Xamarin Forms Test

Introduction

This Exam will test your aptitude in Xamarin Forms. You should attempt to complete each of the requirements listed in each section. If you're unable to complete a requirement, skip it and come back later.

The aim is to create a fully functional app that runs on both Android and iOS with as little platform specific code as possible.

The solution includes a PCL for the Core code as recommended by Xamarin.

A simple MVVM framework is in place consisting of Models, Views and ViewModels in their relevant directories.

It's preferable to use Data Binding to display and update values.

INotifyPropertyChanged has been implemented on the ViewModels and Models so you'll have to ensure you're correctly notifying the changes to properties. Feel free to add Fody.PropertyChanged if you'd prefer.

Use App's public Get property to access the current instance of App, which allows you to use navigation from the ViewModels by accessing the MainPage property of the App.

Setup

Ensure you have Android SDK 19 and 27 installed.

Add the *FFImageLoading Forms Library* to the solution: https://github.com/luberda-molinet/FFImageLoading

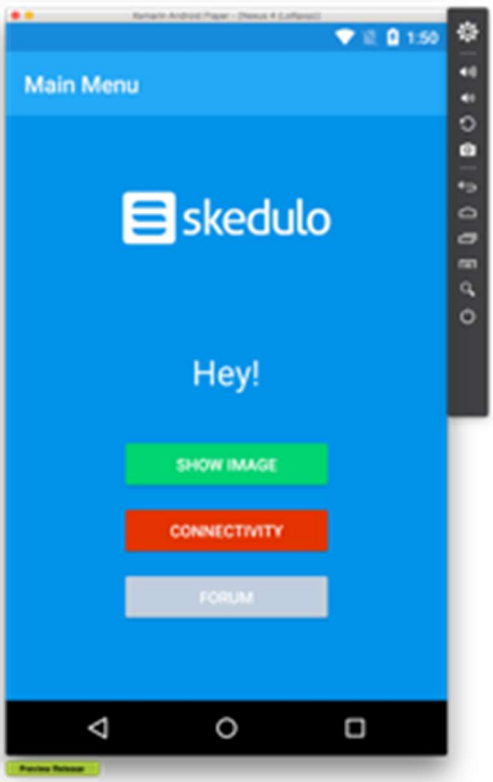
Add the *ConnectivityPlugin* to the solution: https://github.com/jamesmontemagno/ConnectivityPlugin

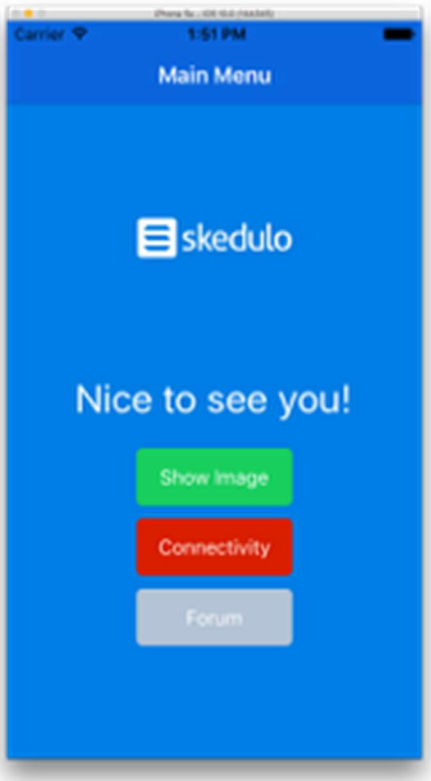
Main Menu

You have been given the following screenshots for Android and iOS. You must construct the interface using Xaml by editing the existing 'ViewMainMenu'. You should aim to duplicate the style as closely as possible.

The Logo asset has been provided in the Assets directory. You should add this to the project so that it can be loaded on Android and iOS. The relevant platform dependant resources have been included.

The content is centered vertically on the view and it must adapt to orientation changes, including scrolling if the view vertical size is too small to display the content.





