HOL: Develop Universal Windows Apps on Windows 10





Contents

[Overview 4](#_Toc434234559)

[Lab 1: Writing Adaptive Code for UWP Apps 6](#_Toc434234560)

[Lab 2: Designing Adaptive Tiles 10](#_Toc434234561)

[Lab 3: Adding Interactive Toast Notifications 11](#_Toc434234562)

[Lab 4: Adding Context Menu 13](#_Toc434234563)

[Lab 5: Roam data to connect your app across devices 15](#_Toc434234564)

[Lab 6: a. Background Tasks 16](#_Toc434234565)

[Lab 6: b. Background Downloaders 18](#_Toc434234566)

[Lab 7: a. Target contracts 19](#_Toc434234567)

[Lab 7: b. Share Source 19](#_Toc434234568)

[Lab 8: Cortana Integration for your app 20](#_Toc434234569)

# Overview

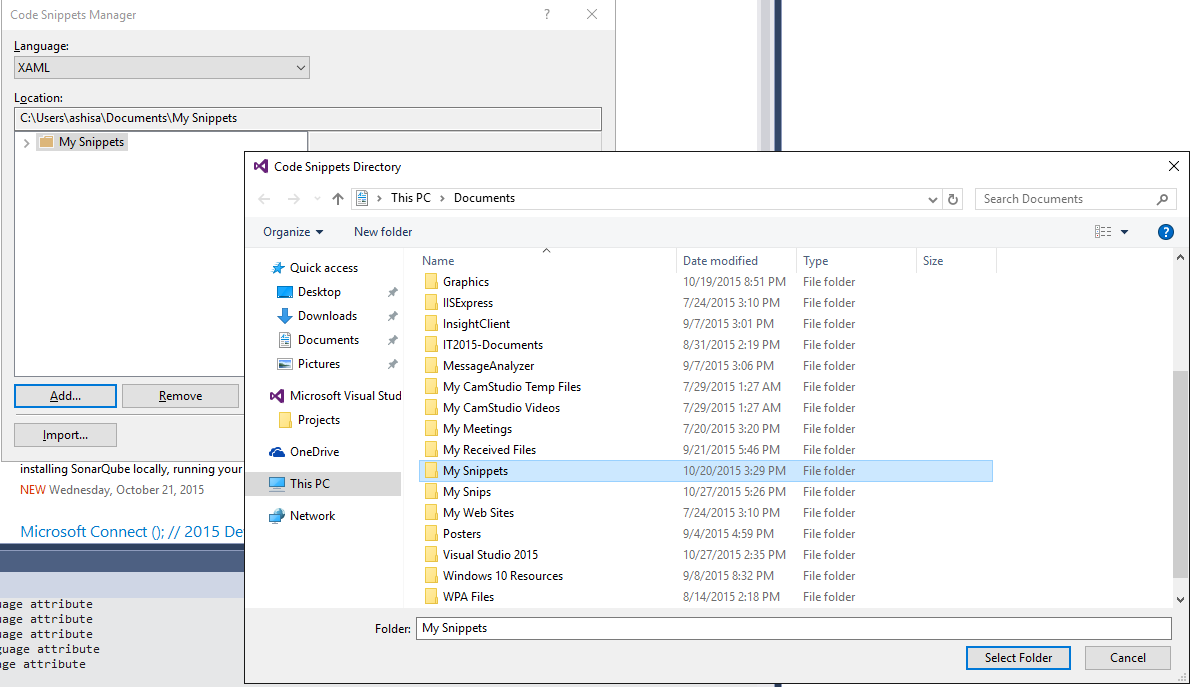
This hands-on lab builds on the content of the breakout session and will provide depth information to the participants about developing Universal Windows Apps on Windows using VS2015. The session includes plenty of demos, code samples and discussions to benefit the developers attending the session –

1. Developing Adaptive Code for UWP Apps
2. Designing Adaptive UI and tailored design
   1. Adaptive triggers
   2. SplitView control
   3. RelativePanel
3. Delighting the users
   1. Live Tiles
   2. Interactive Notifications
   3. Background Tasks & Background transfers
   4. Share Contracts
   5. Context Menu
4. Cortana Voice Commands
5. Using Application data and resources in apps

This lab will help participants develop a greater understanding of the capabilities of UWP Apps on Windows 10. They will also learn how and when they should target specific device families and what steps can they take to make sure their apps behave properly at all times. The session will also provide guidance on how to leverage the unique and delightful features on the UWP platform to keep the users engaged.

