xamarin basics HAND-ON lAB

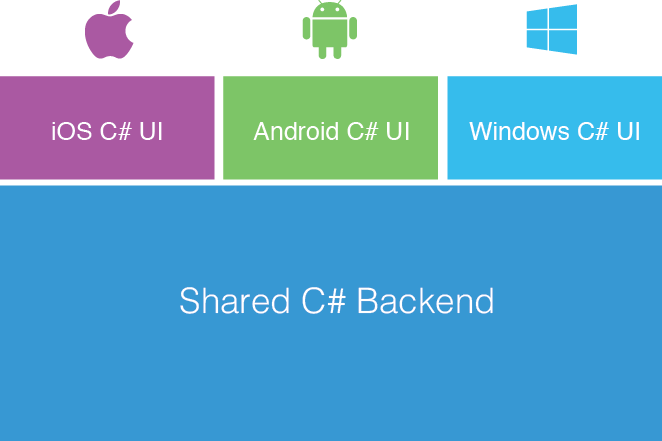
Mohamed Izzat Khair, Technical Evangelist



**overview of xamarin**

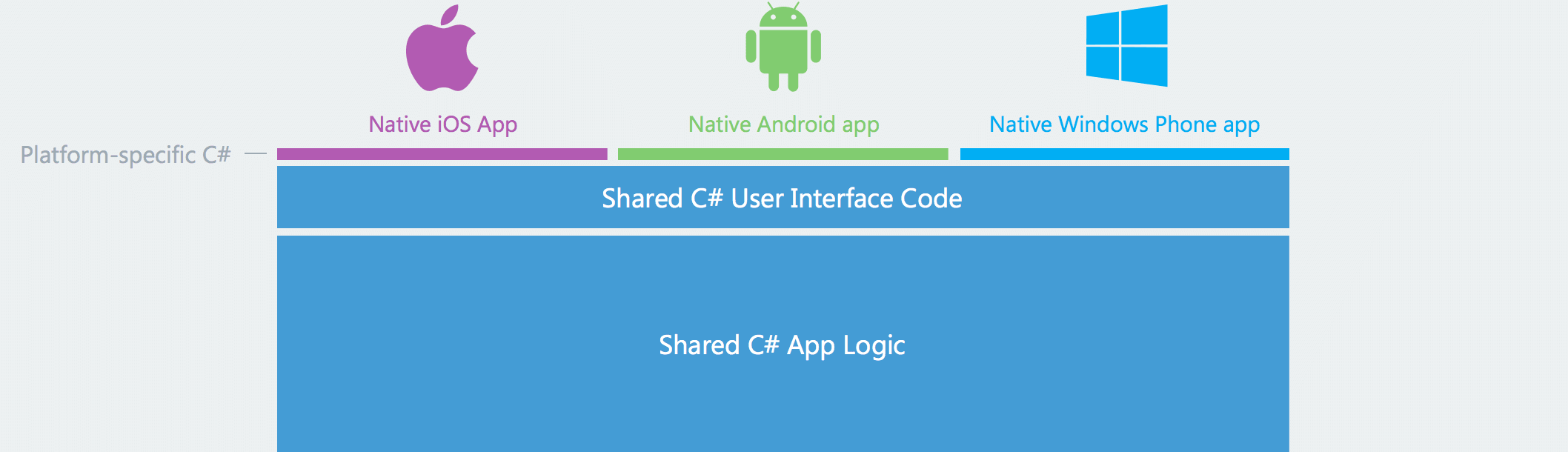
Xamarin allows you to build mobile application on multiple platforms to deliver native experience & performance to users by just using a single programming language, C#. Through Xamarin, developers will still have the access to the native API & library on specific platforms such like Android, iOS and Windows.

traditional xamarin approach



Traditional Xamarin allows you to code natively for platforms such as Android, iOS and Windows while still being able to share a common backend code across all the platforms (Windows, IOS and Android).The main purpose of traditional Xamarin is that you are able to code using the C# .However, you are still required to build up UI for each platform. This is great if you would to have a native user experience.

XAMARIN. Forms



Most of the App Logic and User Interface Codes to be shared across all the platforms with Xamarin Forms. This is a major advantage for using Xamarin Forms Is that the user interface of your application will remain consistent throughout the different platforms (Android, IOS and Windows).However, you can still natively add codes for the specific platform if you would like to do so.

Ways to share code between platforms

Method 1

Shared Projects – Use the Shared Asset Project type to organize your source code, and use #if compiler directives as required to manage platform-specific requirements.

Method 2

Portable Class Libraries – Create a Portable Class Library (PCL) targeting the platforms you wish to support, and use Interfaces to provide platform-specific functionality.

