Perfect. You're on the right path, and your logic is well-structured. Let's walk through your goal of having a **foolproof single-option "Shower" implementation**. I’ll provide corrections and full clean versions of your files to make sure everything runs smoothly in Revit and sets you up for scaling later.

**✅ 1. App.cs (PodDesignApp.cs)**

csharp

CopyEdit

using Autodesk.Revit.DB;

using Autodesk.Revit.UI;

using System;

using System.IO;

using System.Reflection;

using System.Windows.Media.Imaging;

using System.Windows.Media;

namespace PodDesignPlugin

{

public class PodDesignApp : IExternalApplication

{

public Result OnStartup(UIControlledApplication application)

{

string tabName = "POD Design";

try

{

application.CreateRibbonTab(tabName);

}

catch { /\* Tab might already exist, ignore \*/ }

RibbonPanel panel = application.CreateRibbonPanel(tabName, "Pod Types");

string dllPath = Assembly.GetExecutingAssembly().Location;

string iconPath = Path.Combine(Path.GetDirectoryName(dllPath), "icon.png");

PushButtonData showerBtnData = new PushButtonData(

"ShowerButton",

"Shower",

dllPath,

"PodDesignPlugin.ShowerCommand");

if (File.Exists(iconPath))

{

showerBtnData.LargeImage = new BitmapImage(new Uri(iconPath));

}

SplitButtonData splitData = new SplitButtonData("PodTypes", "Pod Types");

SplitButton split = panel.AddItem(splitData) as SplitButton;

split.AddPushButton(showerBtnData);

return Result.Succeeded;

}

public Result OnShutdown(UIControlledApplication application)

{

return Result.Succeeded;

}

}

}

**✅ 2. ShowerCommand.cs**

csharp

CopyEdit

using Autodesk.Revit.UI;

using Autodesk.Revit.DB;

using Autodesk.Revit.Attributes;

using RevitFormPlugin;

namespace PodDesignPlugin

{

[Transaction(TransactionMode.Manual)]

public class ShowerCommand : IExternalCommand

{

public Result Execute(ExternalCommandData commandData, ref string message, ElementSet elements)

{

ShowerWindow window = new ShowerWindow();

window.ShowDialog();

return Result.Succeeded;

}

}

}

**✅ 3. ShowerWindow.xaml.cs**

Make sure this file is under namespace RevitFormPlugin and the XAML has matching class in x:Class.

csharp

CopyEdit

using Autodesk.Revit.UI;

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Windows;

using System.Windows.Controls;

namespace RevitFormPlugin

{

public partial class ShowerWindow : Window

{

private Dictionary<string, string> fileMappings = new Dictionary<string, string>

{

{ "Shower\_Left\_Left", "https://www.chatgpt.com" },

{ "Shower\_Left\_Right", "https://yourwebsite.com/files/S1LRH.rvt" },

{ "Maid\_Center\_Right", "https://yourwebsite.com/files/M3CRH.rvt" }

};

public ShowerWindow()

{

InitializeComponent();

DropdownDoorOpening.SelectionChanged += DropdownDoorOpening\_SelectionChanged;

DropdownShaft.SelectionChanged += DropdownShaft\_SelectionChanged;

}

private void DropdownDoorOpening\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selectType = ((ComboBoxItem)DropdownType.SelectedItem)?.Content.ToString();

string selectedDoor = ((ComboBoxItem)DropdownDoorOpening.SelectedItem)?.Content.ToString();

ImageDoorLeft.Visibility = Visibility.Collapsed;

ImageDoorRight.Visibility = Visibility.Collapsed;

ImageDoorCenter.Visibility = Visibility.Collapsed;

if (selectedDoor == "Left" && selectType == "Shower")

ImageDoorLeft.Visibility = Visibility.Visible;

else if (selectedDoor == "Right" && selectType == "Shower")

ImageDoorRight.Visibility = Visibility.Visible;

else if (selectedDoor == "Center" && selectType == "Shower")

ImageDoorCenter.Visibility = Visibility.Visible;

}

private void DropdownShaft\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

string selectType = ((ComboBoxItem)DropdownType.SelectedItem)?.Content.ToString();

string selectedShaft = ((ComboBoxItem)DropdownShaft.SelectedItem)?.Content.ToString();

ImageShaftLeft.Visibility = Visibility.Collapsed;

ImageShaftRight.Visibility = Visibility.Collapsed;

if (selectType == "Shower")

{

if (selectedShaft == "Left")

ImageShaftLeft.Visibility = Visibility.Visible;

else if (selectedShaft == "Right")

ImageShaftRight.Visibility = Visibility.Visible;

}

}

private void DownloadButton\_Click(object sender, RoutedEventArgs e)

{

string type = (DropdownType.SelectedItem as ComboBoxItem)?.Content.ToString();

string doorOpening = (DropdownDoorOpening.SelectedItem as ComboBoxItem)?.Content.ToString();

string shaft = (DropdownShaft.SelectedItem as ComboBoxItem)?.Content.ToString();

if (string.IsNullOrEmpty(type) || string.IsNullOrEmpty(doorOpening) || string.IsNullOrEmpty(shaft))

{

TaskDialog.Show("Error", "Please select all dropdown values.");

return;

}

string key = $"{type}\_{doorOpening}\_{shaft}";

if (fileMappings.TryGetValue(key, out string fileUrl))

{

try

{

Process.Start(new ProcessStartInfo

{

FileName = fileUrl,

UseShellExecute = true

});

}

catch (Exception ex)

{

TaskDialog.Show("Error", $"Failed to open link: {ex.Message}");

}

}

else

{

TaskDialog.Show("Error", "No file mapping found for this combination.");

}

}

}

}

**✅ 4. .addin file — PodDesignPlugin.addin**

Ensure this is in:

shell

CopyEdit

%AppData%\Autodesk\Revit\Addins\2024\

xml

CopyEdit

<?xml version="1.0" encoding="utf-8" standalone="no"?>

<RevitAddIns>

<AddIn Type="Application">

<Name>PodDesignPlugin</Name>

<Assembly>C:\Path\To\Your\Build\PodDesignPlugin.dll</Assembly>

<AddInId>7D245877-7653-49DE-9495-4663AFF8B628</AddInId>

<FullClassName>PodDesignPlugin.PodDesignApp</FullClassName>

<Issuer>Sreraam Akshay</Issuer>

<VendorId>PNC Architects</VendorId>

<VendorDescription>Sreraam Akshay - Pod Design</VendorDescription>

</AddIn>

</RevitAddIns>

🔧 **Update the Assembly path** to match your compiled DLL location.

**🧠 Final Notes**

* Keep the ShowerWindow.xaml file's x:Class="RevitFormPlugin.ShowerWindow" matching the namespace + class name.
* Now that you're using PodDesignPlugin namespace for the ribbon, and RevitFormPlugin for WPF, that's totally fine as long as references are set properly in the project.
* Once this works, you can easily replicate:
  + MaidSqCommand, PowderCommand, etc.
  + Update App.cs to add those as new buttons under the same SplitButton.