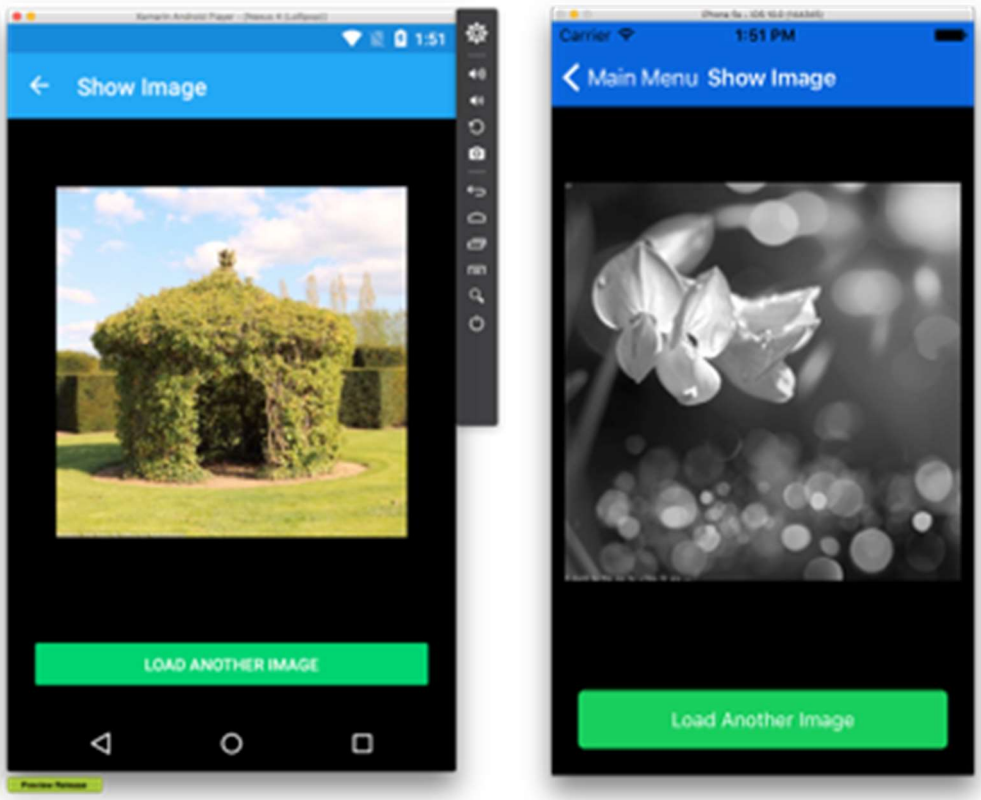
1. The background colour is #007EE5
2. The logo should be positioned with about 80 units of space between it and the greeting.
3. There should be a **horizontal** space of 100 units surrounding the Logo and the buttons to the edge of the screen and the elements should scale according to the view size. I.e. don't fix the width of the elements.
4. The colours of the Buttons are #19CF5E, #D91D00 and #B4C3D5 in order, top to bottom.
5. The Greeting label should have a text size of at least 30. The Greeting should be dynamic according to the time of day. If it's between 12am and 10am, it should show 'Good morning!'. Between 10am and 2pm it should display a random greeting of the following 'Hello!', 'Hi!', 'Hey!'. Between 2pm and 4pm it should display 'Good afternoon!'. From 6pm and onwards it should display 'Good evening!'
6. Tapping the Show Image button should show ViewShowImage
7. Tapping the Connectivity button will show ViewConnectivity
8. Tapping the Forum button will show ViewForum

9. You should be able to scroll the view up and down if there's insufficient vertical view space to display all the controls.

Show Image

The Show Image View will load 'simple.jpg' from lorenflickr.com. If you pass it another filename, it will load a different image (Passing a unique filename will ensure a new image is loaded).

