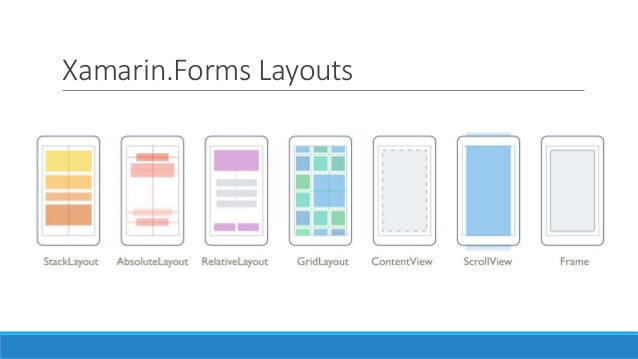
**The laboratory work 7**

**Creating calculator in Xamarin**

Here we will build a simple calculator app using Grid Layout in Xamarin.Forms. So, let us dive into the article. The layout is the style of representing the elements in the applications. The different types of layouts available are,

* Stack Layout
* Grid Layout
* Relative Layout
* Absolute Layout and more.



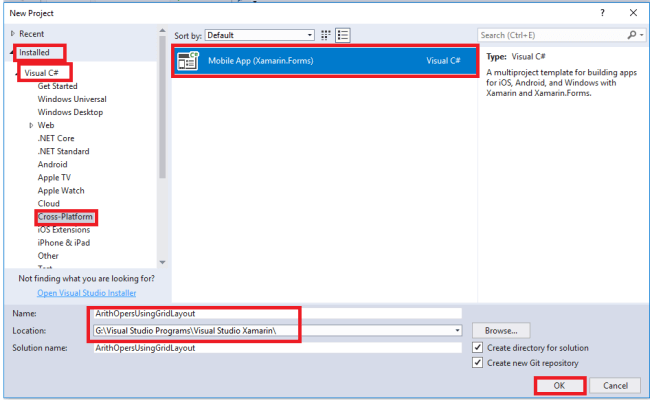
In Grid Layout, elements are arranged in the form of rows (horizontal) and columns (vertical). One row can have any number of columns in it. Grid Layout is somewhat similar to Relative Layout. Now, let us see how to create a simple calculator using Grid Layout. The count of rows and columns starts from 0.

**Prerequisites**

* Windows10
* Visual Studio 2017 Community/ Professional/ Enterprise Edition (Anyone should be installed)
* Xamarin Package

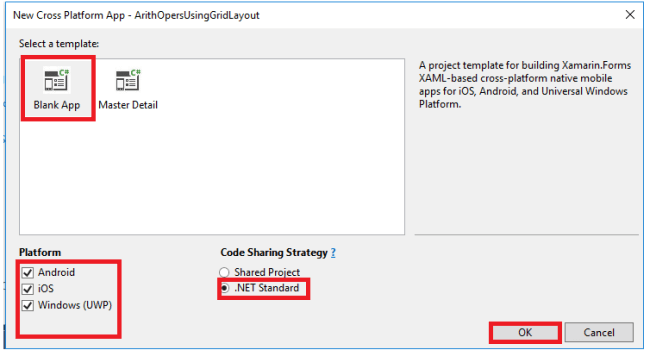
**Step 1**

* Open your Visual Studio 2017 Community Edition.
* Go to File -> New -> Project. Then, a new window will appear.
* Then, under Installed, select Visual C# -> Cross-Platform.
* On the right side of the window, select Mobile App (Xamarin.Forms). Then, give the name of the project and save it in your required location.
* Click OK.



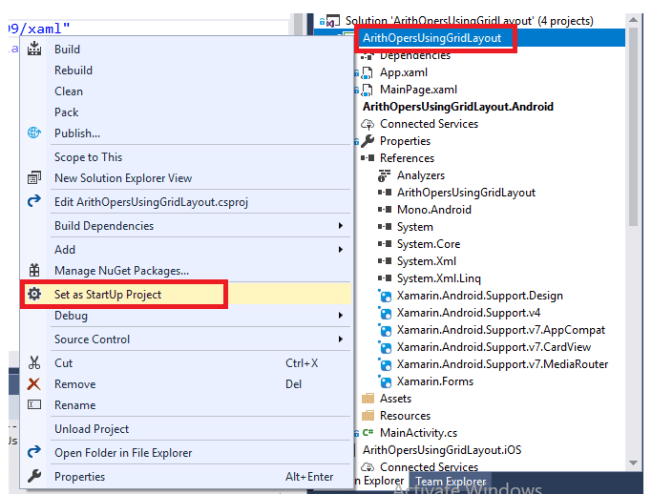
**Step 2**

* Now, a new window will appear on your screen. Under template, select Blank App.
* Select all platforms or your required platforms for executing your application.
* Then, select .NET Standard and click OK.



**Step 3**

Now, open Solution Explorer and select your project then, set your portable file as a startup project.



**Step 4**

Now, write the following XAML Code in your MainPage.xaml file.

