Table of Contents

[**Module 01 Project Instructions** 2](#_Toc506920933)

[**Module 01 Project Step-by-Step Guide** 3](#_Toc506920934)

[**Module 02 Project Instructions** 11](#_Toc506920935)

[**Module 02 Project Step-by-Step Guide** 12](#_Toc506920936)

[**Module 03 Project Instructions** 14](#_Toc506920937)

[**Module 03 Project Step-by-Step Guide** 15](#_Toc506920938)

[**Module 04 Project Instructions** 20](#_Toc506920939)

[**Module 04 Project Step-by-Step Guide** 21](#_Toc506920940)

[**Module 05 Project Instructions** 23](#_Toc506920941)

[**Module 05 Project Step-by-Step Guide** 24](#_Toc506920942)

# **Module 01 Project Instructions**

For Module 01 of your project you will begin by installing Microsoft Visual Studio Community and taking the first steps in getting familiar with the Visual Studio solution. For this week, you will begin your development by accomplishing the following steps:

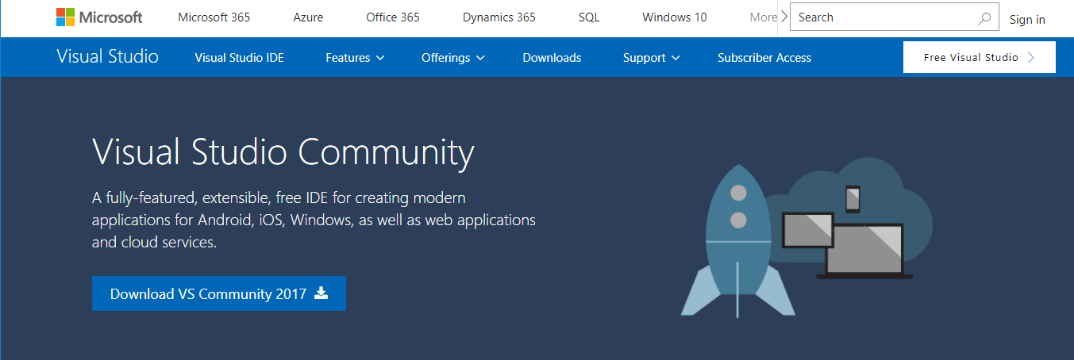
1. Install Microsoft Visual Studio Community. Of course you should customize your developer environment as you need it.
2. Create your first Universal Windows Application project solution, run the debugger, and then remove all errors. Give the project a name and save the project to an organized folder location.
3. Add a new textbox control in the properties window to use as a label for the header at the top of the application and change the default text value to 'Rasmussen College' ". Next, change the font size and the text color of the control. Specifically, you are required to:
   1. Change the name attribute of the control to "txtBoxRas" in the properties window
   2. Write C# in the codebehind file of the MainPage.xaml.cs and change the default font size of the text to "14" when the page loads
   3. Write XAML in the MainPage.xaml and change the default color of the text to "Red" when the page loads

Submit your completed assignment to the drop box below. Please check the **Course Calendar** for specific due dates.

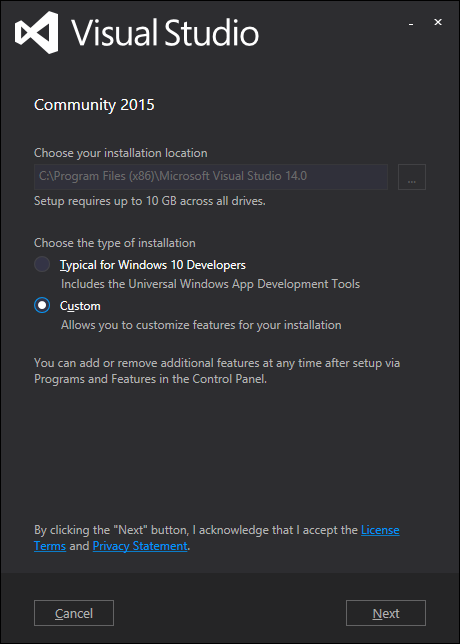
## **Module 01 Project Step-by-Step Guide**

Install Visual Studio Community

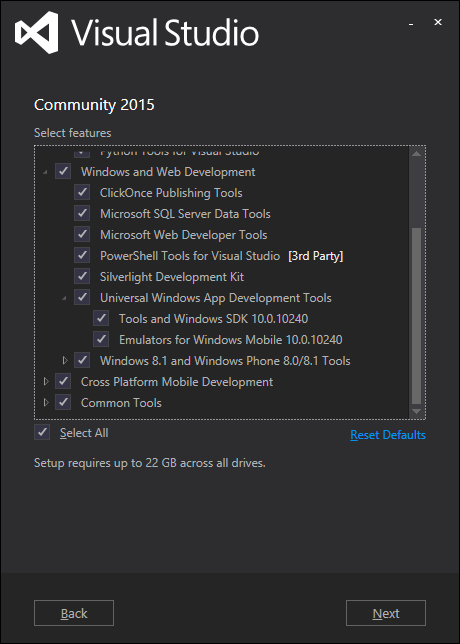
1. Go to: <https://www.visualstudio.com/vs/community/>



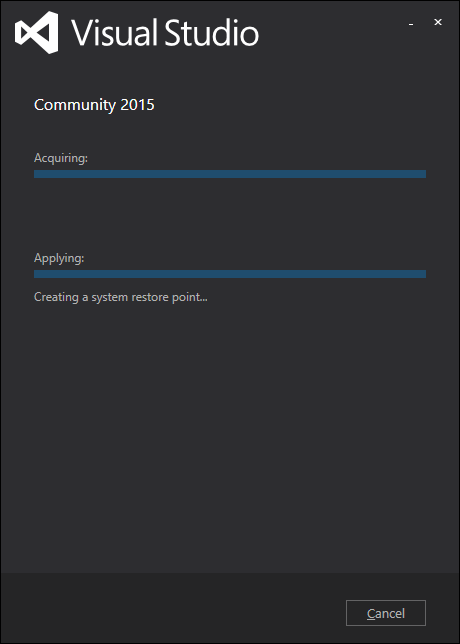
1. Click on the “Download Visual Studio” link and save the file to your desktop. Double click on the Visual Studio install file and walk through all of the default settings. Depending on the speed of your Internet services, give this install up to 1 hour to complete the install process.
2. Click on Custom then click Next



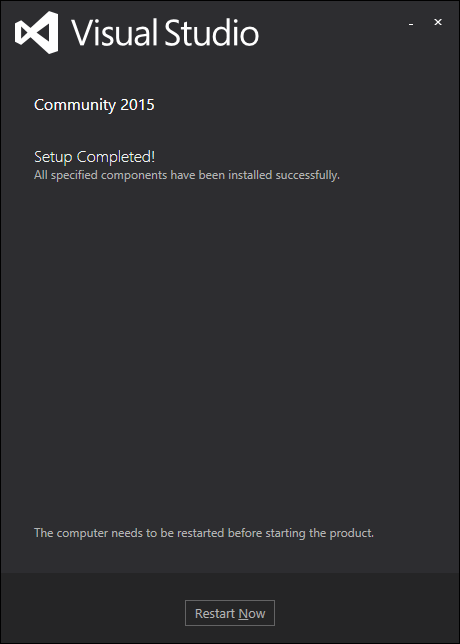
1. Click on “Select All”, you want to make sure Universal Windows is selected then click Next.



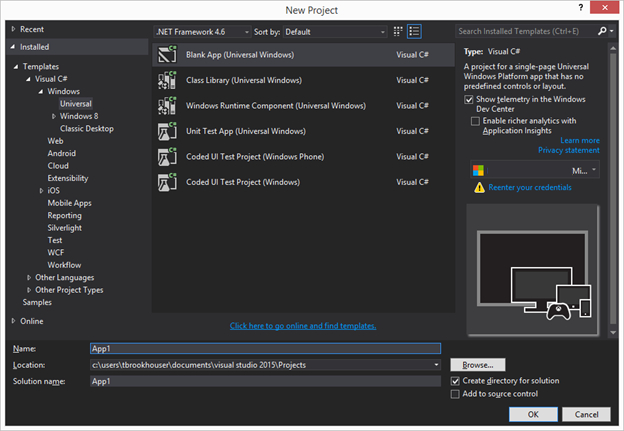
1. Let the install run until it’s complete, and this could take a while to install.



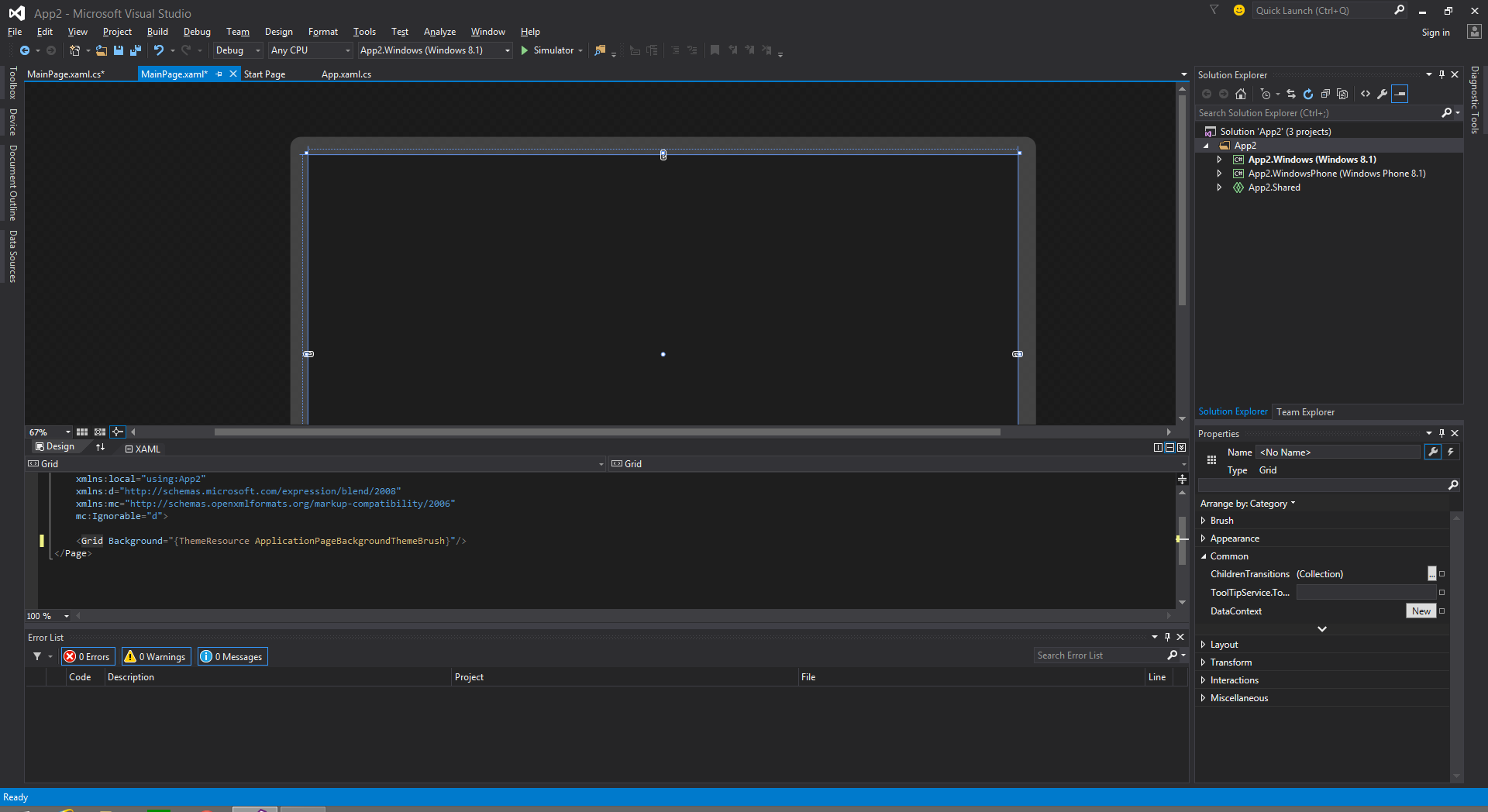
1. After the install is complete you should reboot your laptop.



