# Introduction to Xamarin Projects

## Creating a Xamarin project

* In Visual Studio 2022 select – New Project – Search for the keyword “mobile” – Choose “Mobile App (Xamarin.Forms)”
* Call the App: X01Intro (do not place any special characters or spaces in the name)
* Use a blank app template and select only “Android” (deselect other choices)
* If you are prompted to accept a license agreement do so
* Once the project solution finishes loading, choose “Build – Build Solution”

## First Solution and Projects Layout

There are three projects in the solution:

1. Shared Project: Contains a file for laying out the interface (for now) called MainPage.xaml
   1. This is a content page, which we can also define in code
2. Android Project: Activity that contains the app from the shared project
3. iOS Project: AppDelegate class that launches the app from shared project (not present in our solution as we deselected this option)

## Create an Android Emulator

Select “Tools – Android – Android Device Manager”

Choose “New”. Keep the defaults except change the OS from the dropdown to API 28.

When it finishes downloading and setting itself up choose “Start”

It will take the phone a few moments to start. When it is sitting at the home screen, choose “Debug – Start Without Debugging” or push Ctrl-F5. Your app should deploy to the phone, though if you get a black screen just repeat the Ctrl-F5.

## Layouts in Xamarin

Layout and content can be added both programmatically and using XAML. Open MainPage.xaml (which we will delete soon) and note that the markup within it corresponds to the screen on the phone that is presented when the app starts.

## Accessing screen size – An example that uses the Shared and Android projects

Delete MainPage.xaml. Choose the dropdown beside App.xaml, and then open App.xaml.cs. The constructor here is called from the Android project to initialize our app. Make the changes highlighted below to create variables to store and display screen size information that will be calculated in the Android project:

In App:

public static Size ScreenSize;

public static Label label;

public App ()

{ InitializeComponent();

label = new Label { Text = "Size?" };

MainPage = new ContentPage { Content = label}; //we are replacing MainPage.xaml with our page

}

In the Android project, open MainActivity.cs and within the onCreate method before the LoadApplication(new App()) line add:

App.ScreenSize = new Xamarin.Forms.Size( Resources.DisplayMetrics.WidthPixels, Resources.DisplayMetrics.HeightPixels);

After the loadapplication line add:

App.label.Text = App.ScreenSize.ToString();

Run the app to see screen size in the label. We’re accessing the label in the App class from the Android project and showing something Android “knows about” (screen size). There are other ways to get this info, but this shows one way the two projects can interact. Later we will make use of a library called Xamarin.Essentials which has screen size built in as well as other helpful pieces of functionality to interact with Android.

## Example to show grid of squares

Add these constants to the App class. Const int padding = 10; const int spacing = 5; const int count = 5;

In the App constructor, calculate the appropriate box size based on screen size:

(don’t be too concerned about the math, we are simply determining how big to make each box dynamically based on the screen size Android provided)

var boxSize = Math.Abs((ScreenSize.Width - 2 \* padding + spacing) / count – spacing);

Programatically add a ColorGrid as the MainPage (replace the current “MainPage = …” line with this):

MainPage = new ContentPage {

Padding = padding,

Content = new ColorGrid(boxSize, count, count) { //this will be a grid of BoxViews

RowSpacing = spacing,

ColumnSpacing = spacing,

VerticalOptions = LayoutOptions.CenterAndExpand,

},

};

Add a ColorGrid class that extends the Grid class that will generate a Grid of colored boxes for content:

This can be added within the namespace of the App.xaml.cs file, just don’t place it within the App class itself, so it will be placed after the closing brace for the App class but still within the closing brace for the namespace of the file.

public class ColorGrid: Grid

{

public ColorGrid(double boxSize, int rows, int columns)

{

//just a loop to create a bunch of boxes in the grid with dynamic colors based on position

for (var row = 0; row < rows; row++)

for (var column = 0; column < columns; column++) {

var box = new BoxView {

Color = Color.FromRgb(row \* 256 / rows, column \* 256 / columns, 127),

WidthRequest = boxSize,

HeightRequest = boxSize,

};

Children.Add(box, row, column);

}

}

}

Run the app. It should look something like this, though it will be slightly different on the emulator you have running:

