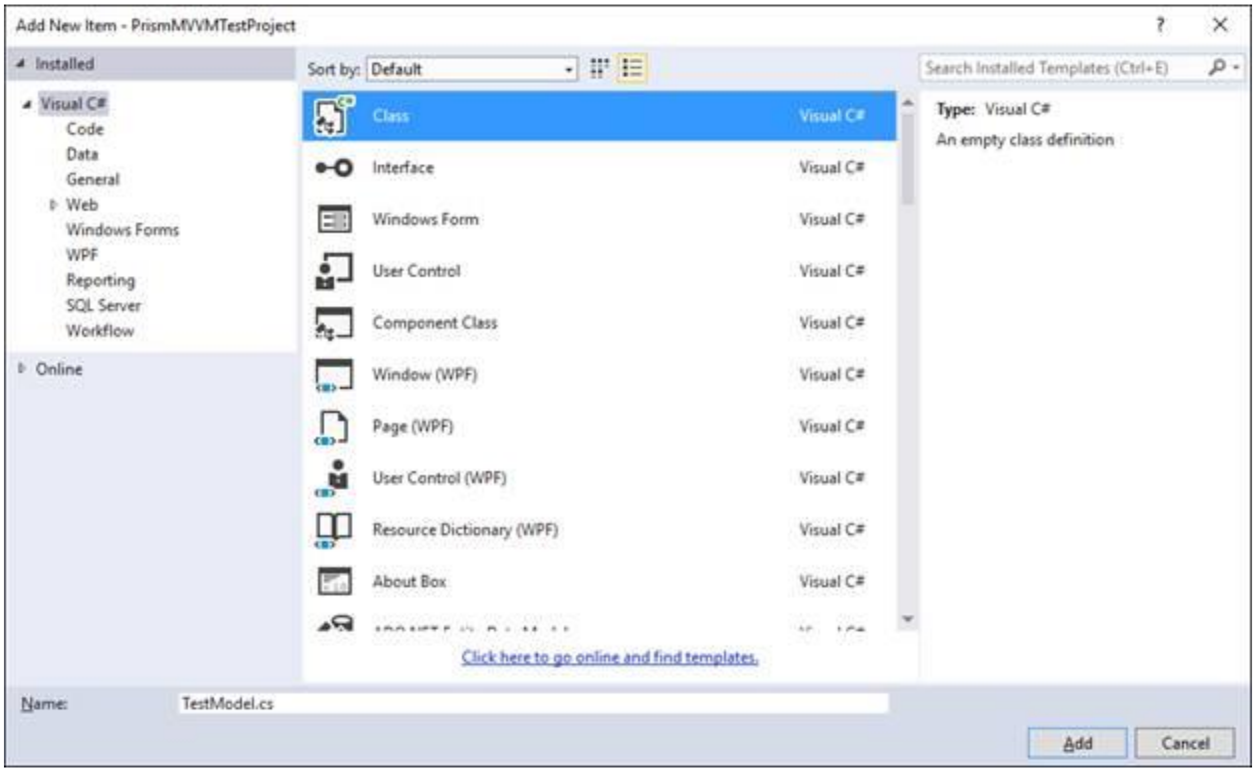
Name: TestView.xaml

ii. Also create a View Named 'PopUpWindow.xaml' as a child page in the Views folder.



Name: PopUpWindow.xaml

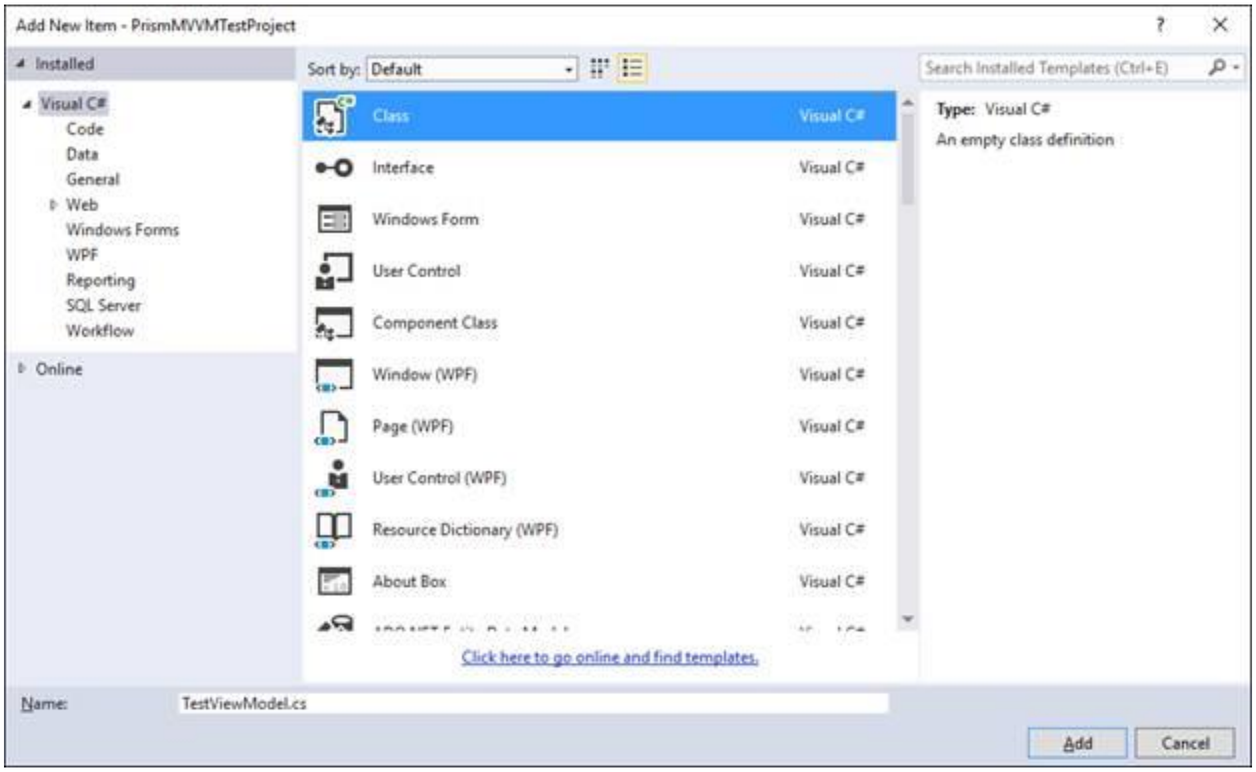
iii. Create a Model Named 'TestModel.cs' in the Models folder.



Name: TestModel.cs

Note: We don't not need Model in this article, I have just created it for the best practices.

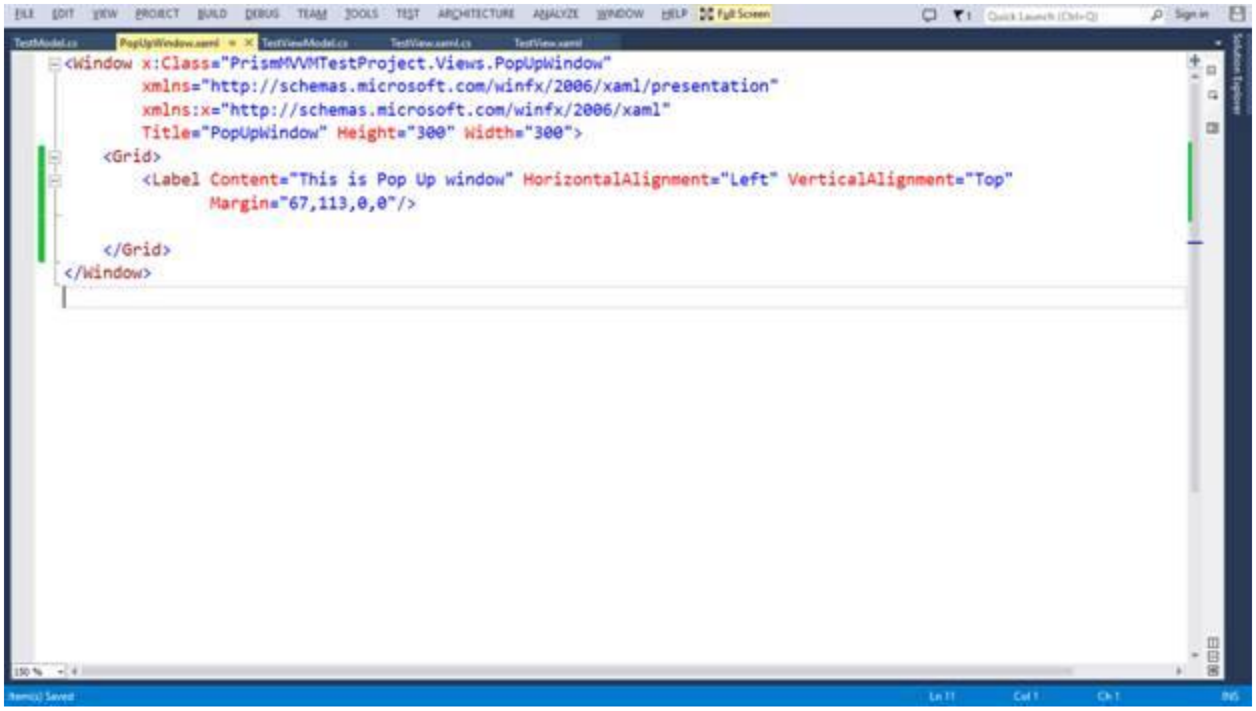
iv. Create a ViewModel Named 'TestViewModel.cs' in the ViewModels folder.