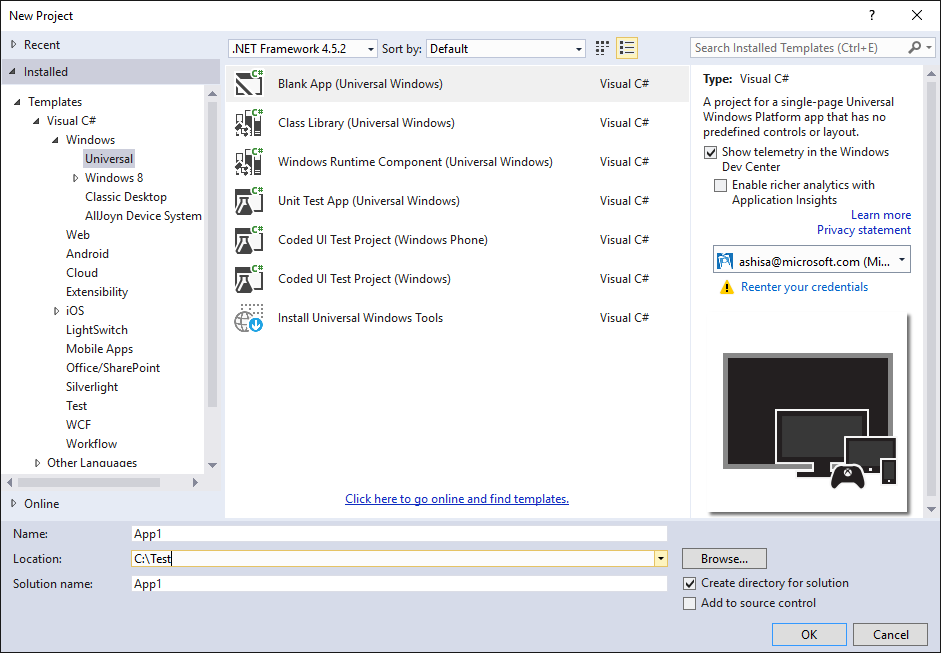
Set up:

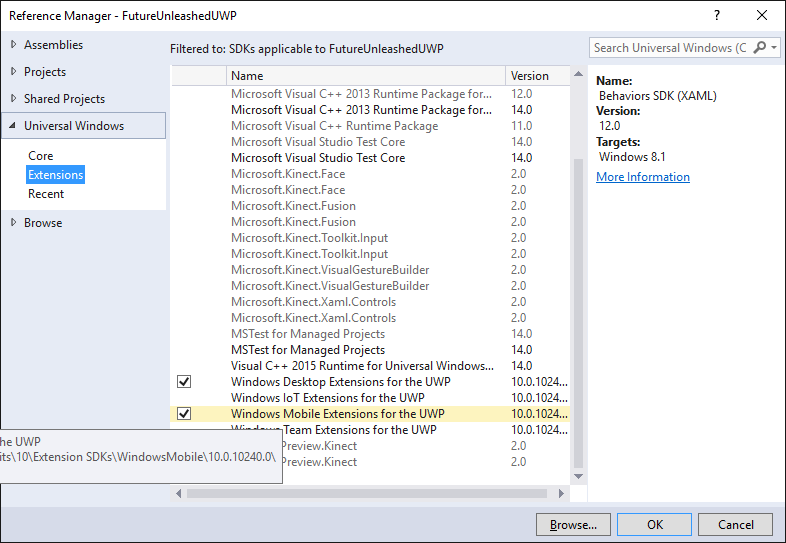
1. **Windows 10 PC** running **Visual Studio 2015** and **Windows 10 SDK**
2. Configure **VS2015** to use code snippets for this HOL –
   1. Click on **Tools** menu
   2. Click on **Code Snippets Manager**
   3. Click on **Add…**
   4. Navigate to **Documents** and create a folder called **My Snippets**
   5. Select this folder and click on **Select Folder -**



* 1. For this lab, we need all the files from here - <https://github.com/FutureUnleashed/HOL/tree/master/UWPLab>
  2. You can also download these files from this URL - <http://1drv.ms/1GXxfg5>
  3. Copy all the files in the downloaded “**My Snippets**” folder to the “**My Snippets**” under Documents folder