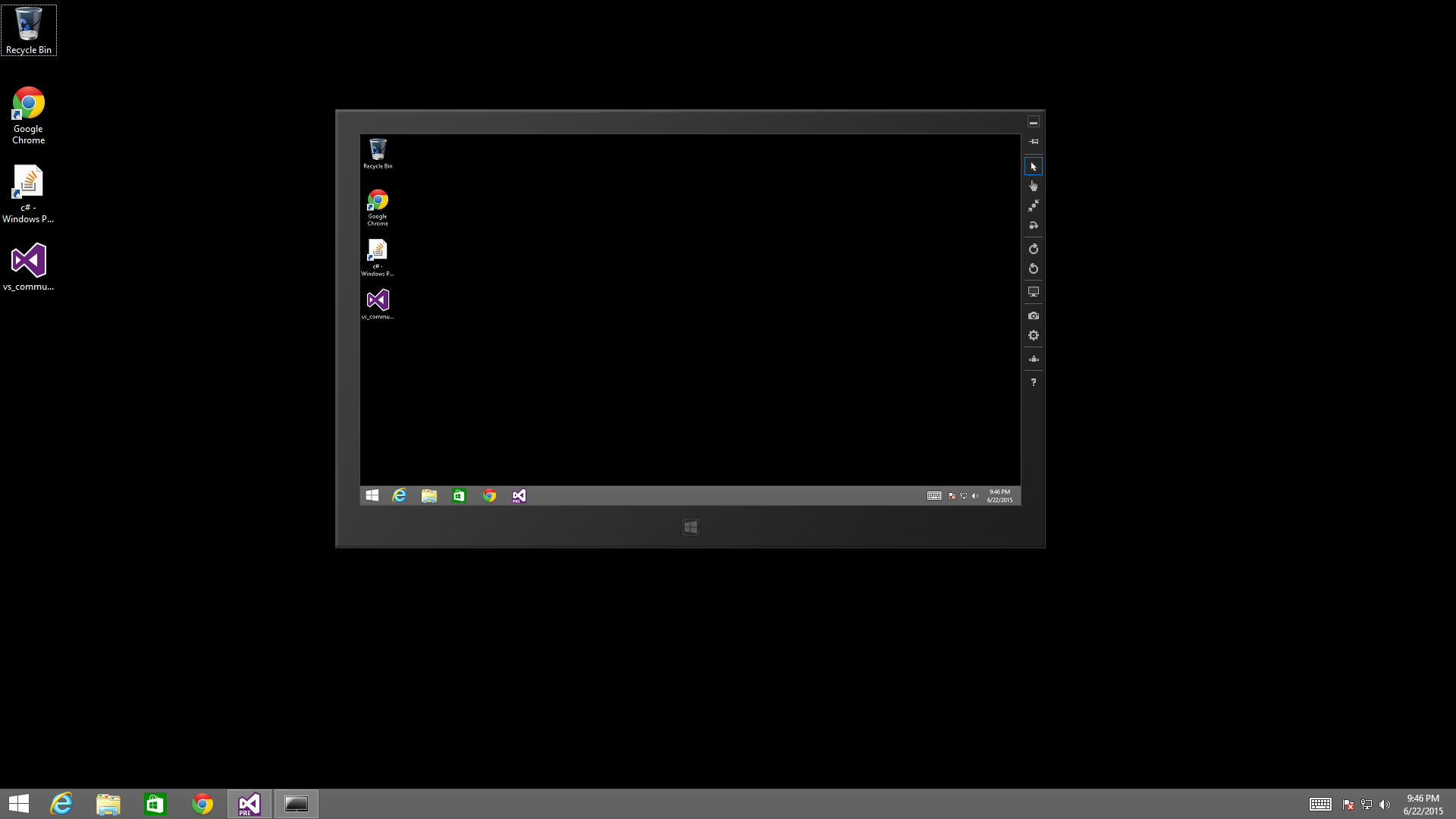
1. Start a new project in Visual Studio and go to the menu and select: New > Project. A new project window will open, if you do not see the Universal Windows templates then you will need load Windows Universal Tools in Visual Studio. In the left panel window, go to Templates > Visual C# > Windows > Universal > Dbl Click on “Blank App (Universal Windows)” > Click the OK button.



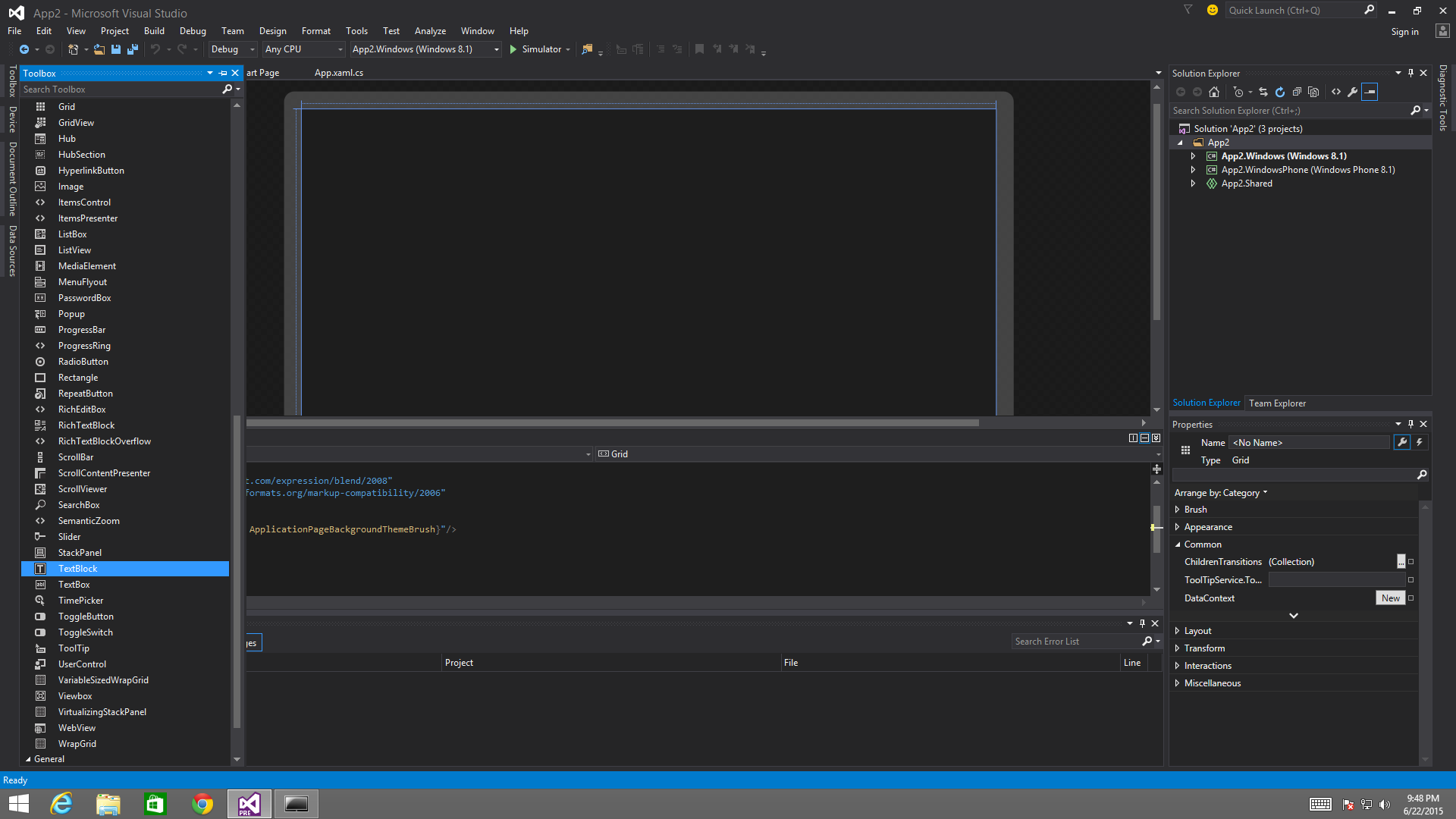
1. Go to the menu and change “local machine” to “simulator” next to the green debug arrow.