*In this Hands-On-Lab we will be using Portable Class Libraries for sharing of code across the class.*

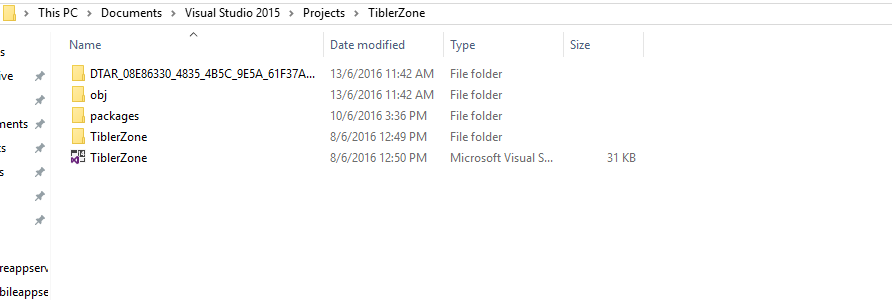
**hANDS ON LAB**

1) Download the TiblerZone folder from the link provided, Extract the folder and Open it

Download Here :[https:/github.com/izzkhair/Xamarin-HandsOnLab](https://github.com/izzkhair/Xamarin-HandsOnLab)



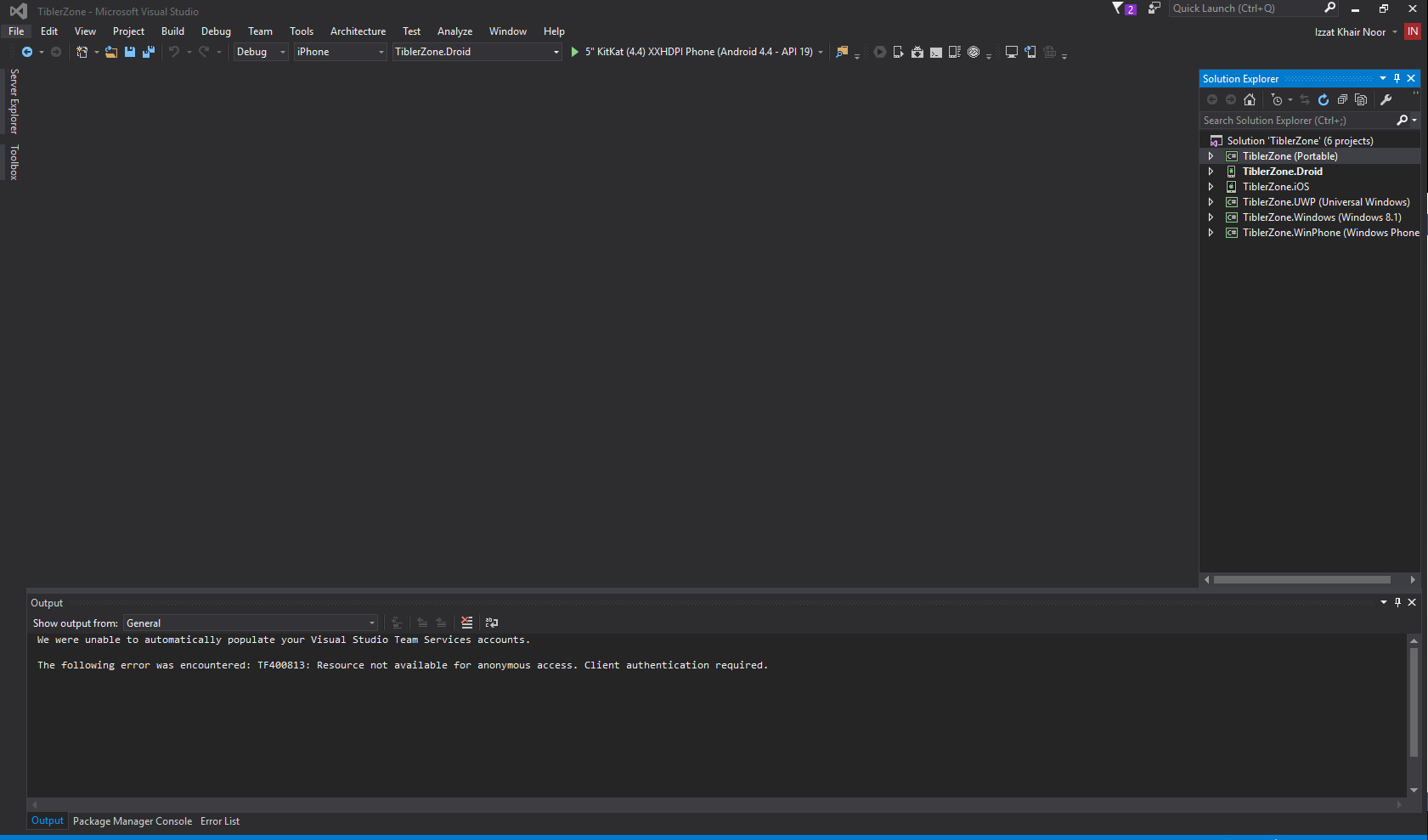
Once Open Will Be Presented with this,



2) Click on the **TiblerZone Solution File** to open the solution in Visual Studio



Once Open Will Be Presented with this,



Solution Explorer

3) **Click on Tools** > **Nuget Package Manager** >**Manage Nuget Package for Solution**

4) On the top of the window there will be the prompt to restore the Nuget Package.