# Lab 1: Writing Adaptive Code for UWP Apps

1. Start **Visual Studio 2015**
2. Create a blank app -  
   
3. Click on **Assets** folder in the project, right click, click on **Add…,** click on **Existing Item…** and select all the file from the downloaded **Assets** folder
4. Open **MainPage**.**xaml**.**cs** and add the Camera Button handler in the constructor –
   1. Type **mainpagecamerabutton** and press tab **twice.** Or,
   2. Right click under the Grid control, click on **Insert snippet…,** click **My Snippets** and select “**Camera button handler in MainPage.xaml.cs**”
   3. Now, right click on the project and add references to –
      1. **Windows Desktop Extensions for the UWP**
      2. **Windows Mobile Extensions for the UWP**



* 1. Delete the line added in the step (b) above and add the adaptive code for camera button handler –
     1. Type **adaptivecamerabutton** and press tab twice. Or,
     2. Right click under the Grid control, click on **Insert snippet…** and select “**Adaptive code for camera button handler**”

1. Add the BackRequest event handler to the MainPage.xaml.cs –
   1. In the constructor, register the BackRequested event handler –
      1. Type **BackButtonEventHandler** and press tab twice. Or,
      2. Right click, click on **Insert snippet…,** click on **My Snippets** and select “**Register BackButton event handler in MainPage.xaml.cs**”
   2. Add the event handler code –
      1. Type **BackButtonFunction** and press tab twice. Or,
      2. Right click, click on **Insert snippet…,** click on **My Snippets** and select “**BackButton Event Handler Function in MainPage.xaml.cs**”
   3. Add code to enable/disable back button visibility in **App.xaml.cs** –
      1. Under the OnLaunched method in App.xaml.cs, find the line that says –

rootFrame.Navigate(typeof(MainPage), e.Arguments);  
  
Add code to toggle the visibility of the back button before this line –

* + - 1. Type **BackButtonVisibility** and press tab twice. Or,
      2. Right click, click on **Insert snippet…,** click on **My Snippets** and select “**Enable BackButton visibility in App.xaml.cs**”

1. Open **MainPage.xaml** and insert the grid and row definitions XAML snippet under the Grid control –
   1. Type **mainpagegriddef** and press tab. Or,
   2. Right click under the Grid control, click on **Insert snippet…** and select “**Row/Column definition for MainPage.xaml**”
2. Add the XAML for the page header Text Block –
   1. Type **mainpagetitletext** and press tab. Or,
   2. Right click under the Grid control, click on **Insert snippet…** and select “**Title text block for MainPage.xaml**”
3. Add Visual State Groups to the page –
   1. Type **mainpagevsgroups** and press tab. Or,
   2. Right click under the Grid control, click on **Insert snippet…** and select “**Visual State Groups for MainPage.xaml**”
4. Add Buttons on the MainPage for the rest of the labs –
   1. Type **mainpagebuttonsxaml** and press tab. Or,
   2. Right click under the Grid control, click on **Insert snippet…** and select “**HOL buttons - MainPage**”
5. In MainPage.xaml.cs, add the button click event handler and wire the page navigation etc. –
   1. Open **MainPage.xaml.cs**
   2. Add the Button Click handler after the constructor –
   3. Type **mainpagebuttonclickcode** and press tab twice. Or,
   4. Right click, click on **Insert snippet…,** click **My Snippets** and select “**Button click handler in MainPage.xaml.cs**”
6. Add a Blank XAML page named **RPanel.xaml** to the project
7. In the MainPage.xaml.cs, uncomment the following line of code –  
     
   this.Frame.Navigate(typeof(RPanel));
8. Open RPanel.xaml in designer and add the XAML to set up the page similar to the MainPage –
   1. Under the Grid control –
      1. Type **mainpagesetupxaml** and press tab. Or,
      2. Right click under the Grid control, click on **Insert snippet…** and select “**XAML Page Setup**”
9. Add the XAML code for RelativePanel
   1. Type **rpanelxaml** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**RelativePanel XAML Code**”
   3. Change the first nested **Stack Panel** in to a **Relative Panel**
   4. Move the Image and the Text Block from the second **Stack Panel** to the **Relative Panel**
   5. Remove the second **Stack Panel** now
   6. Add the Setters to the Wide visual state group using the **rpanelsetterswide** code snippet
   7. Add the Setters to the NarrowView visual state group using the **rpanelsettersnarrow** snippet