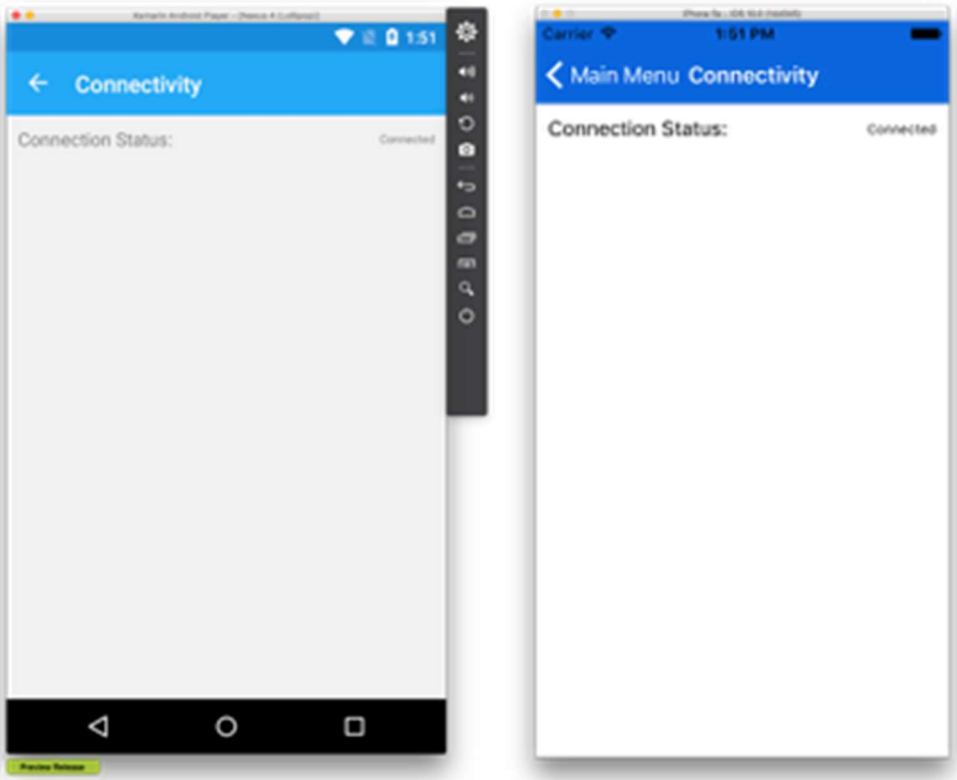
Eventually this view will allow a gallery view of images. Right now the Image control that's used will load the image from scratch each time, meaning the user's bandwidth is being used up. Instead we can use a plugin called 'FFimageLoading' to cache the image locally.



1. Replace the Image control with an FFImageLoading CachedImage control from the FFImageLoading package. This provides cached loading of images and animates the image in after it's loaded. The plugin's github has instructions for ensuring the control renders with Forms, similar to other plugins.
2. Modify the View so that tapping the 'Load Another Image' button forcefully loads a new image.

Connectivity

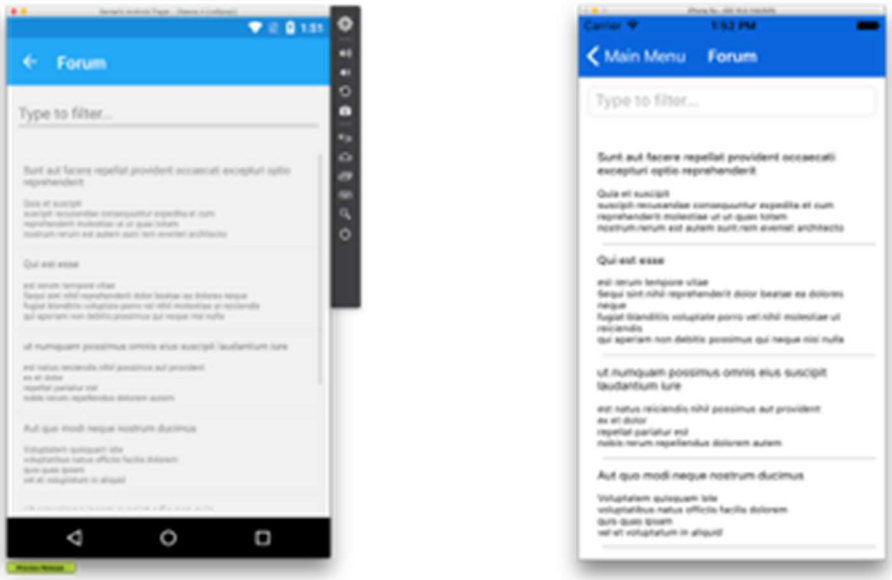
It's handy to detect connectivity changes. This feature requires native platform changes. Fortunately someone has already created a cross platform plugin that supports these changes called 'ConnectivityPlugin'