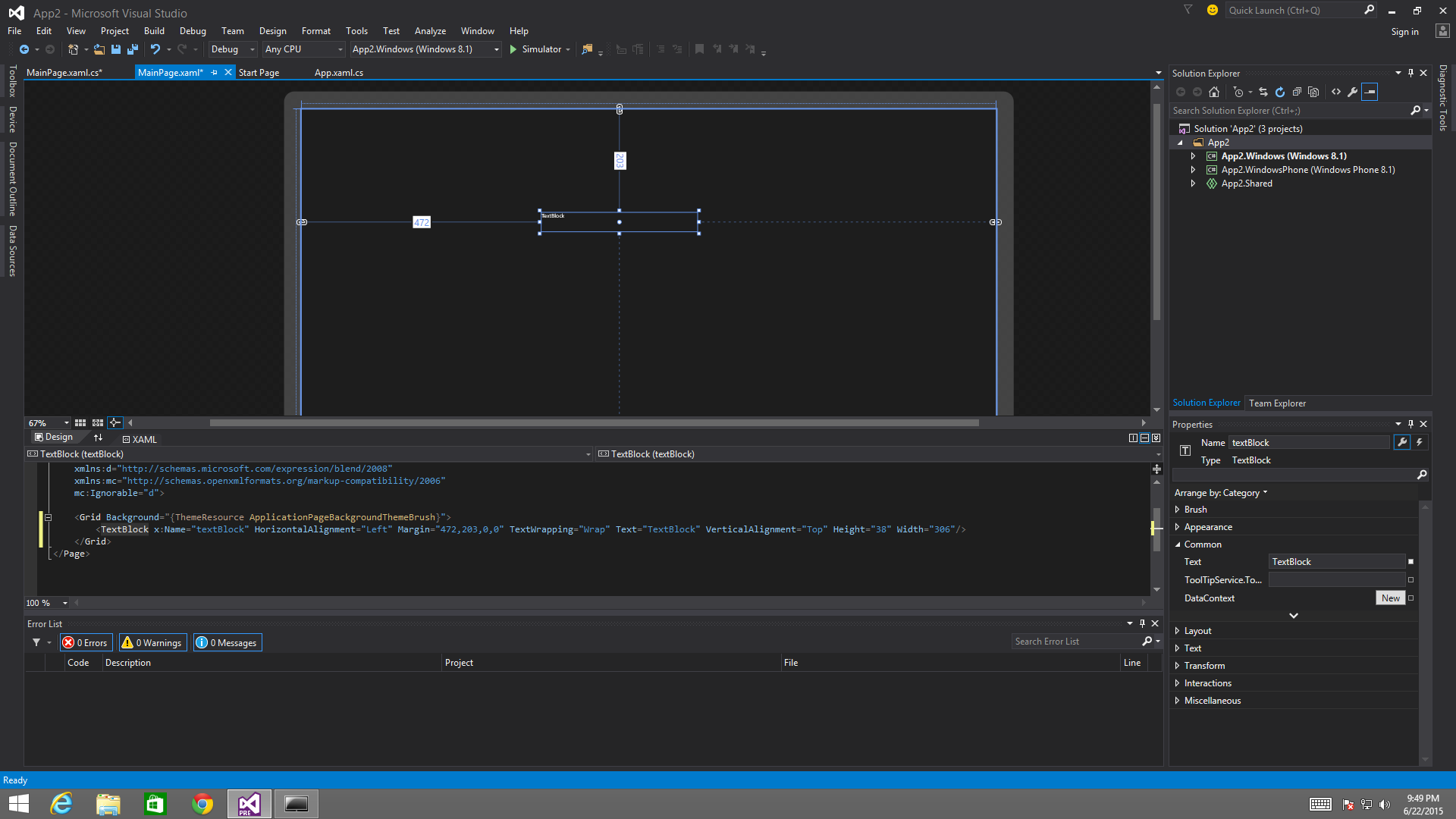
1. Go to the menu and click on Build > Build Solution or press F6 to run the Visual Studio debugger. Watch the status of the build at the bottom blue bar on the far left side.



1. Add a New UI Control to the Application and Expand the Toolbox in the far left panel and select the Textbox UI control.



1. Drag and Drop the Textbox UI control on the designer windows. Go to the right side property panel and change the control ID to “txtBoxRas”.



1. Go to the MainPage.xaml.cs file and type the following in the startup method:

XAML:

<TextBox x:Name="txtBoxRas" HorizontalAlignment="Left" Margin="118,48,0,0" Text="Hello Rasmussen College" TextWrapping="Wrap" VerticalAlignment="Top" Height="24" Width="239"/>

C#:

public void ChangeHeader()

{

txtBoxRas.Text = "Rasmussen College";

txtBoxRas.FontSize = 14;

txtBoxRas.Foreground = new SolidColorBrush(Windows.UI.Colors.Red);

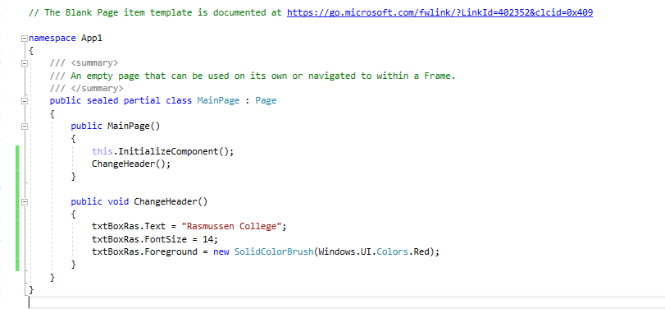
}

or

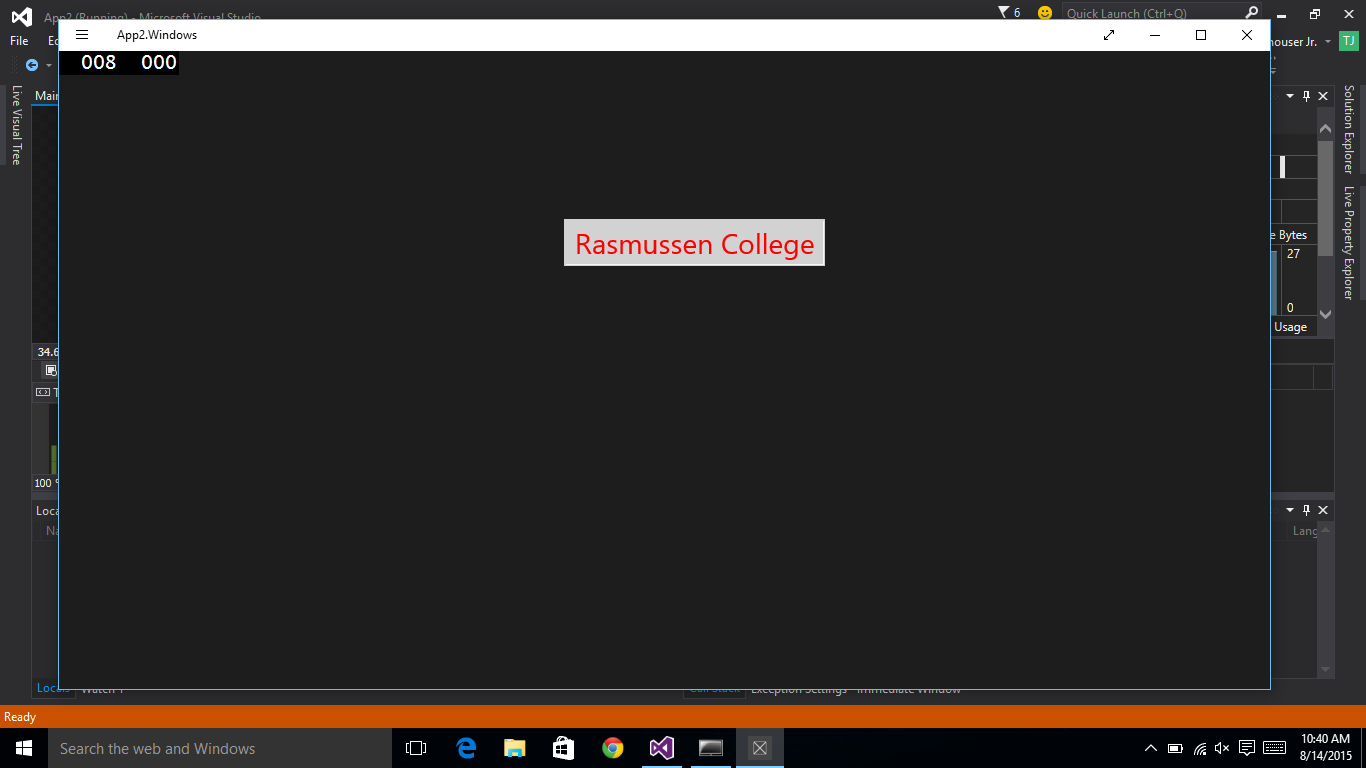
XAML:

Foreground=”#FFFD0000”

Image: Sample of MainPage.xaml.cs code in Visual Studio



1. Click on the Simulator debug arrow then Visual Studio will compile and automatically install your application in the simulator. Find you application in the simulator and run the program.



1. Build solution and test

Done.

# **Module 02 Project Instructions**

For Module 02 of your Course Project, you will build upon your Visual Studio solution and continue to develop your application. In this week and forthcoming weeks, you should feel free to alter or improve your previous code as you refine your ideas for your final application. For this week, you will continue your development by accomplishing the following steps:

1. Visual Studio offers a great way for developers to build and integrate forms. Begin your project assignment by creating a small form within the MainPage.xaml page for collecting two integer values. Then you will add textbox controls to use for labels on the form and use these controls as a means to display a calculated value back to the screen. The page you create should be easy to use, so make sure the buttons and labels are easy to understand. You might need to start with a table or some type of grid to keep your controls organized. Specifically, you are required to:
   1. Add a new textbox UI control to display the output to the screen, and change the name property to "txtBoxDisplay"
   2. Add two more textbox UI controls to enter integer values, and change the textbox name properties to "txtBoxInput1" and "txtBoxInput2"
   3. Add a button UI control so you can call the method and process a mathematical algorithm, and change the button name property to "btnSubmit"
2. Next, you will write a C# method called "ProcessCalc ()" in the MainPage.xaml.cs file that accepts two parameter values, so the parameters can be used in the mathematical algorithm. For this step you will need to make sure you have a basic understanding of encapsulation while creating the method (for more details on method encapsulation please review the Module 02 lecture notes). Specifically, you are required to:
   1. Write a new C# method called "ProcessCalc ()"
   2. Pass two parameter values into the method from the textbox controls on the form
3. Next, create two variables in the method you created in Step #2 and assign one parameter to each variable and process a mathematical algorithm (e.g. var1 + var2). Then display the correct results back to the screen. The C# compiler does not like doing math with different data types, so you might need to convert your values before you do the math unless you create the variables as the correct data type. Specifically, you are required to:
   1. Create two C# variables and assign one parameter to each variable
   2. Use the button control in your form to call the method when the button is pressed
   3. Use the two variables in the method to process a C# mathematical calculation
   4. Display the results back to the screen textbox control name attribute "txtBoxDisplay".