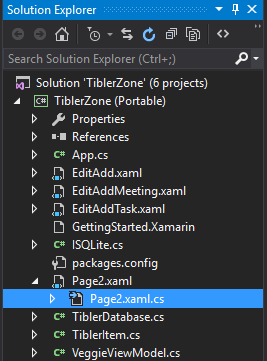
Click on **Restore.**



5) On the right hand side you will see a solution explorer, expand **TiblerZone (Portable)**

Expand **Page2.xaml**

Click on **Page2.xaml.cs**



**nAVIGATing between pages**

6) Scroll down to the end, add the method below inside of the namespace and class

void listItemSelected(object sender, SelectedItemChangedEventArgs e)

{

if (e.SelectedItem == null) return;

todoItem = (TiblerItem)e.SelectedItem;

if (todoItem.category == "Lecture")

{

Navigation.PushAsync(new EditAdd(todoItem.ID));

}

else if (todoItem.category == "Task")

{

Navigation.PushAsync(new EditAddTask(todoItem.ID));

}

else if (todoItem.category == "Meeting")

{

Navigation.PushAsync(new EditAddMeeting(todoItem.ID));

}

else

{

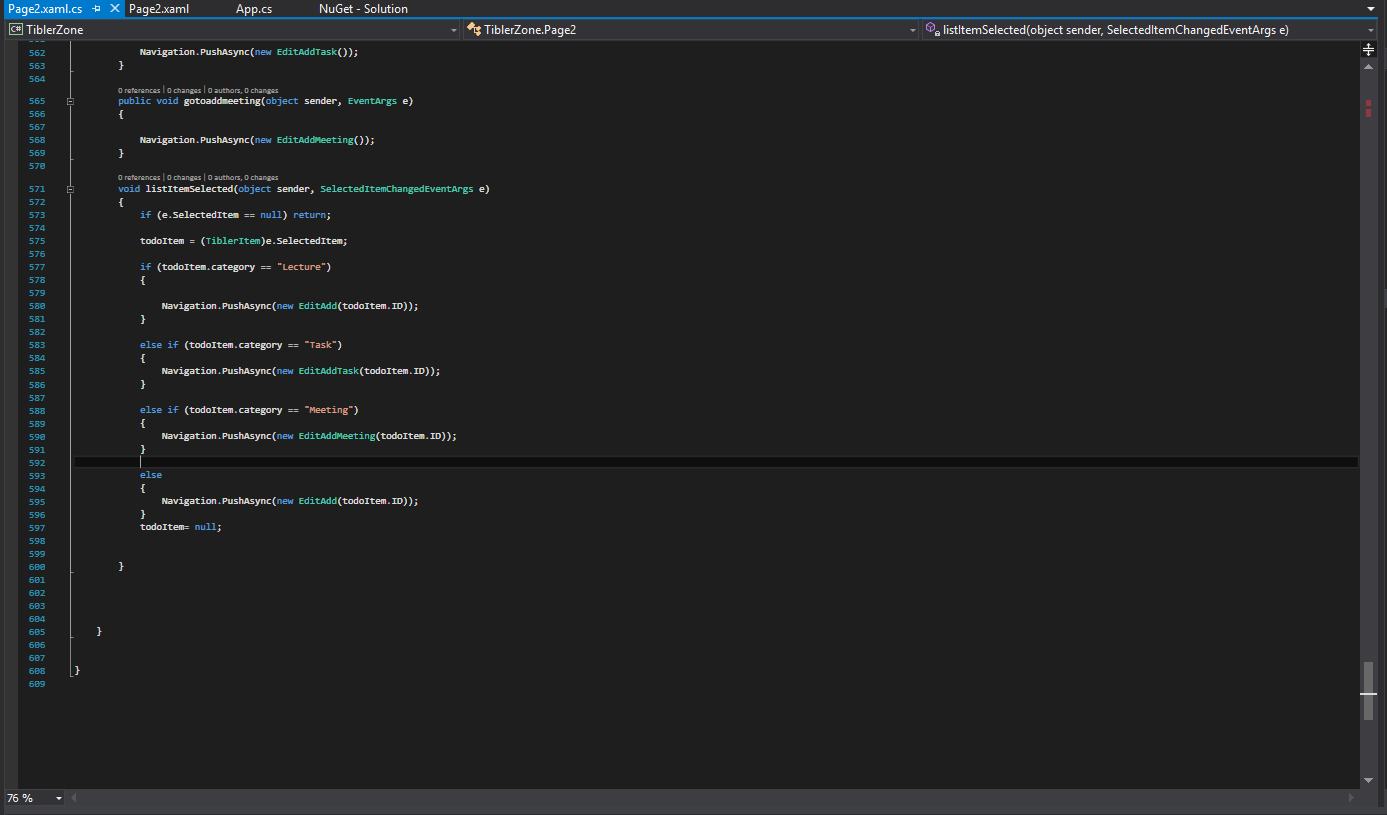
Navigation.PushAsync(new EditAdd(todoItem.ID));