# Lab 2: Designing Adaptive Tiles

1. Open MainPage.xaml.cs and uncomment the following line –  
     
   **LaunchNVApps();**
2. Also uncomment the **LaunchNVApps()** function
3. Run the app and click on the **Tiles** button

# Lab 3: Adding Interactive Toast Notifications

1. Add a new XAML page and name it **ToastPage.xaml**
2. Open MainPage.xaml.cs and uncomment the following line –  
     
   this.Frame.Navigate(typeof(ToastPage));
3. Open ToastPage.xaml in designer and add the snippet for the page setup –
   1. Type **mainpagesetupxaml** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**HOL Buttons - Toasts**”
4. Open ToastPage.xaml.cs and after the constructor, add the button click handler –
   1. Type **toastswitchcasecode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and click on **“C# Code for Toast Switch..Case”**
5. After this button click handler function, add the code for Normal Toast –
   1. Type **normaltoastcode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for normal toasts”**
6. After this button click handler function, add the code for Toast with actions –
   1. Type **actiontoastcode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for toasts with action”**
7. After this button click handler function, add the code for toast with inputs –
   1. Type **inputtoastcode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for toasts with inputs”**
8. After this button click handler function, add the code for another toast with inputs –
   1. Type **inputtoast2code** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for toasts with inputs 2”**
9. After this button click handler function, add the code for toast with selection –
   1. Type **selecttoastcode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for toasts with selection”**
10. After this button click handler function, add the code for reminder toast –
    1. Type **remindtoastcode** and press tab twice. Or,
    2. Right click, click on **Insert snippet…** and select “**C# Code for reminder toast”**
11. After this button click handler function, add the code for a snooze toast –
    1. Type **snoozetoastcode** and press tab twice. Or,
    2. Right click, click on **Insert snippet…** and select “**C# Code for snooze and dismiss toast”**
12. After this button click handler function, add the code for another snooze toast –
    1. Type **snoozetoast2code** and press twice. Or,
    2. Right click, click on **Insert snippet…** and select “**C# Code for snooze and dismiss toast 2”**

# Lab 4: Adding Context Menu

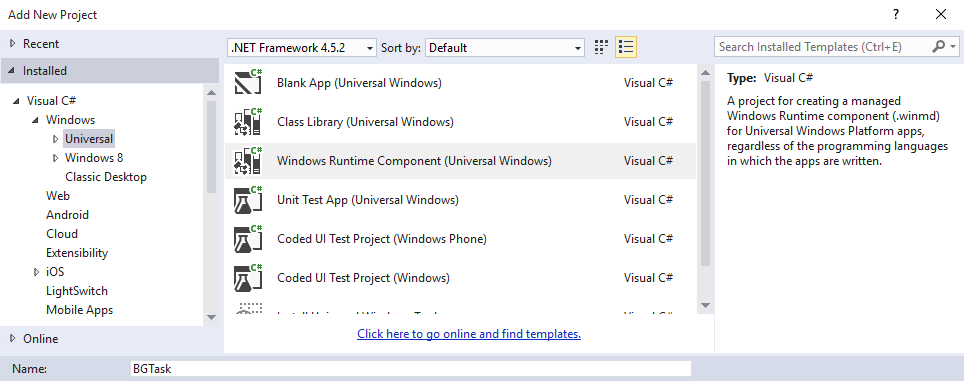
1. Open MainPage.xaml.cs and uncomment the following line –  
     