Name: TestViewModel.cs

Step 4: Add a label on the child window page named ‘PopupWindow.xaml’

```
<Label Content="This is Pop Up window" HorizontalAlignment="Left" VerticalAlignment="To
p"

Margin="67,113,0,0"/>
```



```
<Window x:Class="PrismMVVMTestProject.Views.PopUpWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="PopUpWindow" Height="300" Width="300">
    <Grid>
        <Label Content="This is Pop Up window" HorizontalAlignment="Left" VerticalAlignment="Top"
               Margin="67,113,0,0"/>

    </Grid>
</Window>
```

Step 5:

a. Add the below namespaces on the TestViewModel page,

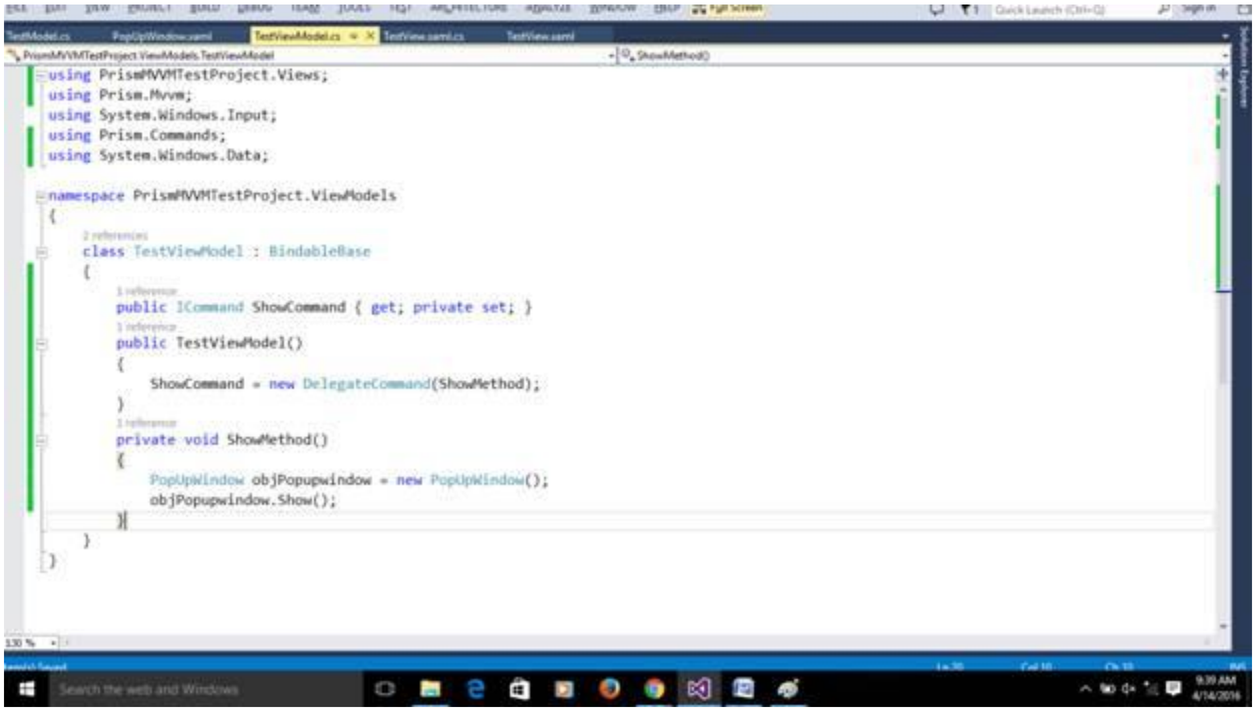
- ‘Prism.MVVM’ To inherit the class named ‘Bindable Base’.
- ‘PrismMVVMTestProject.Views’ To accessing child window in this page.
- System.Windows.Input To add ICommand.
- Prism.Commands To Use DelegateCommand.