}

todoItem= null;

}

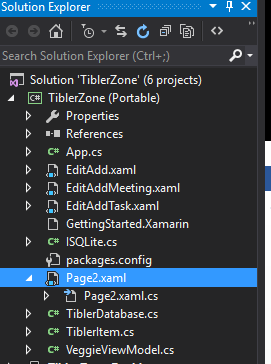
It would look like this, once the code is added.



**Explanation**

The method is responsible for navigating the list view items to the right page. So if the user tapped on an item in the list view it will redirect to the **right page (Lecture, Task and Meeting Page).**

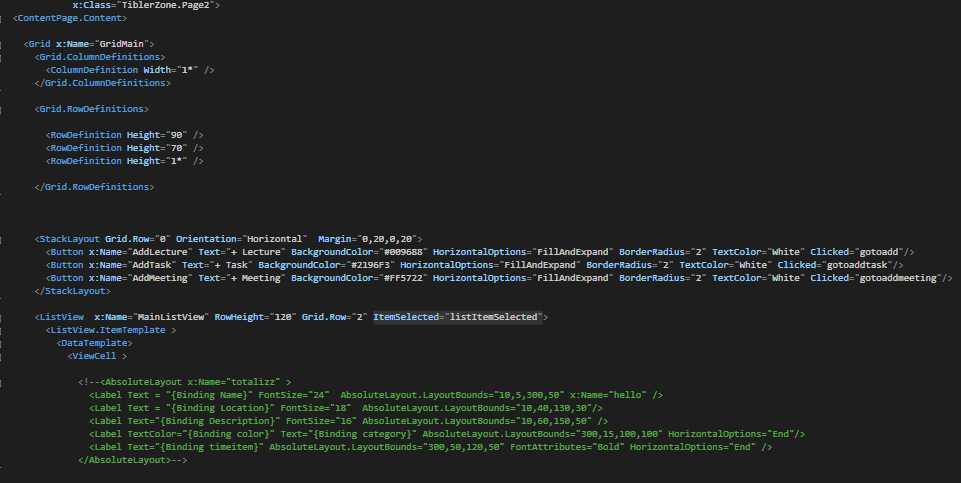
7) Go to Solution Explorer on the right and Click on Page2.Xaml



8) Find the List View Task with the name **MainListView**

**Add the Property Code below to the List View tag**

ItemSelected="listItemSelected"



**Explanation**

This is ensure that every time an item is tapped on the list view, it will call the method name **listItemSelected.** This methods contains various navigation code to navigate the item to the individual page (Completed in Step 6).

9) Go Back to Page2.xaml.cs

Add the following code to populate the picker in the Dictionary nametocolor

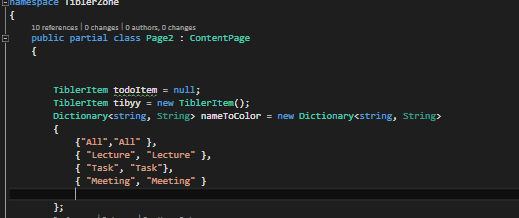
{"All","All" },

{ "Lecture", "Lecture" },

{ "Task", "Task"},

{ "Meeting", "Meeting" }

It would look like this, once the code is added.



**Explanation**

This is populate the **picker** (AKA **dropdown bar**) to allow you to sort between the different types of tasks.

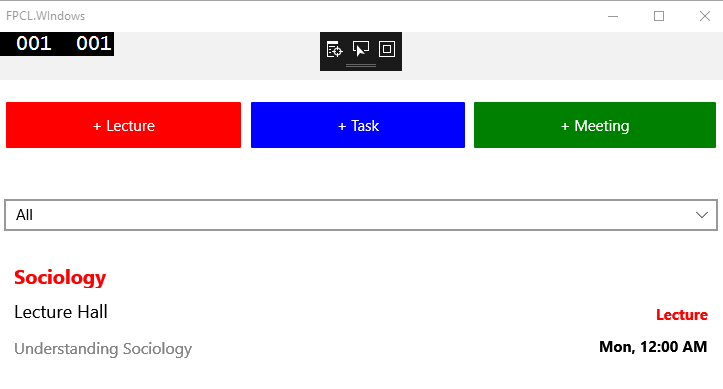
10) Select TiblerZone.UWP & Local Machine