Submit your completed assignment to the drop box below. Please check the **Course Calendar** for specific due dates.

## **Module 02 Project Step-by-Step Guide**

1. Add one textbox in form for the display
2. Add two more textboxes in the form for user input
3. Add one button to process C# calculations
   1. Make sure you change the default value of the textboxes

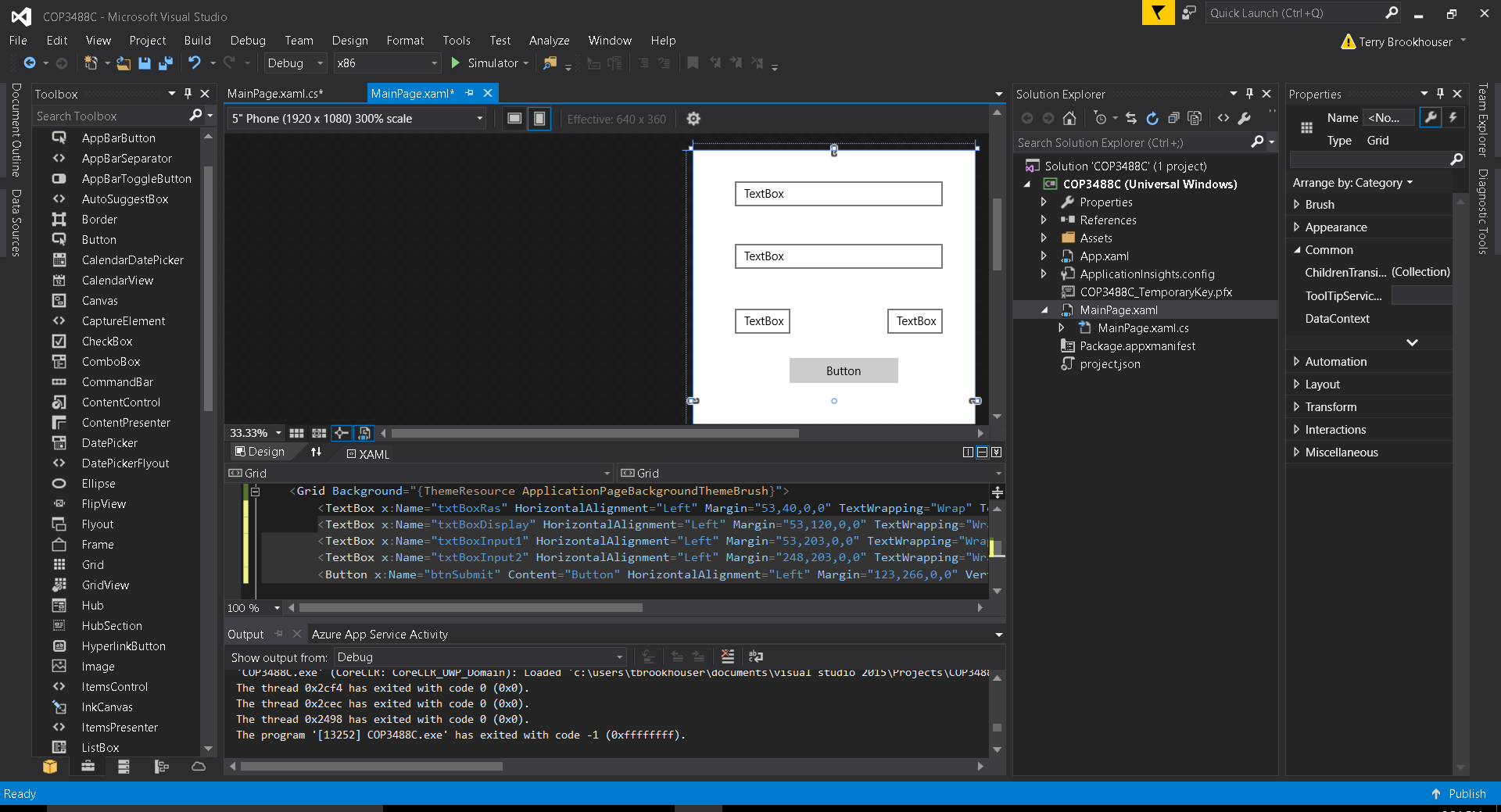
Copy & Paste C# code into the MainPage.xaml

<TextBox x:Name="txtBoxDisplay" HorizontalAlignment="Left" Margin="53,120,0,0" TextWrapping="Wrap" Text="0" VerticalAlignment="Top" Width="266"/>

<TextBox x:Name="txtBoxInput1" HorizontalAlignment="Left" Margin="53,203,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"/>

<TextBox x:Name="txtBoxInput2" HorizontalAlignment="Left" Margin="248,203,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top"/>

<Button x:Name="btnSubmit" Content="Calculate" HorizontalAlignment="Left" Margin="123,266,0,0" VerticalAlignment="Top" RenderTransformOrigin="-0.031,0.188" Width="139"/>



1. Once the button is added to the form, double click on the button to automatically create the C# event handler for the button click.

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

{

ProcessCalc();

}

1. Create a C# method called ProcessCalc ()
2. Add C# to process a simple addition calculator and display the results to the screen.

private void ProcessCalc()

{

Int32 Var1 = Convert.ToInt32(txtBoxInput1.Text) + Convert.ToInt32(txtBoxInput2.Text);

txtBoxDisplay.Text = Convert.ToString(Var1);

}

1. Build solution and test

Done.

# **Module 03 Project Instructions**

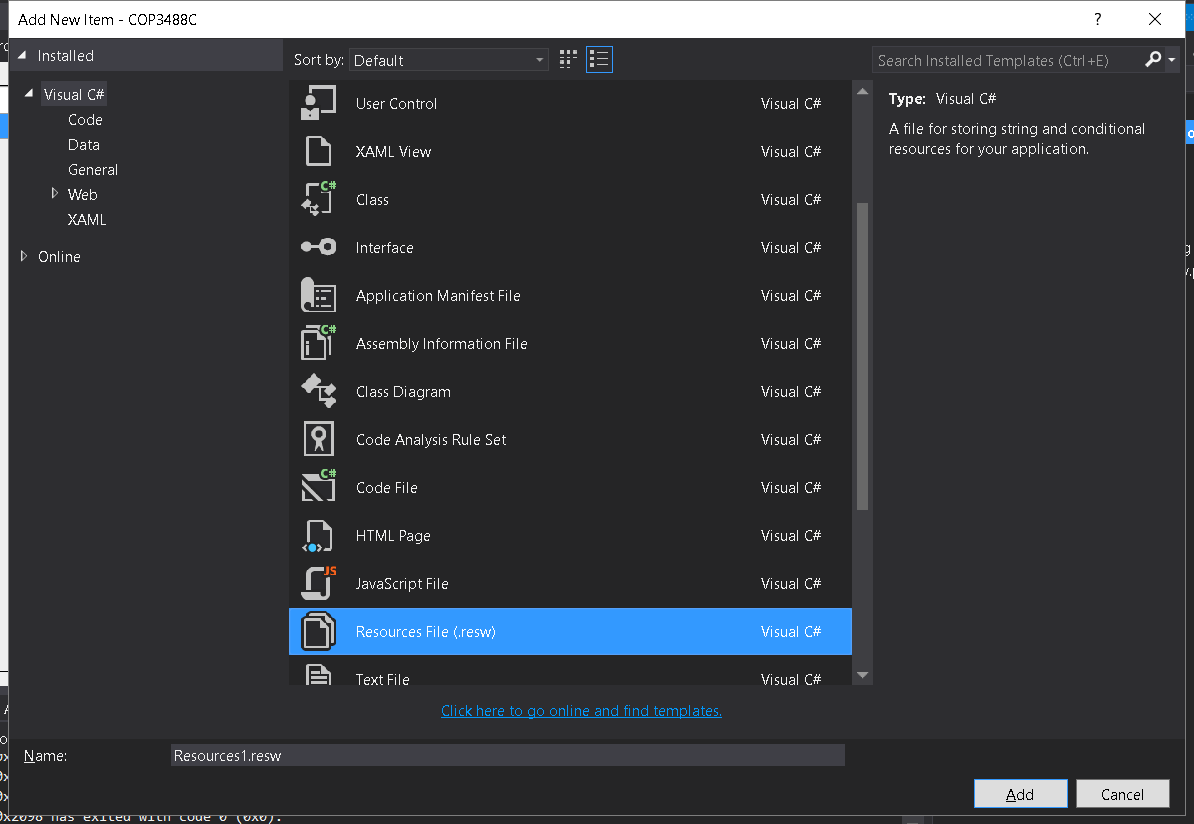
For Module 03 of your Course Project, you will build upon your Visual Studio solution and continue to develop your application. As usual, you should feel free to alter or improve your previous code as you refine your ideas for your final application. For this week, you will continue your development by accomplishing the following steps:

1. Begin this week's assignment by adding a new resource (resw) file in your project. Add the new resource and bind the resource to a UI control. Make sure the string value of the resource is something that you can reuse throughout the application. For instance, the literal value "Designed by: 'Your Name' Software" would be a string value that could potentially be used in the footer of every page in the application. Specifically, you are required to:
   1. Add a new resource.resw file to the project
   2. Use the resource to store a string value
   3. Add a new hyperlink UI control to use as a label in the MainPage.xaml page
   4. Bind the resource to the new UI control in the application
2. Next, write three new C# arrays (or lists) of DataObjects and populate the DataObjects with several **course numbers, course names,**and**course descriptions**from your degree program. Then add several button controls to the MainPage.xaml and show each course number in the button controls. (**Note:** In your final app, you will need to provide a data list of all 14 courses in the upper division of your program, but in this week you should only add data for two or three courses as you will want to move the data to a new page in upcoming weeks.)  
     
   To summarize, you are required to:
   1. Create three new DataObjects (arrays or lists) in the MainPage.xaml.cs file
   2. Assign each of your course numbers, course names and course descriptions to the DataObjects
   3. Show the course number for each button
3. Next, write C# code that dynamically displays the course information back to the screen when you click on the course button. There are a number of ways to accomplish this (e.g. popup message boxes, new pages, or tool tips). Write the code to get the appropriate information from the DataObject you created in Step #2.
4. This week is a good time to start thinking more carefully about the final design and layout of your application. Recall that you will eventually have to add more detailed information on each course, general information on the Program (Program Vision and Program Objective), and the biography pages of four imaginary faculty. In this week, you added specific course information on the MainPage.xaml, but it is highly recommended that you eventually use the MainPage.xaml as a main home page that links to new pages with course lists or detailed content pages.  
     
   Start thinking about what your home page will eventually look like and how you will want to organize all the program data. A good home page should be simple, visually attractive, and provide only the most basic options for continuing to the main content; keep these guidelines in mind as you finalize your ideas. (Also, keep in mind that you could also use a C# timer to display a GIF image as a splash screen that appears briefly before the MainPage.xaml page loads.)

Submit your completed assignment to the drop box below. Please check the **Course Calendar** for specific due dates.

## **Module 03 Project Step-by-Step Guide**

1. Right click on the project solution > Add > New Item
2. A template window will open and select “Resource File (.resw)”



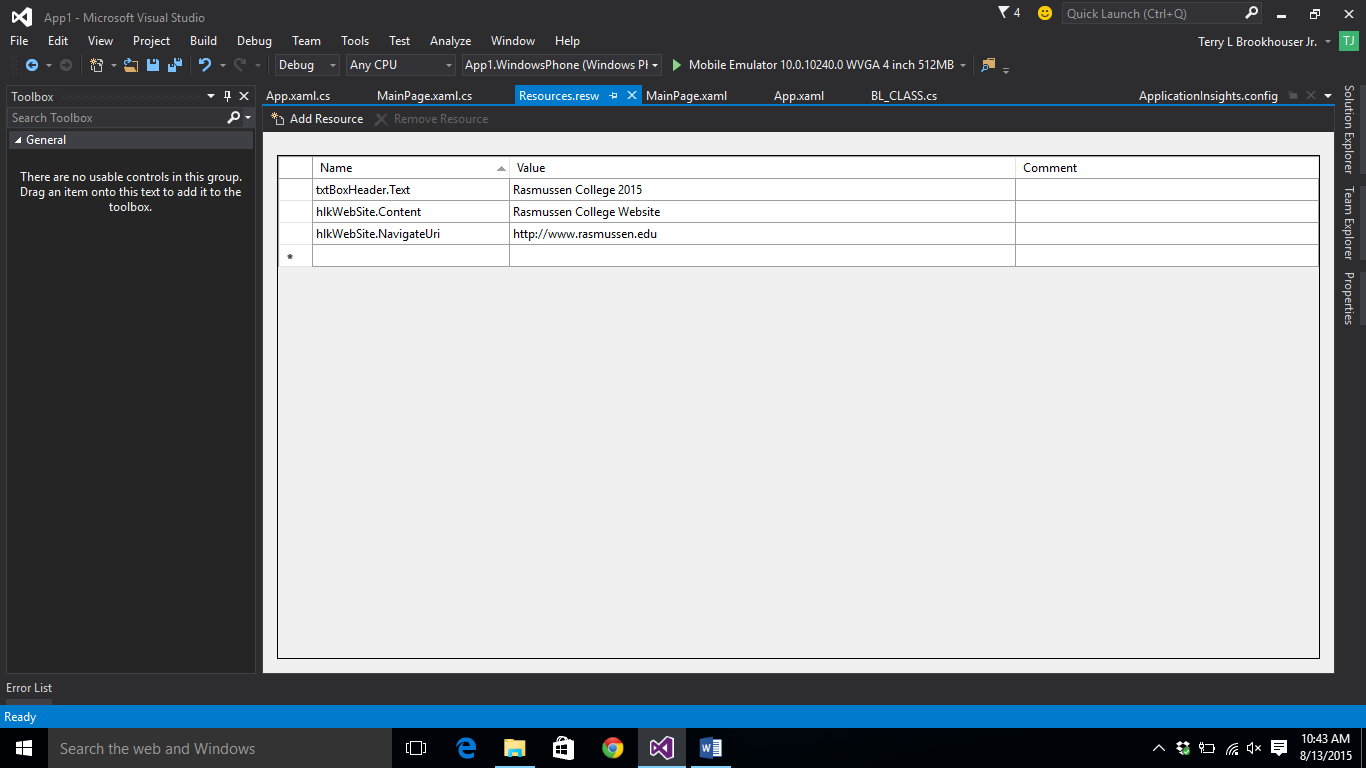
1. This is how to add a resource item to the Resource.resw file. To get the resource item to load into the UI control, you must add a unique x:Uid property to the UI control.

Name: hlkWebSite.NavigateUri

Value: http://www.rasmussen.edu

**Note:** The name hlkWebSite.NavigateUri must be typed exactly the same as above.

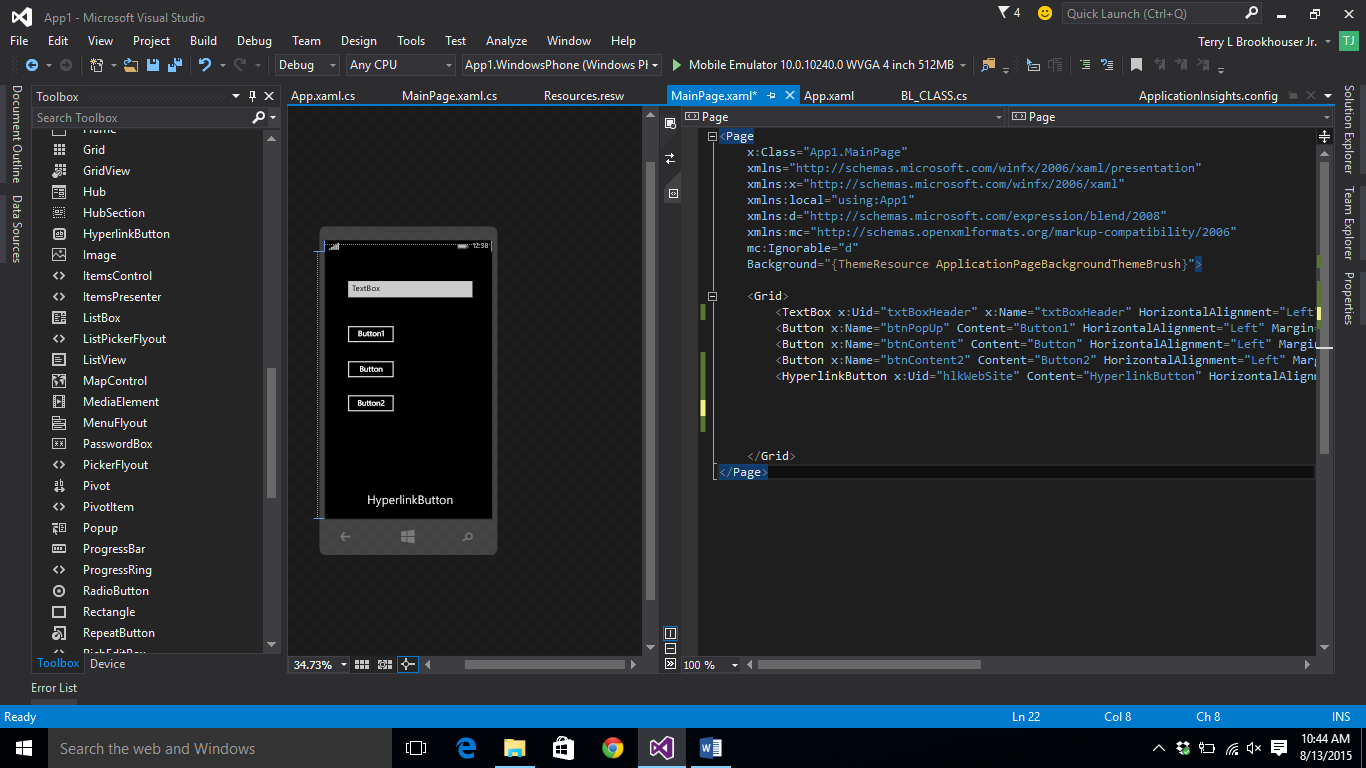
Image #2: The “hlkwebSite” control x:Uid is added to the Resource.resw file as an item



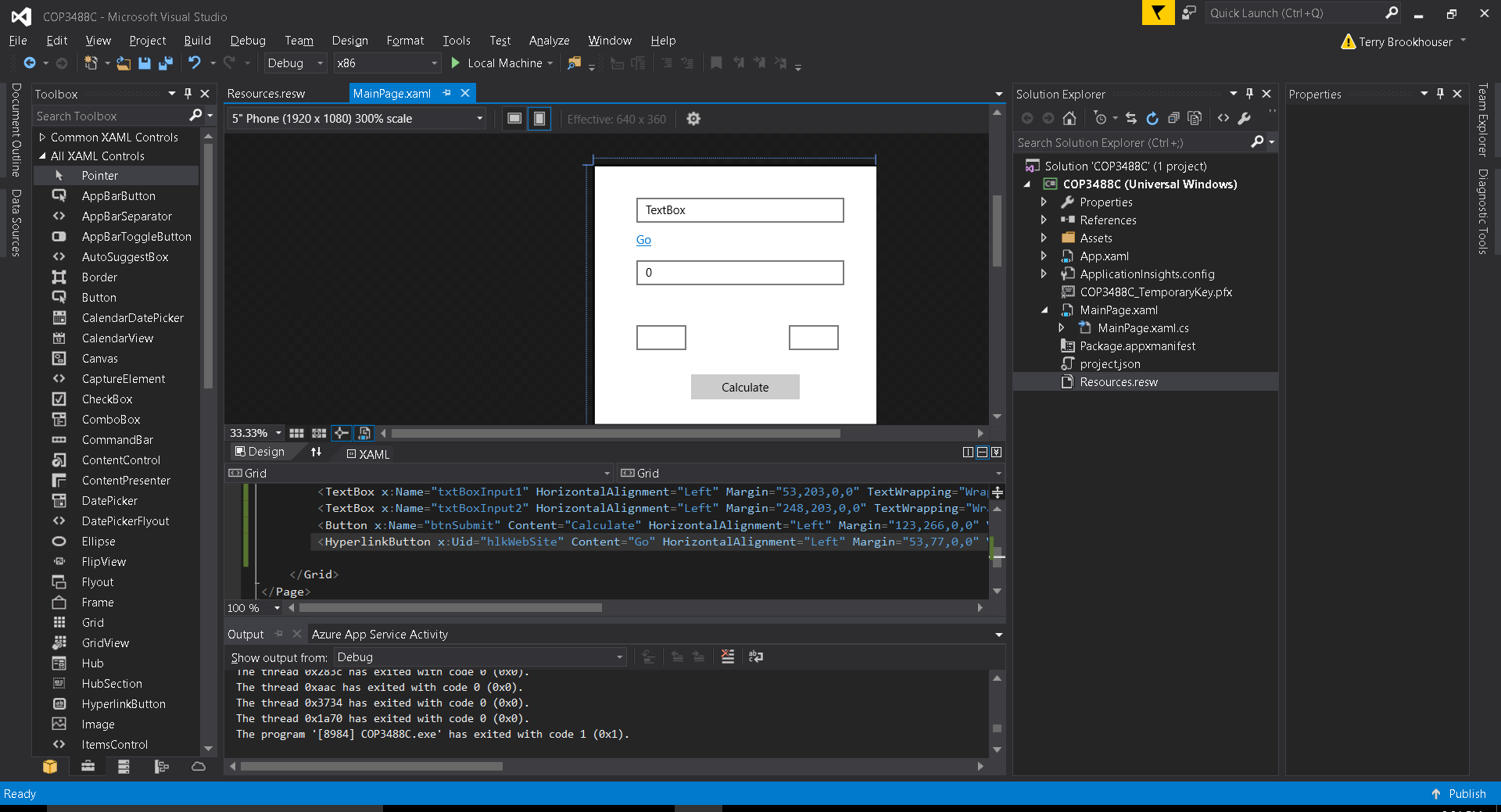
1. Next reference the Resource name from the MainPage.xaml page.