b. Create a command named ‘ShowCommand’

c. Attach that command to the method named ‘ShowMethod’ which will act like event where will add the show the child window from the view model.

d. Create a object of child window and show it,

```
using PrismMVVMTestProject.Views;
using Prism.Mvvm;
using System.Windows.Input;
using Prism.Commands;
using System.Windows.Data;
namespace PrismMVVMTestProject.ViewModels
{
    class TestViewModel: BindableBase
    {
        public ICommand ShowCommand
        {
            get;
            private set;
        }
        public TestViewModel() {
            ShowCommand = new DelegateCommand(ShowMethod);
        }
        private void ShowMethod() {
            PopUpWindow objPopupwindow = new PopUpWindow();
            objPopupwindow.Show();
        }
    }
}
```

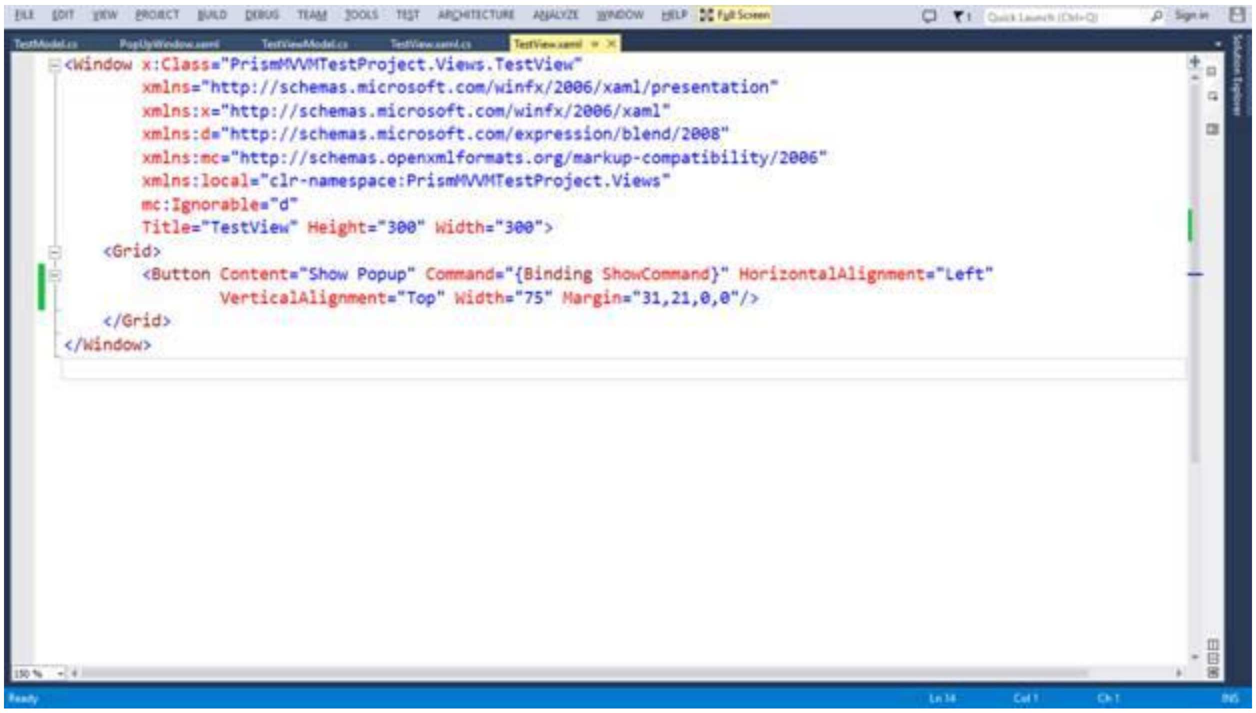


```
using PrismMVVMTestProject.Views;
using Prism.Mvvm;
using System.Windows.Input;
using Prism.Commands;
using System.Windows.Data;

namespace PrismMVVMTestProject.ViewModels
{
    class TestViewModel : BindableBase
    {
        public ICommand ShowCommand { get; private set; }
        public TestViewModel()
        {
            ShowCommand = new DelegateCommand(ShowMethod);
        }
        private void ShowMethod()
        {
            PopUpWindow objPopupwindow = new PopUpWindow();
            objPopupwindow.Show();
        }
    }
}
```

Step 6:

· Add a button to show the child window with ShowCommand property and bind it that command which we have created in view model.

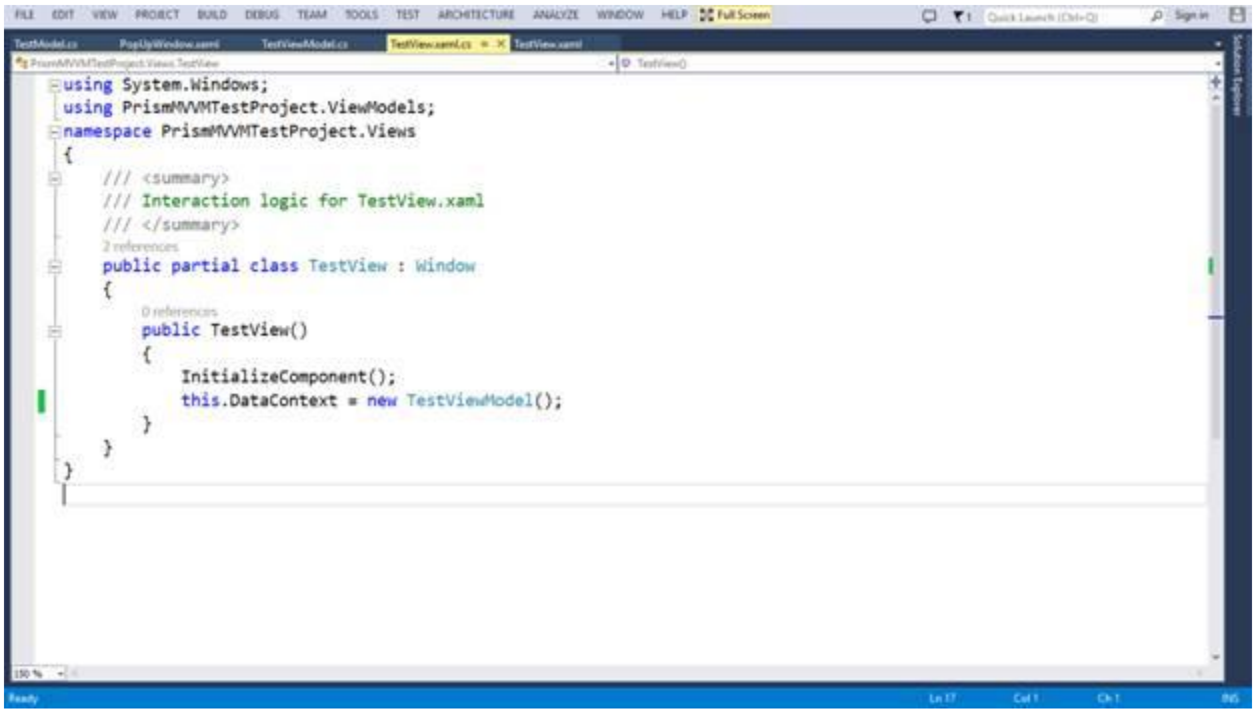
```
<Button Content="Show Popup" Command="{Binding ShowCommand}" HorizontalAlignment="Left"

VerticalAlignment="Top" Width="75" Margin="31,21,0,0"/>
```


FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP Full Screen
TestModel.cs PopUpWindow.xaml TestViewModel.cs TestView.xaml.cs TestView.xaml
Sign in
Solution Explorer
Ready

```
<Window x:Class="PrismMVVMTestProject.Views.TestView"
        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:PrismMVVMTestProject.Views"
        mc:Ignorable="d"
        Title="TestView" Height="300" Width="300">
    <Grid>
        <Button Content="Show Popup" Command="{Binding ShowCommand}" HorizontalAlignment="Left"
                VerticalAlignment="Top" Width="75" Margin="31,21,0,0"/>
    </Grid>
</Window>
```

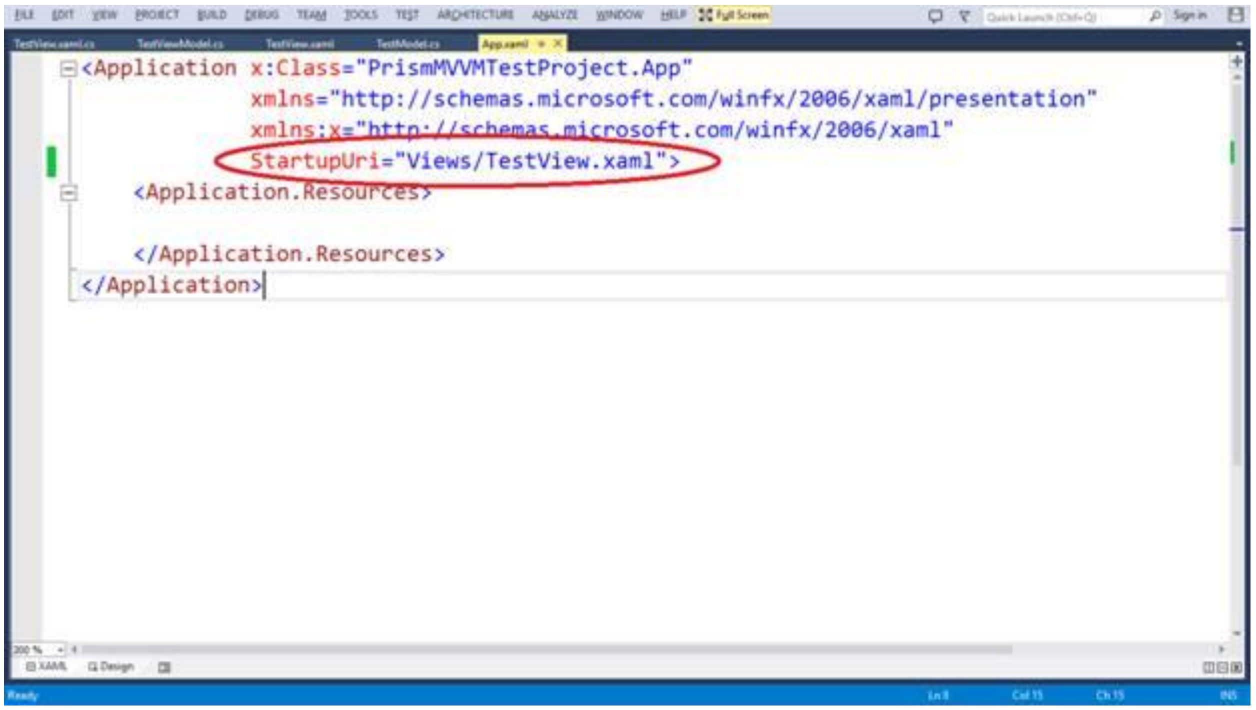
Step 7: Add ‘PrismMVVMTestProject.ViewModels’ namespace and bind Data Context of TestView Page to the ViewModel named ‘TestViewModel’.

```
using System.Windows;
using PrismMVVMTestProject.ViewModels;
namespace PrismMVVMTestProject.Views
{
    /// <summary>
    /// Interaction logic for TestView.xaml
    /// </summary>
    public partial class TestView: Window
    {
        public TestView()
        {
            InitializeComponent();
            this.DataContext = new TestViewModel();
        }
    }
}
```