**Click on Local Machine to run it**

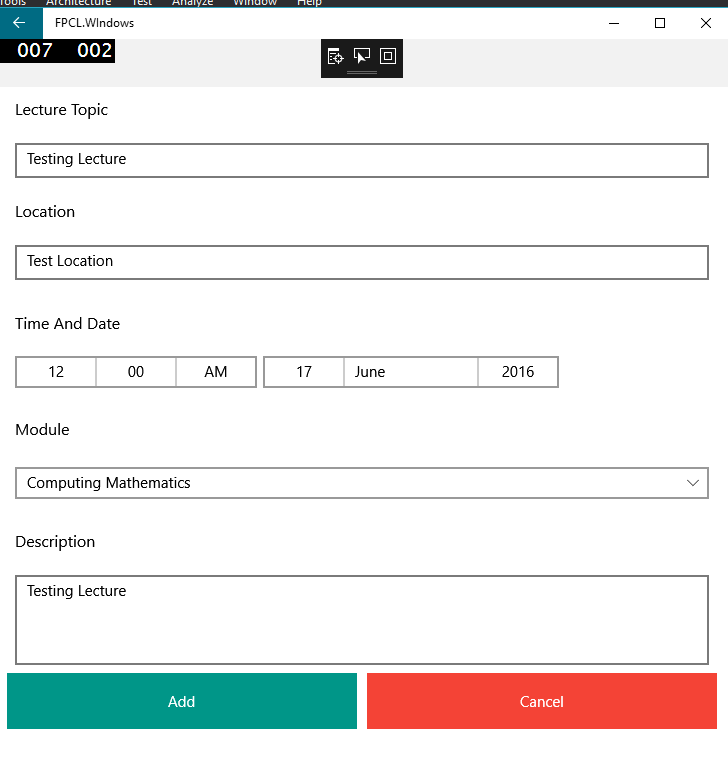


11) Populate the List View and SQLite Database

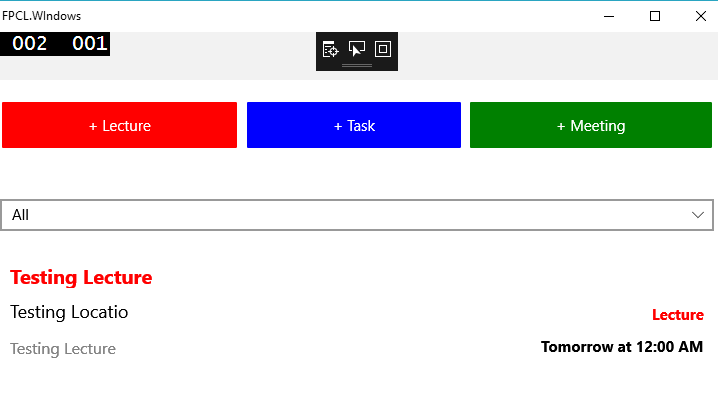
**Add New Items such like new Task, Meetings and Lecture**



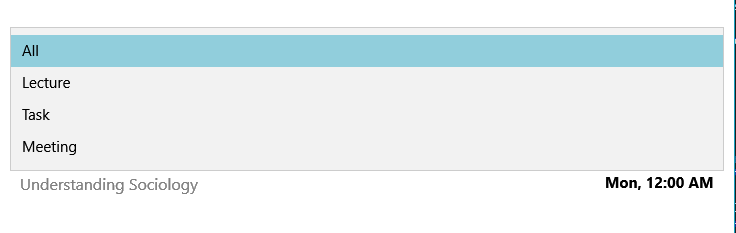
12) Enter the various details and click on the **Add button**



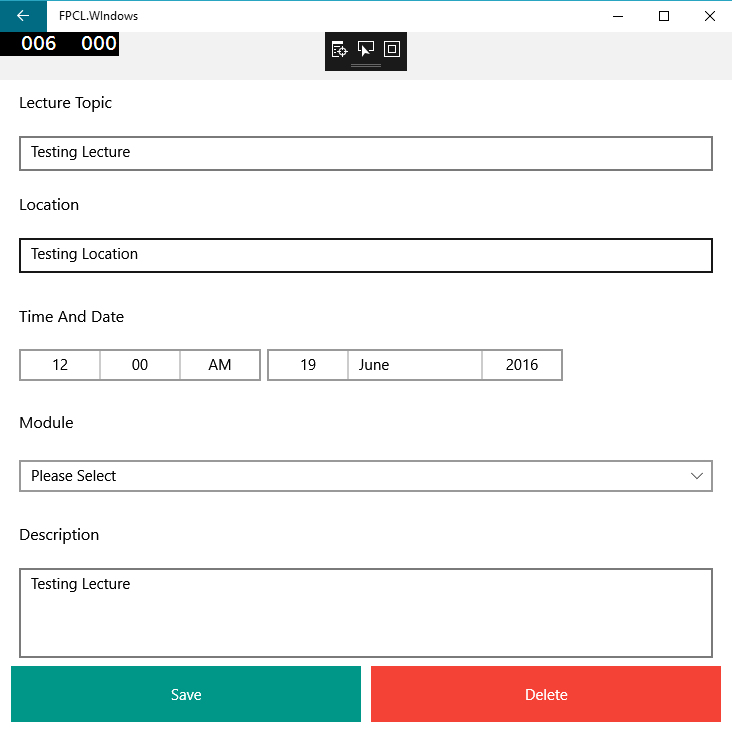
Now you will see **list view being populated** with the input you entered just now



