2016

Evaluation of Xamarin Forms for MultiPlatform Mobile Application Development

Amer A. Radi
Grand Valley State University

Evaluation of Xamarin Forms for Multi-Platform Mobile Application Development

By
Amer A. Radi
05, 2016

Evaluation of Xamarin Forms for Multi-Platform Mobile Application Development

By
Amer A. Radi

A project submitted in partial fulfillment of the requirements for the degree of

Master of Science in

Computer Information Systems

At

Grand Valley State University

05, 2016

Dr. Robert Adams **5/1/2016**

Table of Contents

Abstract

The main purpose of this project is to explore Xamarin Forms: How good is Xamarin at providing the "write once run anywhere" model? How does it handles architectural differences? How Xamarin hides details of the platform, and are there areas where the developer has to know platform differences?

The high level goal of this project is to be able to build a simple application that has a single code base in one language for several mobile platforms using Xamarin forms.

The primary focus is on native app development with Android, iOS and Windows Phone, and not on hybrid or mobile web development.

1. Introduction

The world of mobile phone applications has been growing at a phenomenal rate dominated by three major platforms: Apple iOS, Google Android, and Microsoft Windows. In order to attract a large number of users to an application, developers should target more than one platform, if not all of them. However this is not an easy task. The three platforms have different operating systems, development languages, APIs, application life cycle models, local storage, and caching strategies. Implementing an application for all three platforms essentially requires the application to be developed three times.

A new kind of development framework is emerging that claims to solve this development hurdle. The most well known is Xamarin. By using a cross-platform framework like Xamarin, developers can create applications for multiple platforms using the same code base. Xamarin claims it is easy, fast, and efficient.

The purpose of this project is to explore the Xamarin framework to determine if it is truly a “write-once-run-anywhere” framework. The methodology used to test Xamarin is to develop a simple mobile application that exercises common features seen in many mobile applications. Furthermore, our project explores several hardware-centric features such as accessing the GPS. The goal is to provide an unbiased assessment for mobile developers.

2. Xamarin Forms Overview

Xamarin is a application framework solution (Figure 1) that helps developers create cross platform mobile applications for Android, iOS, and Windows easily and rapidly. Using Xamarin, developers create their mobile application by creating a single code base written in C#. With Xamarin developers can use all C# language features such as Generics and Async, and at the same time have the ability to access any mobile specific features and build a native UI. The Xamarin library provides many common controls and layouts that are mapped directly to the native ones at run time, so that applications look fully native.

Figure 1: Xamarin forms mapped to native OS libraries.

Reprinted from Xamarin Mobile Application Development (P.), by Dan Hemes ,2015. Apress

There are two approaches to using Xamarin forms. The first and most common one is called “portable class libraries”. With this approach a library is created that contains the application logic, and targets the specific platforms that the application will support. It may also include interfaces to provide a specific functionality for a specific platform. The other approach is called “shared projects”. With shared projects developers organize code in a shared asset project and use #if compiler directives to specify the functionality for a specific platform (see Figure 2).

```
#if __MOBILE__
// Xamarin iOS or Android-specific code
#endif

#if __IOS__
// iOS-specific code
#endif

#if __TVOS__
// tv-specific stuff
#endif

#if __WATCHOS__
// watch-specific stuff
#endif


#if __ANDROID__
// Android-specific code
#endif

#if __ANDROID_11__
// code that should only run on Android 3.0 Honeycomb or newer
#endif
```

Figure 2: shared projects strategy,#if compiler directives

Xamarin forms provides a template that contains two components required for any application: an “Application” component which is responsible of initializing the application, and “Page” component that represents a single screen in the application. Xamarin forms provides different templates for pages to fit the developer’s needs. Each template has its own specific visualization and behavior (see Figure 3).