<HyperlinkButton x:Uid="hlkWebSite" Content="Go" HorizontalAlignment="Left" Margin="53,77,0,0" VerticalAlignment="Top"/>

Image #1 This is an example of an x:Uid property added to a hyperlink UI control called “hlkWebSite”



You application might look something like this:



1. Build Solution
2. Next, Add three new buttons to the form so you can display course numbers, names, and descriptions.

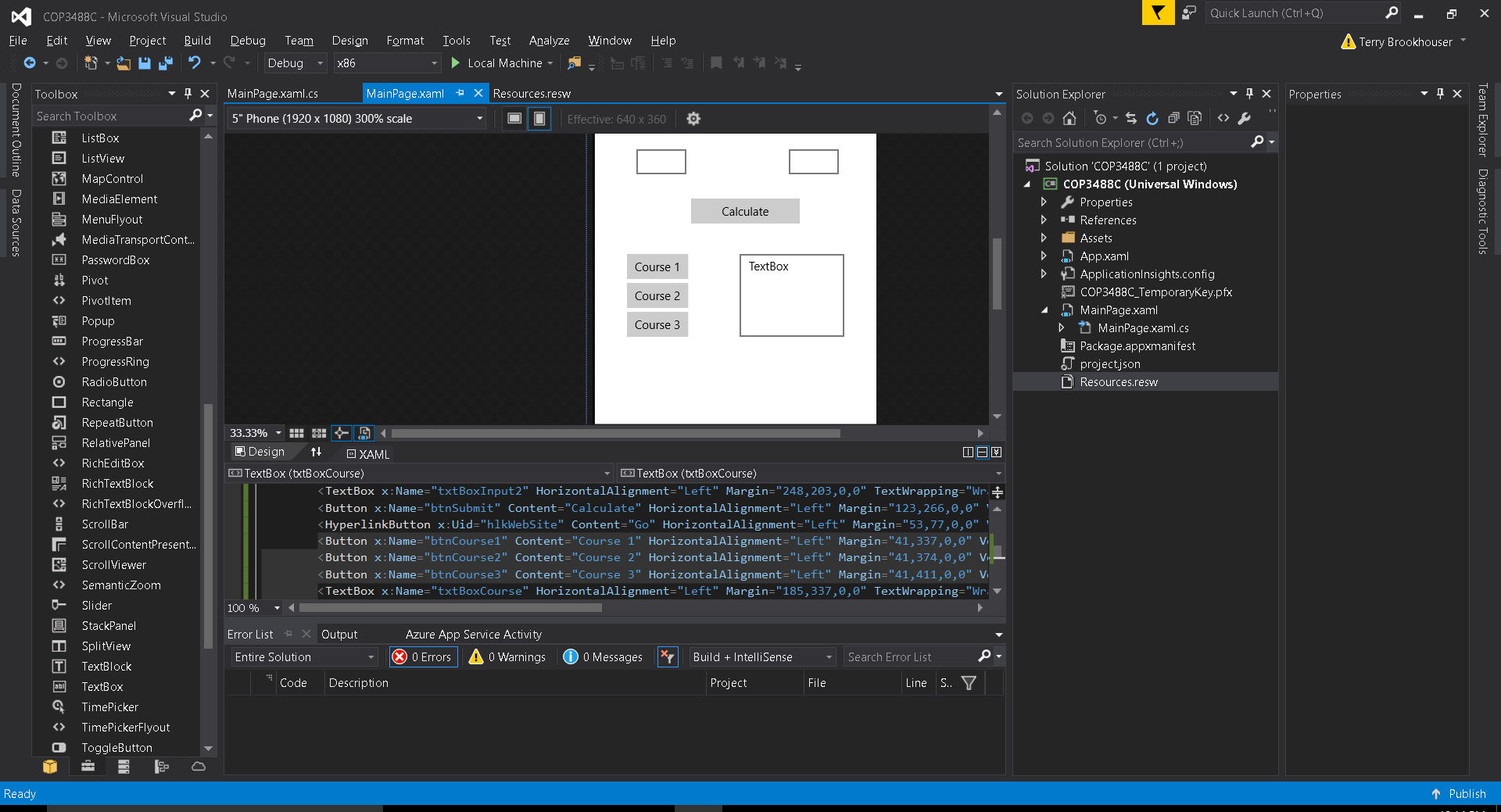
<Button x:Name="btnCourse1" Content="Course 1" HorizontalAlignment="Left" Margin="41,337,0,0" VerticalAlignment="Top" Click="btnCourse1\_Click"/>

<Button x:Name="btnCourse2" Content="Course 2" HorizontalAlignment="Left" Margin="41,374,0,0" VerticalAlignment="Top"/>

<Button x:Name="btnCourse3" Content="Course 3" HorizontalAlignment="Left" Margin="41,411,0,0" VerticalAlignment="Top"/>

1. Make sure each new button name property is renamed appropriately.
2. Add a new textbox UI control to display the values back to the screen

<TextBox x:Name="txtBoxCourse" HorizontalAlignment="Left" Margin="185,337,0,0" TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Height="106" Width="134"/>



1. Double click each button so VS will automatically create the event handler method.

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

{

}

1. Add a Using namespace reference called Windows.UI.Popups to the top of the MainPage.xaml.cs page.

using Windows.UI.Popups;

1. Add a C# property called VarOutput to the MainPage.xaml.cs class file but do NOT put this property inside another method. It must be within the page class but outside of the methods, a good place to put properties is at the top of the page or within a class file.

public static string VarOutput {get; set;} //place property within the class

1. Add a C# array to the new C# button EventHandler method and populate the values into the array

string[] names = new string[3] { "COP3488C", "UWP1", "This course is mobile app development." };

1. Loop through the array and then add the values to a string variable and display the values to either a popup window or the textbox UI control (txtBoxCourse). Add this C# code to the same button EventHandler method with the

for (int i = 0; i < names.Length; i++)

{

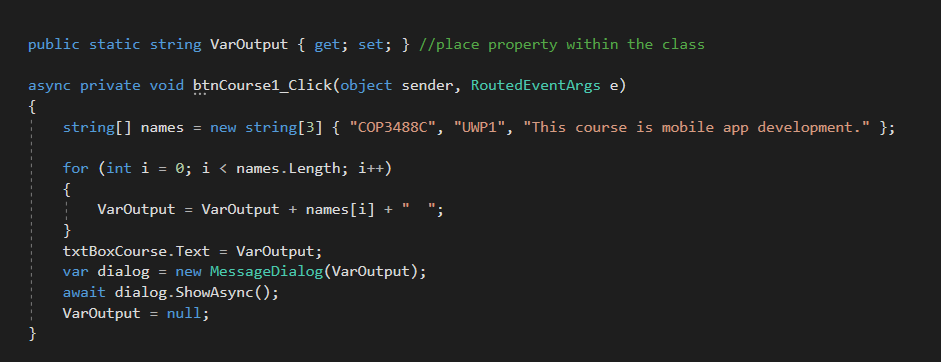
VarOutput = VarOutput + names[i] + " ";

}

txtBoxCourse.Text = VarOutput;

var dialog = new MessageDialog(VarOutput);

dialog.ShowAsync();



1. Build solution and test
2. Done.

# **Module 04 Project Instructions**

For Module 04 of your Course Project, you will build upon your Visual Studio solution and continue to develop your application. As usual, you should feel free to alter or improve your previous code as you refine your ideas for the final application. For this week, you will continue your development by accomplishing the following steps:

1. Begin this lab by creating a new class in your project solution using Visual Studio. Load a new class from the Templates dialog window and name the new class "BL\_PageContent". Make sure the class file is in the same namespace as the project. You will also want to let the class file load in the root folder path of the program. Make sure that the MainPage.xaml inherits the methods from your class "BL\_PageContent"
2. Next, create a method called "CourseCredits()" that contains course credit hours and course perquisites for the two to three courses from your degree program that you worked on last week. Add encapsulation to the method(s) within your class "BL\_PageContent". There are different types of encapsulation, so be sure to implement the type of member access modifier to the methods that allows MainPage.xaml to inherit the methods. You can change the encapsulation in the declaration of the method. Specifically, you will:
   1. Write a C# method called "CourseCredits()"
   2. Set the encapsulation for the method(s) so they can be inherited from MainPage.xaml.cs.