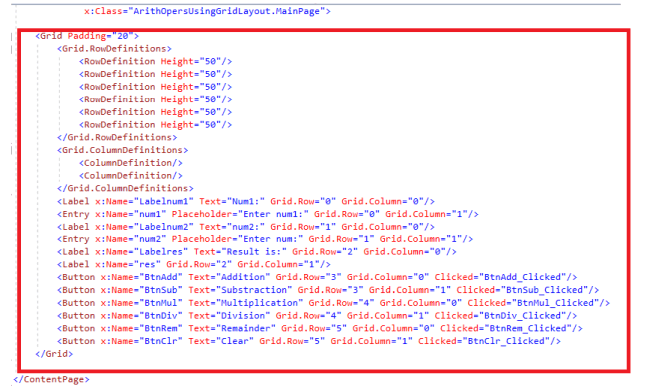
**XAML Code**

|  |
| --- |
|  <Grid Padding="20">       <Grid.RowDefinitions>           <RowDefinition Height="50" />           <RowDefinition Height="50" />           <RowDefinition Height="50" />           <RowDefinition Height="50" />           <RowDefinition Height="50" />           <RowDefinition Height="50" />       </Grid.RowDefinitions>       <Grid.ColumnDefinitions>           <ColumnDefinition/>           <ColumnDefinition/>       </Grid.ColumnDefinitions>       <Label x:Name="Labelnum1" Text="Num1:" Grid.Row="0" Grid.Column="0" />       <Entry x:Name="num1" Placeholder="Enter num1:" Grid.Row="0" Grid.Column="1" />       <Label x:Name="Labelnum2" Text="num2:" Grid.Row="1" Grid.Column="0" />       <Entry x:Name="num2" Placeholder="Enter num:" Grid.Row="1" Grid.Column="1" />       <Label x:Name="Labelres" Text="Result is:" Grid.Row="2" Grid.Column="0" />       <Label x:Name="res" Grid.Row="2" Grid.Column="1" />       <Button x:Name="BtnAdd" Text="Addition" Grid.Row="3" Grid.Column="0" Clicked="BtnAdd\_Clicked" />       <Button x:Name="BtnSub" Text="Substraction" Grid.Row="3" Grid.Column="1" Clicked="BtnSub\_Clicked" />       <Button x:Name="BtnMul" Text="Multiplication" Grid.Row="4" Grid.Column="0" Clicked="BtnMul\_Clicked" />       <Button x:Name="BtnDiv" Text="Division" Grid.Row="4" Grid.Column="1" Clicked="BtnDiv\_Clicked" />       <Button x:Name="BtnRem" Text="Remainder" Grid.Row="5" Grid.Column="0" Clicked="BtnRem\_Clicked" />       <Button x:Name="BtnClr" Text="Clear" Grid.Row="5" Grid.Column="1" Clicked="BtnClr\_Clicked" />   </Grid> |

In the above XAML code we create a Grid Layout by writing Grid in tags. Then, we apply padding on the layout for 20; .i.e., when we create any element in the grid layout the text of those elements are moved to 20 value to the element of it; i.e., if we create a button and enter any text then, that text will be placed after 20 values from each side of  button. We then create Rows and Columns in the layout.

Grid.RowDefinitions is used to declare that we want to create a row and similarly, Grid.ColumnDefinitions is used to create columns in the layout. Now, we use RowDefinition for creating a row i.e., if we want to create 4 rows then we declare RowDefinition 4 times. Similarly, if we want to create 10 columns then, we declare ColumnDefinition 10 times. We can adjust the height of the rows and width of the columns.

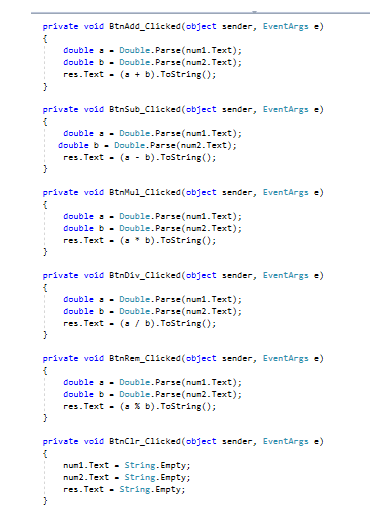
After declaring the required rows and columns we declare our required elements for the application. Now, we want to place those elements in the required rows and columns then we define the required row and column with Grid.Row is used to display the Row and Grid.Column is used to display the Column.