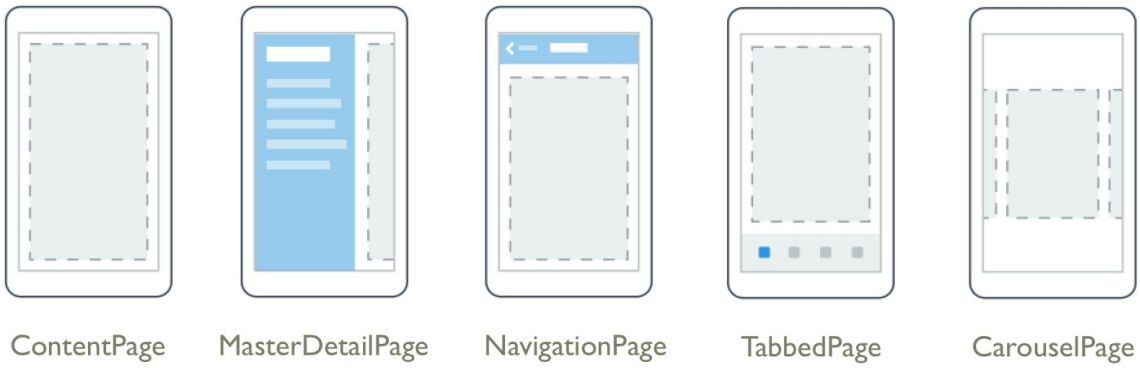

Figure 3: different page templates provided by Xamarin forms

Xamarin also provides View classes that provide most of the standard visual controls such as Entry, Frame, and Box View. Naturally, these can be adjusted visually by using their properties such as background color, and text color. It also allow developers to control their behaviors, and use events such as a text-changed event (something very familiar to most .Net developers). Finally, for each of these controls Xamarin defines a renderer that creates a native representation of it at the run time (Figure 4).

Figure 4: Xamarin forms render role

3. Application Design

3.1 Requirement Specification

As mentioned above, the methodology used to evaluate Xamarin is to develop a simple mobile application that exercises common features seen in many mobile applications. The overall goal is to to determine how well Xamarin supports the "write-once-run-anywhere" model. Therefore, the first step was to identify the required features. Our application is a simple inventory adjustments application that has the following features:

- View a list of items with their costs.
- Add, edit and delete items.
- View the available quantity of a specific items in all warehouses.
- Adjust in: add a specific quantity of an item to a warehouse.
- Adjust out: remove a specific quantity of an item from a warehouse.
- Transfer: transfer a specific quantity of an item from one warehouse to another.

These features will require the application pull data from a database (either local or remote), display that information, allow the user to drill-down to find more detail, and allow the user to modify the information. These features are very typical of many mobile applications such as Instagram, Evernote, Google Mail, etc.

3.2 User Interface Design

The application is constructed with different views that the user can navigate through. The home screen displays an overview of all available items with their cost, along with the ability to add a new item (Figure 5). Note that we are not concerned about styling the application. Rather, our concern is on features, not design.