   this.Frame.Navigate(typeof(ContextMenu));
2. Add a new XAML page and name it ContextMenu.xaml
3. Open ContextMenu.xaml in designer and add the code to set up the page –
   1. Type **mainpagesetupxaml** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**XAML Page Setup**”
4. Add the code to add a variable size wrap grid and a tested stack panel –
   1. Type **contextmenupanels** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**Containers for context menu lab**”
5. Under the stack panel, add a button that has a flyout menu attached to itself
   1. Type **contextmenubutton** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**Button with a flyout menu**”
6. Open ContextMenu.xaml.cs and add button click handler after the constructor -
   1. Type **contextmenubuttoncode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for Button Flyout Event Handlers**”
7. Open ContextMenu.xaml and add an image control that has a flyout menu attached to itself
   1. Type **contextmenuimage** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**Image with a flyout menu**”
8. Just above the Grid control on this page, add the flyout resource XAML for the image flyout menus –
   1. Type contextmenupageresource and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**Flyout page resource**”
9. Open ContextMenu.xaml.cs and add image click handlers after the button click handlers-
   1. Type **contextmenuimagecode** and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for Image Flyout Event Handlers**”
10. Open ContextMenu.xaml and add a command bar in that has a flyout menu attached to itself -
    1. Type **contextmenucmdbar** and press tab. Or,
    2. Right click, click on **Insert snippet…** and select “**Command bar with a flyout menu**”
11. Open ContextMenu.xaml.cs and add the code for command bar click handlers after other event handlers -
    1. Type **contextmenucmdbarcode** and press tab twice. Or,
    2. Right click, click on **Insert snippet…** and select “**C# Code for Command Bar Flyout Event Handlers**”

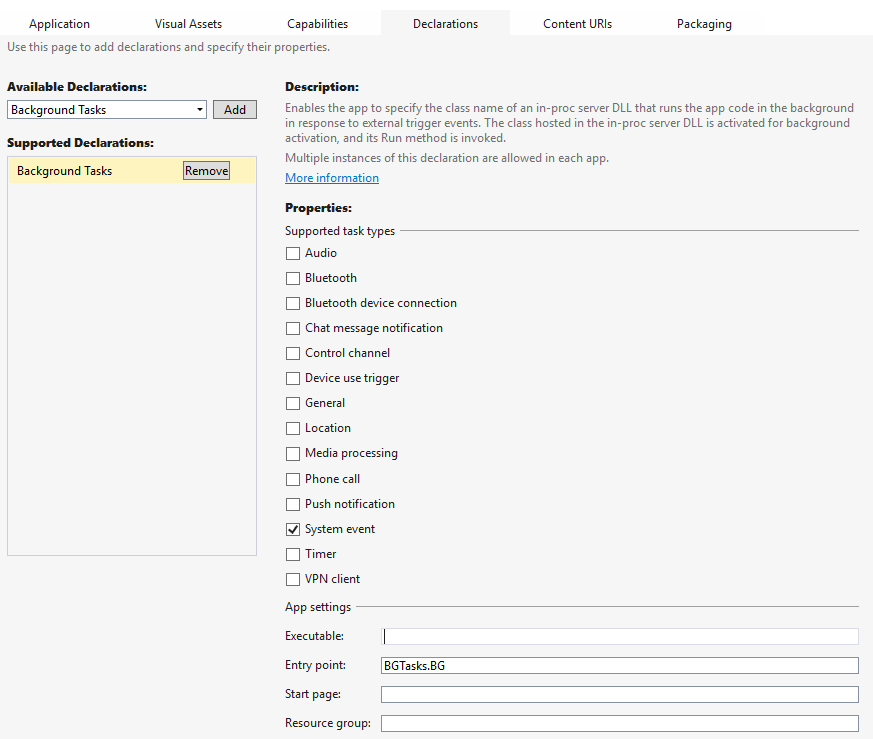
# Lab 5: Roam data to connect your app across devices

1. Add a new XAML page and name it RoamingData.xaml
2. Open MainPage.xaml.cs and uncomment the following line –  
     
   this.Frame.Navigate(typeof(RoamingData));
3. Open RoamingData.xaml in designer and add the code to set up the page –
   1. Type **mainpagesetupxaml** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**XAML Page Setup**”
4. Add the XAML for a media element –
   1. Type **roamingdatamediaelement** and press tab. Or,
   2. Right click, click on **Insert snippet…** and select “**Media Element XAML for roaming data page**”
5. Open RoamingData.xaml.cs and add the code after the constructor to handle the media element events –
   1. Type roamingdatacode and press tab twice. Or,
   2. Right click, click on **Insert snippet…** and select “**C# Code for roaming data**”

# Lab 6: a. Background Tasks

1. a. Right Click on Solution > Add > New Project > **Windows Runtime Component**

b. Give it a name – BGTasks.