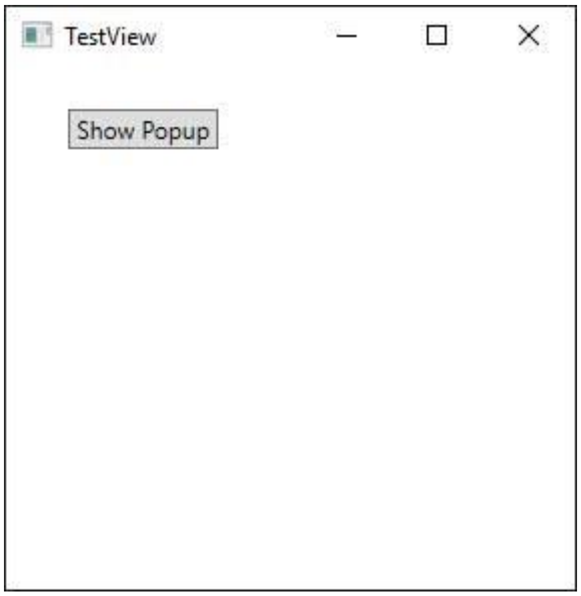
```
using System.Windows;
using PrismMVVMTestProject.ViewModels;
namespace PrismMVVMTestProject.Views
{
    /// <summary>
    /// Interaction logic for TestView.xaml
    /// </summary>
    public partial class TestView : Window
    {
        public TestView()
        {
            InitializeComponent();
            this.DataContext = new TestViewModel();
        }
    }
}
```

Step 8: Change the ‘StartupUri’ from default page ‘MainWindow’ to ‘TestView’ page,

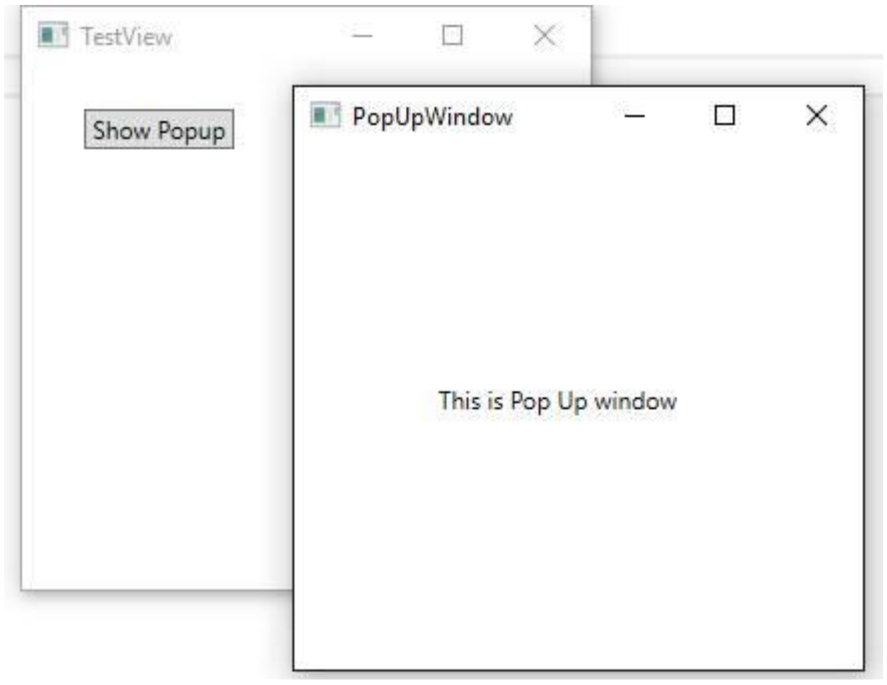


```
<Application x:Class="PrismMVVMTestProject.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="Views/TestView.xaml">
    <Application.Resources>

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

Run the page and See the Output:



After click on the ‘Show PopUp’ button, it will show the child window.


TestView
Show Popup
PopUpWindow
This is Pop Up window