You can now also **sort the items** by its type (All, Lecture, Task and or Meeting)



If you tap on any item on the list view it will then **redirect you to display details** of the items. This page allows you to **modify or delete** the list view

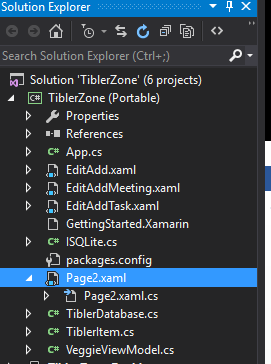


13) **Close** the Tibler App

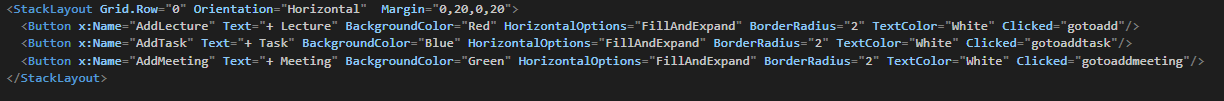
**Modifying USER INTERFACE**

Now we are going to make some changes to the User Interface of the application

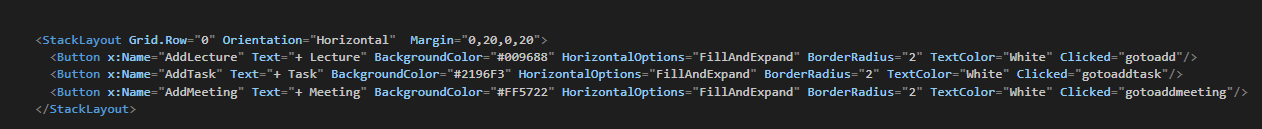
14) Open **Page2.xaml**



15) Change the Background color from **Red, Blue and Green** to #009688, #2196F3 #FF5722. This can be found in the **Stack Layout Tag**



It will look like this one it is completed



**Explanation**

This is done so as to change the colour of the 3 buttons (+ Lecture, + Task, +Meeting)

16) Open **Page2.xaml.cs**

17) **Change all**

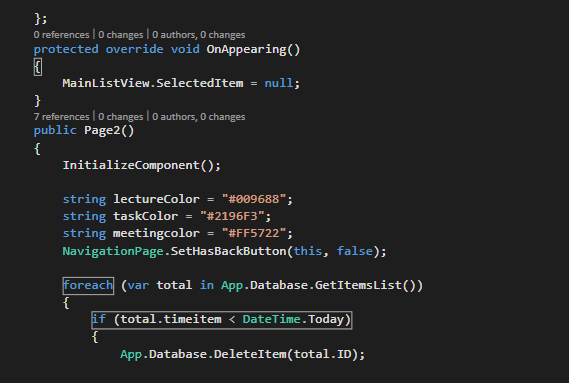
string lectureColor = "Red" TO total.color="#009688"

string taskColor = "Green" TO total.color="#2196F3"

string meetingColor = "Blue" TO total.color="#FF5722"

**\*Hint: Use ctrl+F to find the three strings**

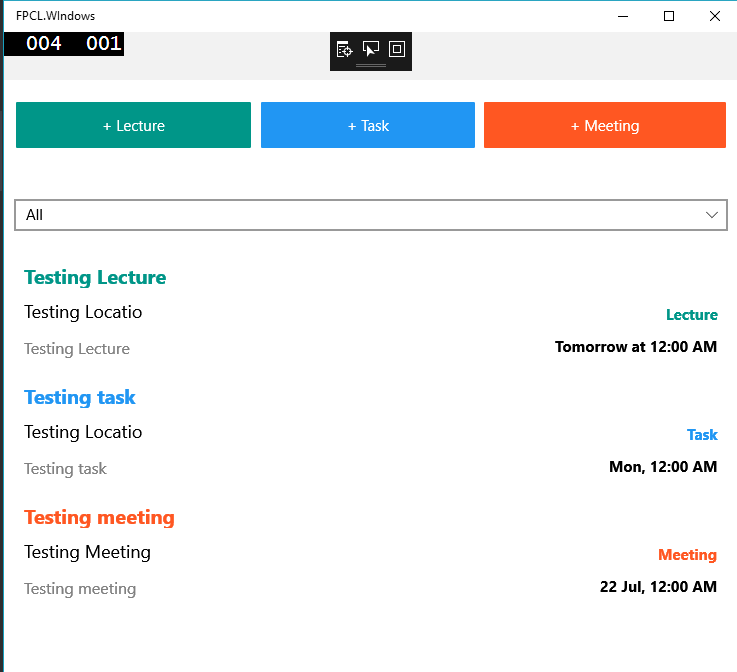
The part of the code will look like this one it is completed.