1. Implement the **IBackgroundTask** interface to **BG** class. It is under the namespace Windows.ApplicationModel.Background
2. Code Snippet : **BGTaskClass**
3. Go to Package.appmanifest > Declarations > Background Tasks > Add.
4. Check **System event**
5. Enter **BGTasks.BG** in the **Entry point**
6. Right Click on the **FutureUnleashedUWP Project** > Add new item > BlankPage > name it **BGTaskPage**
7. Right click on References in **FutureUnleashedUWP** Project > Add Reference > Projects > Solution > check **BGTasks** > Ok

Code Snippet: **BGTaskPage**

1. Pin the **FutureUnleashedUWP** Project to start > **Disconnect** the computer from Network > You can now see the live tile being updated with the last access time of network.

# Lab 6: b. Background Downloaders

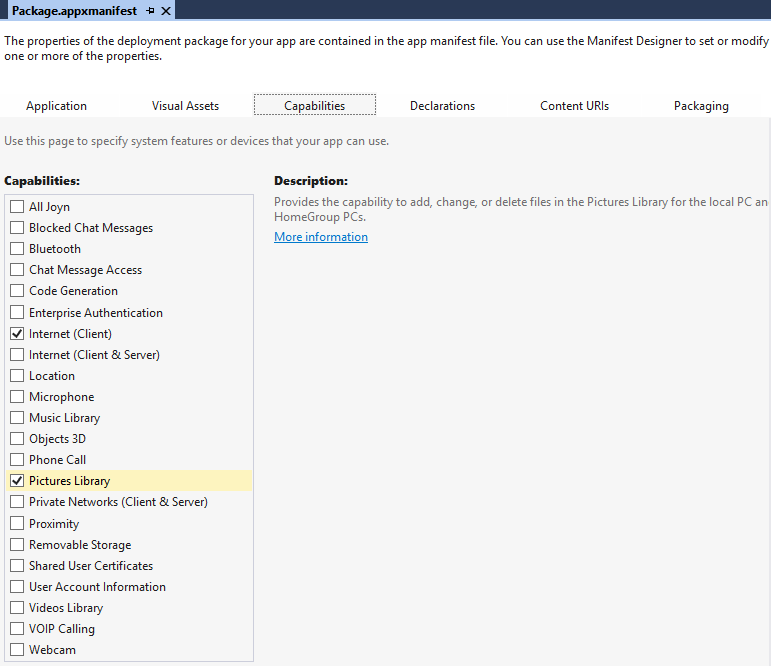
1. Right Click on **FutureUnleashedUWP** project > Add New Item > Blank Page > name it BackGroundTransfer.xaml
2. Open BackGroundTransfer.xaml

codesnippet : **BGTransferXaml**

1. a. Open **BackgroundTransfer.xaml.cs** Code Snippet: **BGTransferButton**b. Uncomment the last line

Code Snippet **– BGTransferProgress**

Create a new object of CancellationTokenSource in the constructor.

1. Go to Package.appxmanifest > Capabilities > **Picture Library**.

# Lab 7: a. Target contracts

1. Right click on FutureUnleashedUWP project > File > New > Blank Page > name it **ShareTarget.xaml**
2. a. Go to **Package.appmanifest** > Declarations > **ShareTarget** > Add
3. Add new **DataFormat** – **Text** , **html** , **storagefiles** (seperately)
4. Check **Support Any File** type checkbox
5. Go to App.xaml.cs

Code Snippet : **OnShareTargetActivated**

1. Go to **ShareTarget**.xaml  
   Code snippet : **ShareTargetXaml**
2. Go to **ShareTarget**.xaml.cs  
   Code snippet : **ShareTarget**

# Lab 7: b. Share Source

1. Right Click on FutureUnleashedUWP > Add New Item > Blank Page > Name it ShareSource.xaml
2. Open ShareSource.xaml

Code snippet : **ShareSourceXAML**

1. Open ShareSource.xaml.cs

Code snippet:

a. **ShareButtonCSharp**

b. **ShareSourceOnNavigatedTo**  
c. **ShareHandler**

# Lab 8: Cortana Integration for your app

1. Right Click on FutureunleashedUWP project > Add New Item > **XML File** > name it **VoiceCommandDefinition.xml**
2. Open Package.manifest > Capabilities > **Microphone**
3. Code Snippet : **VCD**
4. Open App.xaml.cs

Code snippet : **CortanaRegister** and **CortanaOnActivated**

1. Right click FutureunleashedUWP > Add New Item > Blank Page > Name it CortanaPage.xaml

Open CortanaPage.xaml

Code snippet : **CortanaXaml**

1. Open CortanaPage.xaml.cs

Code snippet : **CortanaPage**