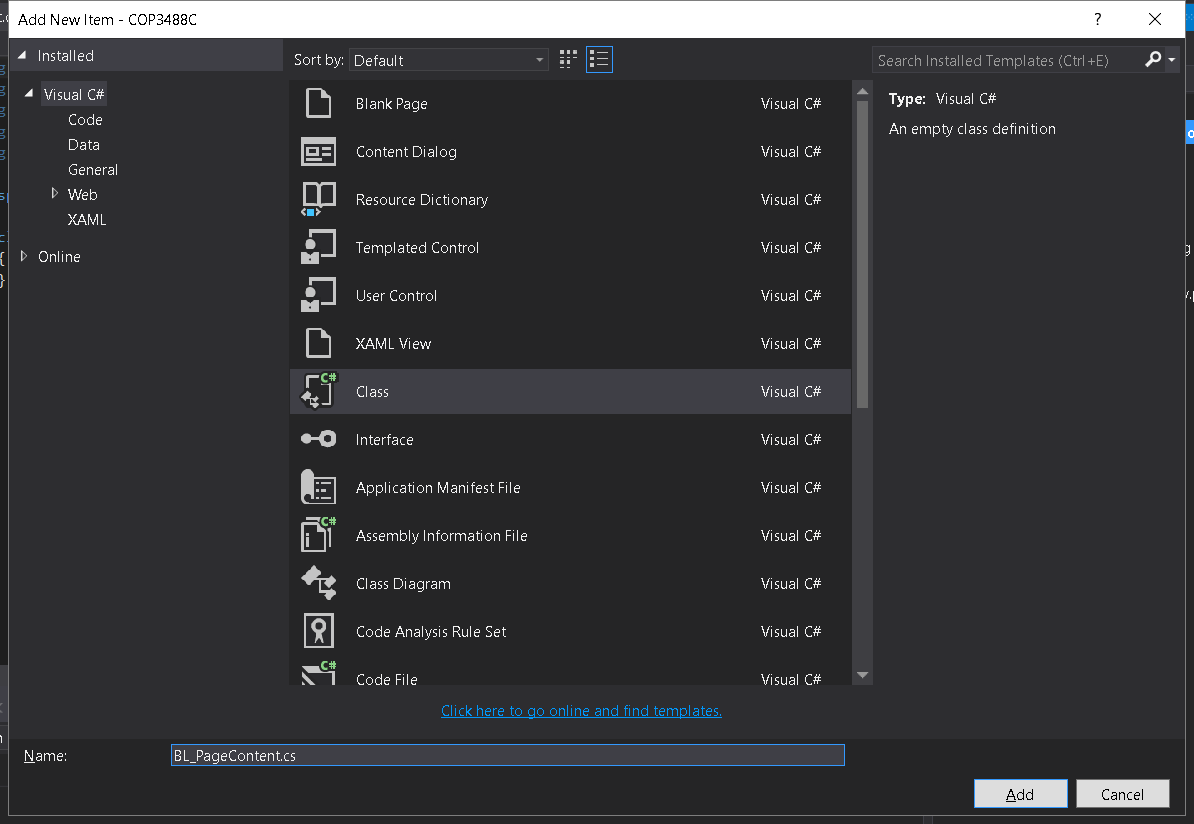
**IMPORTANT:** At this time you should be taking steps towards the final design of your application. Will you continue to list each course on the MainPage.xaml page or will you create a separate page for the overall list of courses and provide a link to that on the MainPage.xaml? As mentioned earlier, it is highly recommended that you use the MainPage.xaml as a simple home page that provides links to more detailed lists and content. Now is the time to start implementing this design. You will now be adding a lot more content to your application and it is best to start creating proper page locations for that content now and avoid creating new pages (and moving around content) in the last week of the project.

1. Finally, for this week you will create one more DataObject to support CourseCredits. Add the content for each course to the DataObject for CourseCredits.  
     
   At this time, you should also have created new pages for each of the two to three courses that you have been working with. You will need to load the content from each DataObject onto each page. Design your page so that when the user clicks on the course button, that page should load with the all the detailed course information (course name, number, descriptions, credit hours, and prerequisites) and all the appropriate content is accessible from the new page that loads. You might want to abandon any popup or tooltips you created earlier and simply show all the course information on the same page.
   1. Add new pages to the project solution for each course button
   2. Add two new textbox controls and use these controls as a label for new pages which will contain the Course Numbers and Course Descriptions
2. This week is also a good time to be thinking about how the final layout for the other pages in your application will look. Will you have one page with all four of the faculty bios? Will you create a faculty list of names and then provide separate pages for each of them? Where will the program information go? Now that you have command of encapsulation methods and DataObjects, you may want to start designing these pages in Visual Studio and begin adding your content right away to start seeing how the new pages look and function.

Submit your completed assignment to the drop box below. Please check the **Course Calendar** for specific due dates.

## **Module 04 Project Step-by-Step Guide**

1. Right click on the project solution > Add > New Item > Select the class item in the template window.
   1. Change the name of the class file to “BL\_PageContent”



1. Create a new method “Course1” in the new class file with a public access modifier
2. Create a new C# property to contain the array string values called “VarOutput”
3. Move the C# code from MainPage.xmal.cs to the new method

class BL\_PageContent

{

public static string VarOutput { get; set; }

public static void CourseCredits ()

{

//string VarOutput = null;

string[] names = new string[3] { "COP3488C", "UWP1", "This course is mobile app development." };

for (int i = 0; i < names.Length; i++)

{

VarOutput = VarOutput + names[i] + " ";

}

}

}

1. Go to MainPage.xmal.cs and remove extra C# code and variables
2. Change the C# so it does not reference the local variable but now it should call the new Class method “BL\_PageContent.CourseCredits()” and once the method is called the property will be loaded so the C# application can reference the new class property “BL\_PageContent.VarOutput”.

async private void btnCourse1\_Click(object sender, RoutedEventArgs e)

{

BL\_PageContent.CourseCredits();

txtBoxCourse.Text = BL\_PageContent.VarOutput;

var dialog = new MessageDialog(BL\_PageContent.VarOutput);

await dialog.ShowAsync();

}

1. Build solution and test

Done.

# **Module 05 Project Instructions**

For Module 05 of your Course Project, you will build upon your Visual Studio solution and continue to develop your application. As usual, you should feel free to alter or improve your previous code as you refine your ideas for your final application.

This week you will need to start adding your data in earnest and verify that the navigation is functioning properly between all the new pages.

For this week, you will continue your development by accomplishing the following steps:

1. XAML language is remarkably flexible and powerful. One demonstration of this is the way XAML can function much like cascaded style sheets (css). In this task, demonstrate that you have an understanding of how to use XAML and change the background color of at least one of the UI controls. In the step-by-step guide we created a button and used XAML to change the background color of the button.
2. Since you are now comfortable adding new pages to your project solution, start adding other new pages to your application including the Faculty list and the Program information. Your Faculty bios should make use of images, but you should also feel free to play around and try integrating a video or an audio file into your application. You should also complete adding the data for the full 14 courses in your program.
3. For this step, you will add a C# property to the class file "BL\_PageContent" and make the property encapsulation accessible from the "MainPage.xaml.cs" or from any other relevant XAML page you have created. Please note that when you make this class property accessible from any particular XAML page, then it can also be seen from the other pages). One example of a C# property you could add is text that appears on every page and notes your authorship: (public static string CreatedBy { get; set; } ). Use the property as a way to pass information between two different classes. Please note that while it is very common to use properties and build objects to share data between pages, for this course you will simply use the property to retrieve the assigned data from the BL\_PageContent class.

Submit your completed assignment to the drop box below. Please check the **Course Calendar** for specific due dates.

Save your Visual Studio Solution in a zip file before submitting. The name of the file should be your first initial and last name, followed by an underscore and the name of the assignment, and an underscore and the date. An example is shown below:

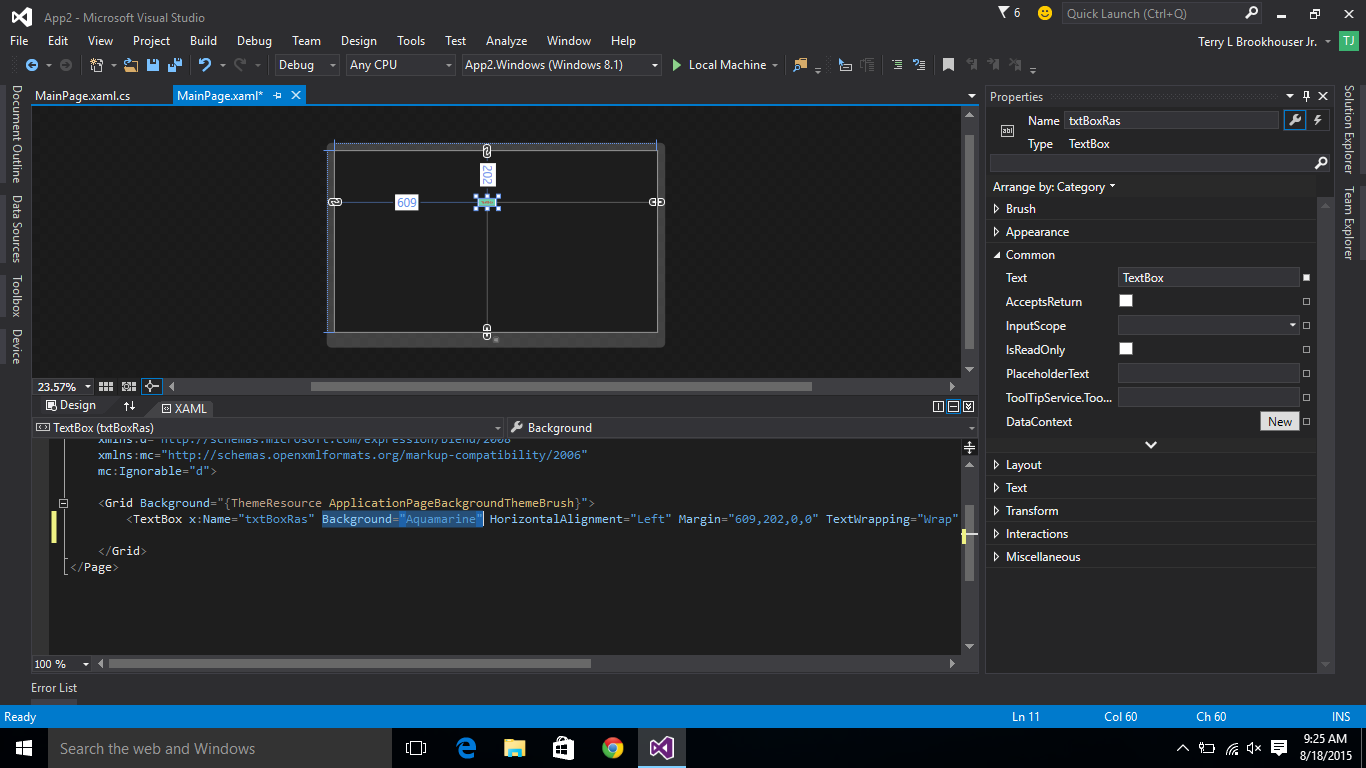
## **Module 05 Project Step-by-Step Guide**

1. Change the background property of the UI control

Background=”Aquamarine”

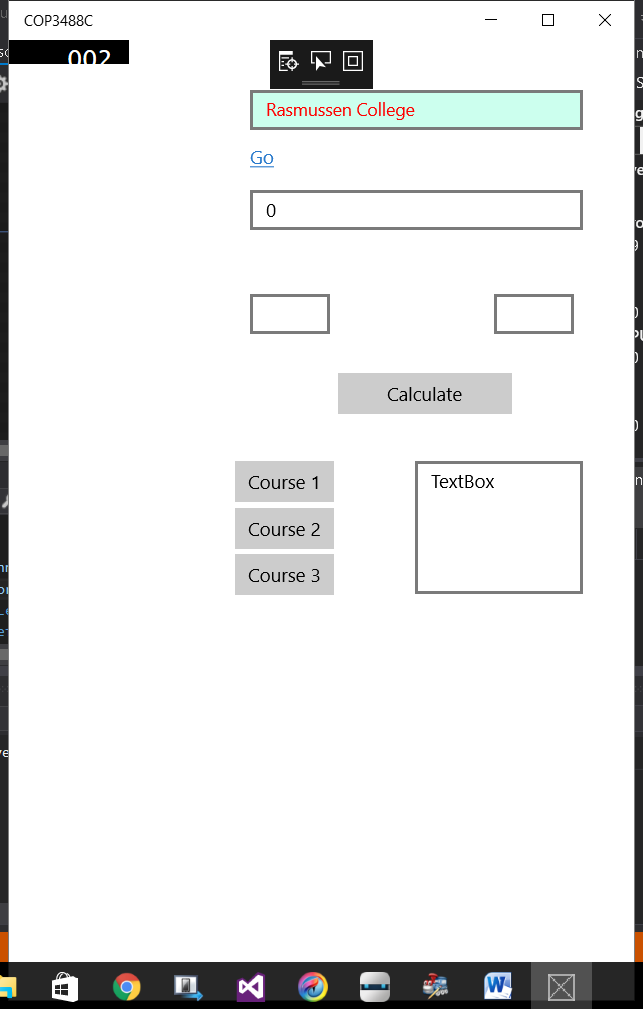
<TextBox x:Name="txtBoxRas" Background="Aquamarine" HorizontalAlignment="Left" Margin="53,40,0,0" TextWrapping="Wrap" Text="TextBox" VerticalAlignment="Top" Width="266"/>

Image #1 XAML background property

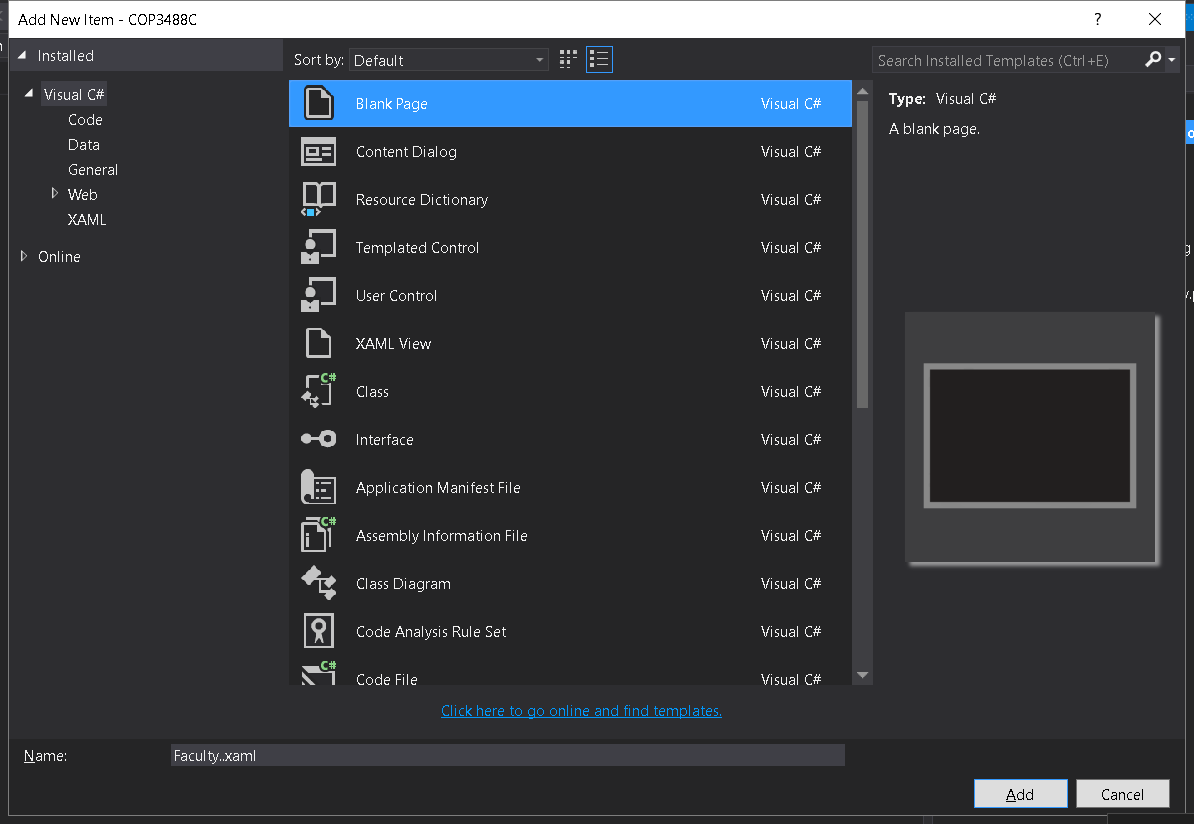


1. Build solution and test

Image #2 Screen shows an Aquamarine background in the UI control



1. Add a new page to your project solution
   1. Right click on project solution > Add > New Item
   2. In the template window select “Blank Page”
   3. Change the name of the page to “Faculty.xaml”



1. Add a new image to the top of this page
2. Add a new xaml property called source and add this URL
   1. http://www.rasmussen.edu/images/logo-internal.png

XAML:

<Image x:Name="image" HorizontalAlignment="Left" Height="100" Margin="10,16,0,0" VerticalAlignment="Top" Width="340" Source="http://www.rasmussen.edu/images/logo-internal.png"/>

1. Add a link to this new Faculty.xmal page from MainPage.xaml
   1. Add the new HyperLink to MainPage.xaml
   2. Double click the HyperLink so VS will automatically create the C# event handler
   3. Add this C# code to the HyperLink\_Click method
   4. this.Frame.Navigate(typeof(Faculty));

XAML:

<HyperlinkButton x:Name="hypLnkPage2" Content="Page" HorizontalAlignment="Left" Margin="123,77,0,0" VerticalAlignment="Top" Click="hypLnkPage2\_Click"/>

C#:

private void hypLnkPage2\_Click(object sender, RoutedEventArgs e)

{

this.Frame.Navigate(typeof(Faculty));

}

1. Build solution and test
2. Next, let’s add a property to the class file “BL\_PageContent.cs”.

C#:

public static string CreatedBy { get; set; }

1. Reference the new CreatedBy class property from both Faculty.xaml and the MainPage.xaml pages
   1. Notice the xaml below enabled “IsReadOnly” so the value cannot be changed and appears locked.
   2. The board around the textbox is white so this UI control now appears to be a label and not an interactive textbox.
   3. Add this xaml tag to both pages

XAML:

<TextBox x:Name="txtBoxFooter" HorizontalAlignment="Left" Margin="41,488,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="376" BorderBrush="#FFFDFDFD" IsReadOnly="True"/>

* 1. Go to the Faculty.xaml.cs and MainPage.xaml.cs and add C# to populate the textbox value field when the page loads.

C#:

public MainPage()

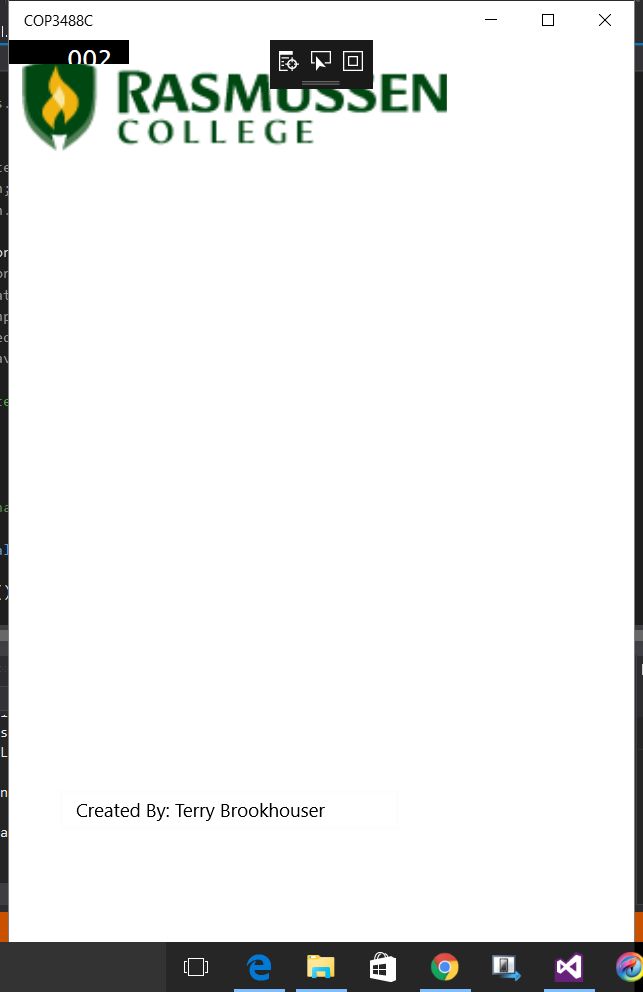
{

this.InitializeComponent();

BL\_PageContent.CreatedBy = "Created By: Your Name";

txtBoxFooter.Text = BL\_PageContent.CreatedBy;

}



1. Build solution and test

Done.