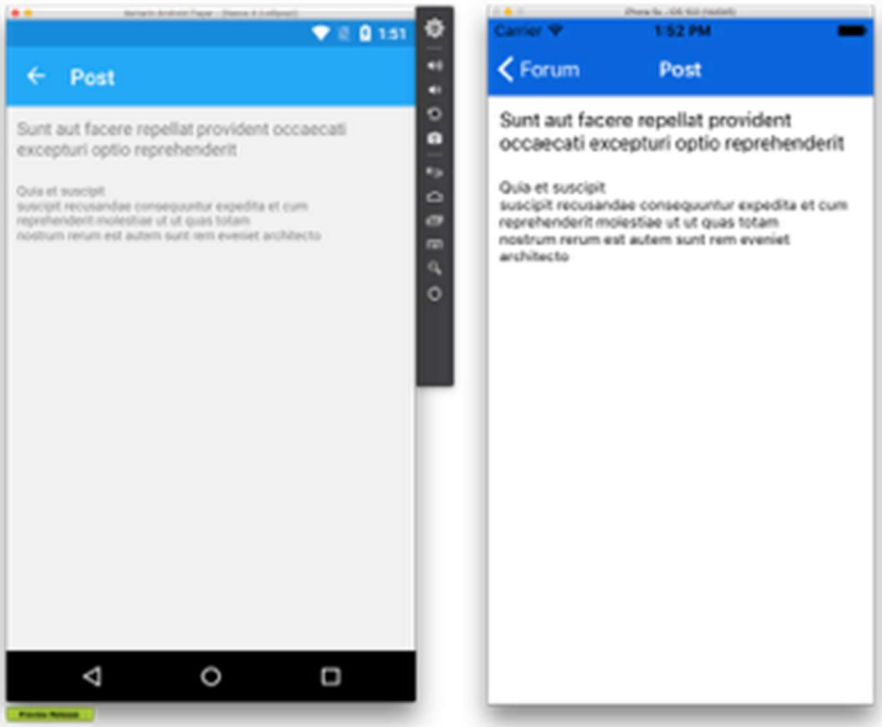
1. Update the ConnectionStatus property with the current connectivity state when the View shows. It should show 'Connected' when connected and 'Disconnected' when disconnected.
2. Ensure you update ConnectionStatus when the connectivity changes and ensure the interface updates. Test this works by toggling flight mode or terminating your internet connection.

Forum

The forum is used to display posts from users. Currently it displays all posts and doesn't provide a way to view additional details.



1. Modify the Forum view to filter the Posts that are visible when you enter text into the text box. The results should update immediately and you should ensure you allow case insensitive searching. You will have to update ServiceForum to filter the posts. It's recommended to use LINQ to do this if you can.
2. Tapping a Post in the list should open ViewPost and display the Title and Body from the Post. You will need to inform ViewPost about the chosen Post model and then ensure the view displays the Title and Body.



Bonus Challenges

These are additional challenges for which you can obtain bonus points and show off your skills.

1. Modify the Main menu and ensure that the NavigationBar on iOS is styled with the colour #0C667A and the Title text is #FFFFFF. This requires Native customisation on iOS.
2. Modify ServiceForum so that Id's for posts are assigned sequentially and dynamically instead of hardcoded and aren't passed into the constructor.
3. Modify ViewPost to include a field for a User name. Create a ServiceUser that has 2 users, "Fred" and "Wilma" with id's 1 and 2 respectively. Lookup these User names dynamically using the UserId property from ModelPost and then display this name in the new User name field you added to ViewPost.