**Explanation**

This is done to change the Colour of the 2 text controlsfor the different types of items **(Lecture, Meeting & Task)**

The App will look like this once it is completed

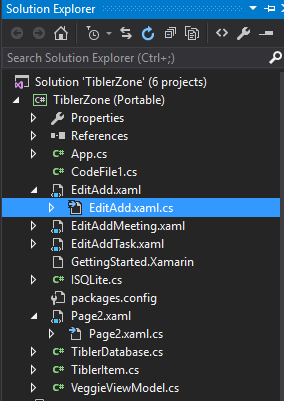


Notice that the color of the buttons and text change. Now, each types of items added to the database & displayed on list view will follow the color of its type (**Lecture, Task & Meeting**).

**CREATING NEW APP LOGIC**

We are going to add some basic app logic for the app. specifically, we are going to add some codes to validate the user inputs such as validating the date input & many more.

18) Open **EditAdd.xaml.cs**



19) **Copy** the code below

if (tibleritemsave.Name == "" || tibleritemsave.Name == null)

{

DisplayAlert("Invalid Name", "Please Enter Valid Name ", "OK");

}

else if (tibleritemsave.Location == "" || tibleritemsave.Location == null)

{

DisplayAlert("Invalid Location", "Please Enter Valid Location ", "OK");

}

else if (tibleritemsave.Description == "Please Enter Description Here" || tibleritemsave.Description == null)

{

DisplayAlert("Invalid Decription", "Please Enter Valid Description ", "OK");

}

else if (tibleritemsave.timeitem.Date < DateTime.Today)

{

DisplayAlert("Invalid Date & Time", "Please Enter Future Date & Time ", "OK");

}

else if (tibleritemsave.timeitem.Date == DateTime.Today && tibleritemsave.timeitem.TimeOfDay < DateTime.Now.TimeOfDay)

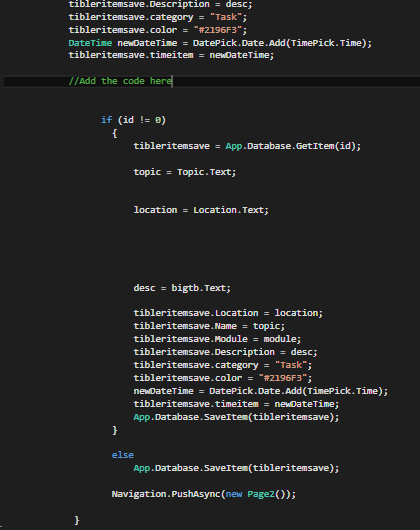
{

DisplayAlert("Invalid Date & Time", "Please Enter Future Date & Time ", "OK");

}

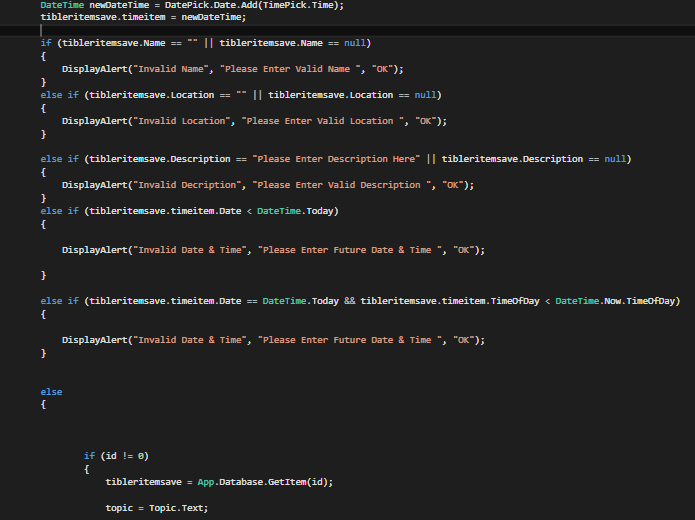
20) **Paste the code that you have just copied**

See below for reference



Add the code here

The Code will look like this once it is completed



21)Add an **Else Statement** as shown below

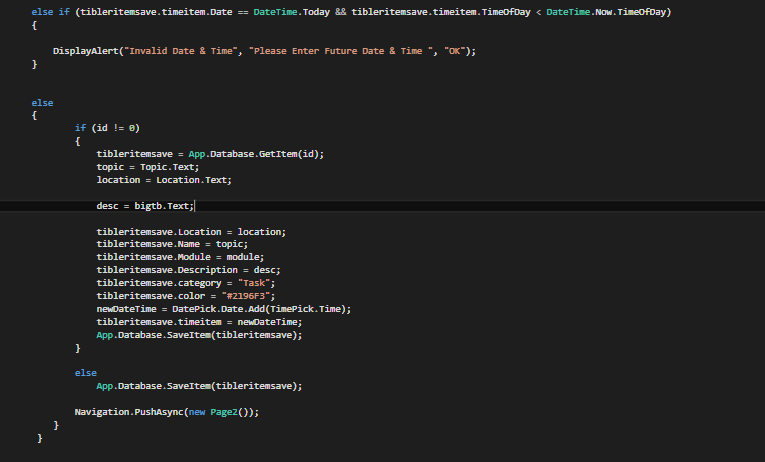
else

{

\*Existing codes remain inside

}

The Code will look like this once it is completed



22)Now do the same for **EditAddTask.xaml.cs** and **EditAddMeeting.xaml.cs (Step 18-20)**

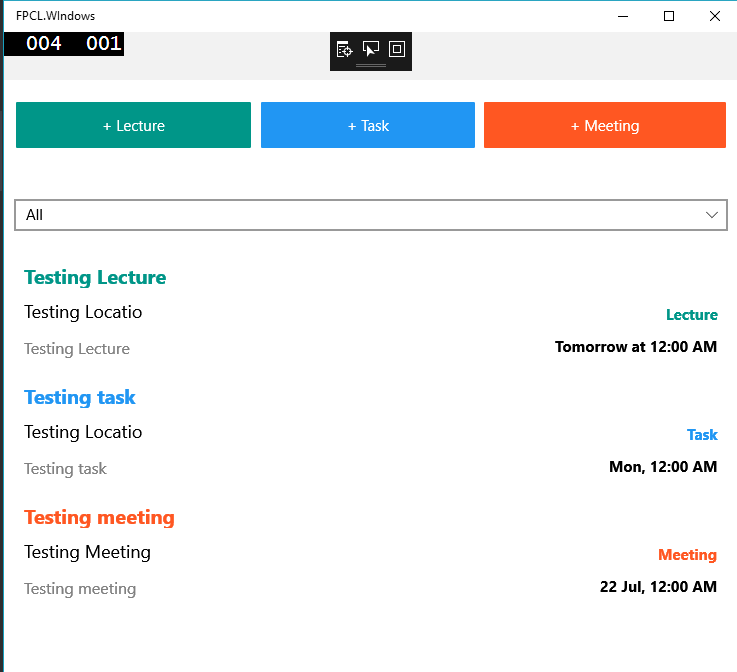
**Explanation**

Now every time before you add a task, meeting or lecture. The app will validate the data being input. For example, the date for the item must be in the future and not a date in the past OR the name of the item couldn’t be left blank.

23) **Run** the application



24) Click on **+ Lecture**



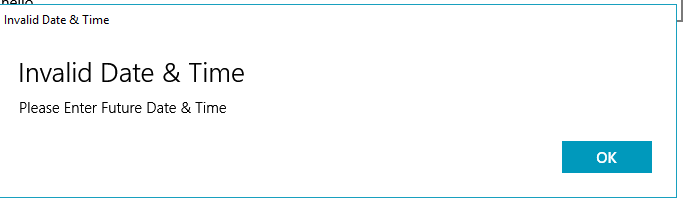
25) Try leaving the name blank or change the date to the past.

**You will receive a prompt to re-enter the details**

This is the prompt for leaving the name as blank



This is the prompt for choosing a date in the past



You are required to enter a future date or time. Even if it is the same day, but if you entered a time in the past, you would still receive this prompt.

Download Completed codes here:

<https://github.com/izzkhair/Xamarin-CompletedHandsOnLab>

**After completing all the steps above,**

You should be able to:

* Navigate between multiple pages
* Calling C# methods for events occurrence from a XAML file
* Modify User Interface from the C# or backend code file
* Modify User Interface in XAML
* Create your own app logic
* Of course, Create a **Xamarin Forms** Application of your own

I hope that through this app that I have built, you could use it as a resource for you to learn more about Xamarin or C# programming language. **You are allowed to modify or improve** the app further more to your likings☺. If there is any problem or doubts, feel free to email me at [**a-momoha@microsoft.com**](mailto:a-momoha@microsoft.com)**.**

**Mohamed Izzat Khair**

**Technical Evangelist**