**Step 5**

Now, open your MainPage.xaml.cs for writing the C# code of your application.

|  |
| --- |
| 1. private void BtnAdd\_Clicked(object sender, EventArgs e) { 2. double a = Double.Parse(num1.Text); 3. double b = Double.Parse(num2.Text); 4. res.Text = (a + b).ToString(); 5. } 6. private void BtnSub\_Clicked(object sender, EventArgs e) { 7. double a = Double.Parse(num1.Text); 8. double b = Double.Parse(num2.Text); 9. res.Text = (a - b).ToString(); 10. } 11. private void BtnMul\_Clicked(object sender, EventArgs e) { 12. double a = Double.Parse(num1.Text); 13. double b = Double.Parse(num2.Text); 14. res.Text = (a \* b).ToString(); 15. } 16. private void BtnDiv\_Clicked(object sender, EventArgs e) { 17. double a = Double.Parse(num1.Text); 18. double b = Double.Parse(num2.Text); 19. res.Text = (a / b).ToString(); 20. } 21. private void BtnRem\_Clicked(object sender, EventArgs e) { 22. double a = Double.Parse(num1.Text); 23. double b = Double.Parse(num2.Text); 24. res.Text = (a % b).ToString(); 25. } 26. private void BtnClr\_Clicked(object sender, EventArgs e) { 27. num1.Text = String.Empty; 28. num2.Text = String.Empty; 29. res.Text = String.Empty; 30. } |

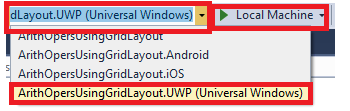


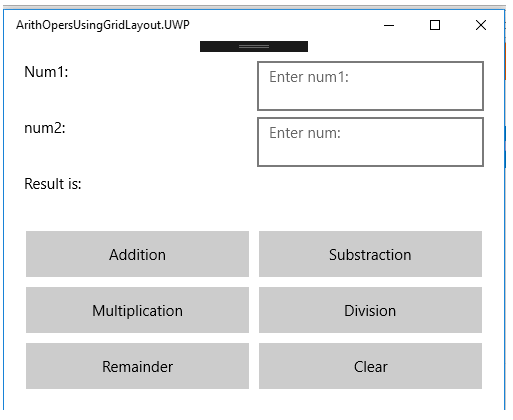
**Step 6**

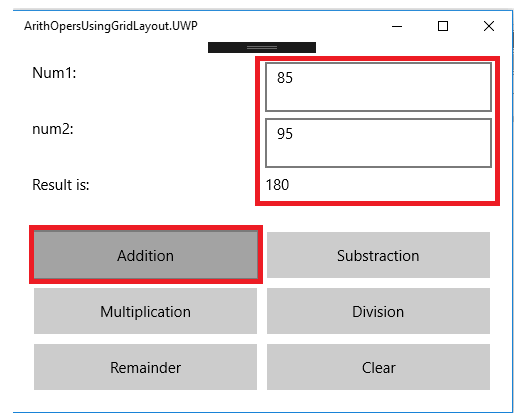
Then, on your portable file click on the build option to build your app.

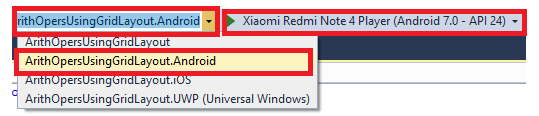
**Step 7**

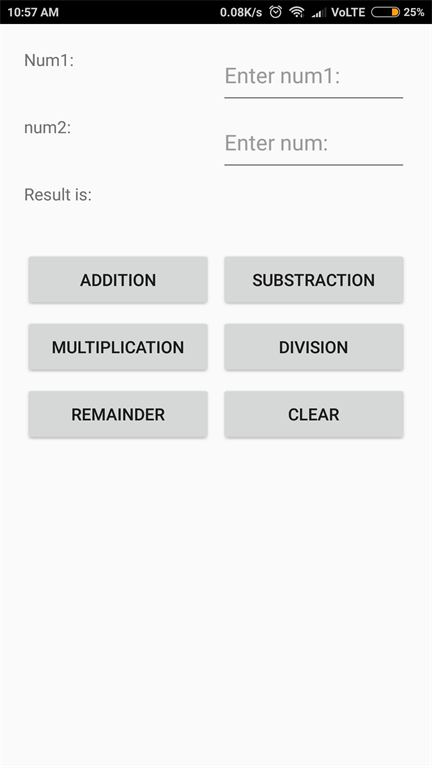
Then, select the type of platform you want to run your application.

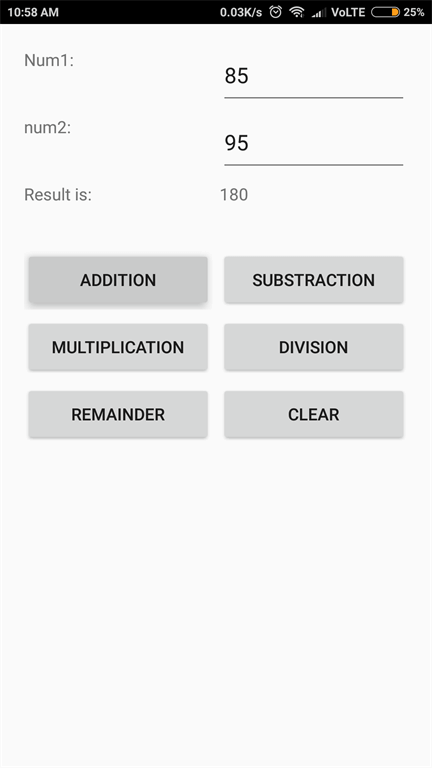












**Conclusion**

Now, we have created a simple calculator application using Xamarin.Forms with Grid Layout.