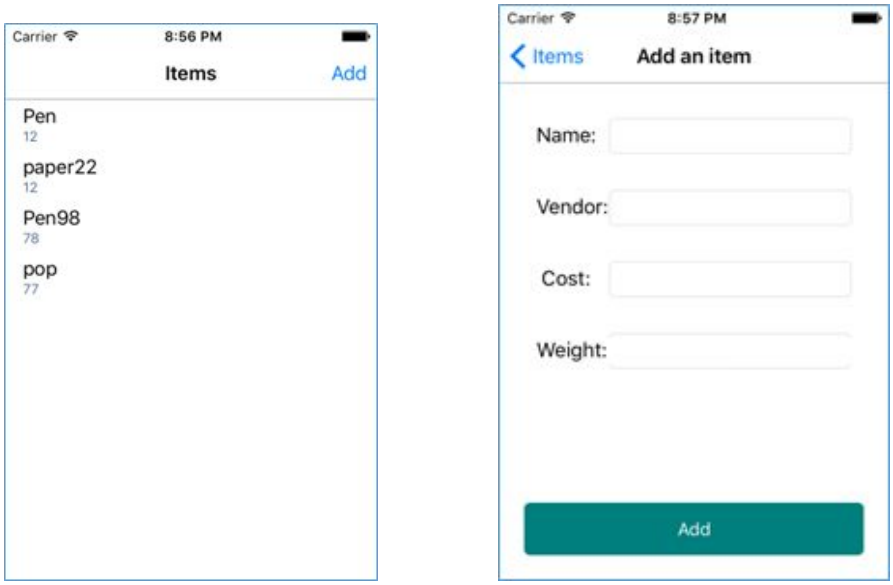

Figure 5: the home Page and Add page

If an item is selected from the list , the item properties page is displayed where the properties of this item can be edited (Figure 6).

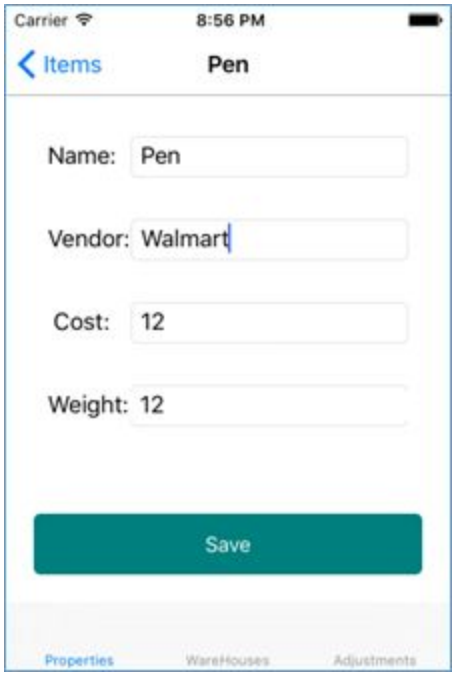

Figure 6: Edit item Page

Figure 6 also shows that a set of tabs is displayed at the bottom of the Edit screen. The Warehouses tab shows the quantity of this item in various warehouses (Figure 7).

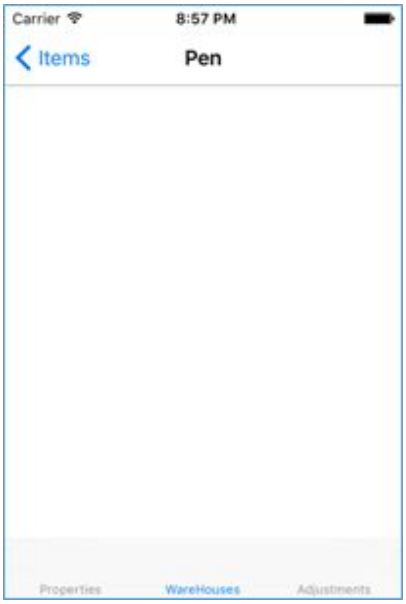

Figure 7: Item/Warehouse Page

The Adjustments tab displays the Inventory Adjustment screen which permits the user to change the quantity of the item in a warehouse (Figure 8).

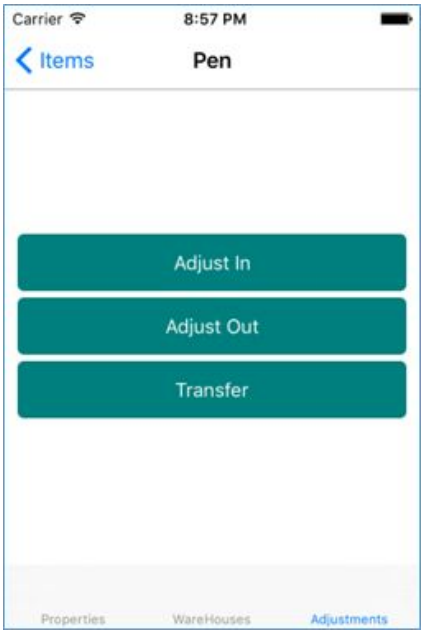

Figure 8: Inventory Adjustment Page

The Inventory Adjustment page also provides access to different Adjustment pages (Figure 9).

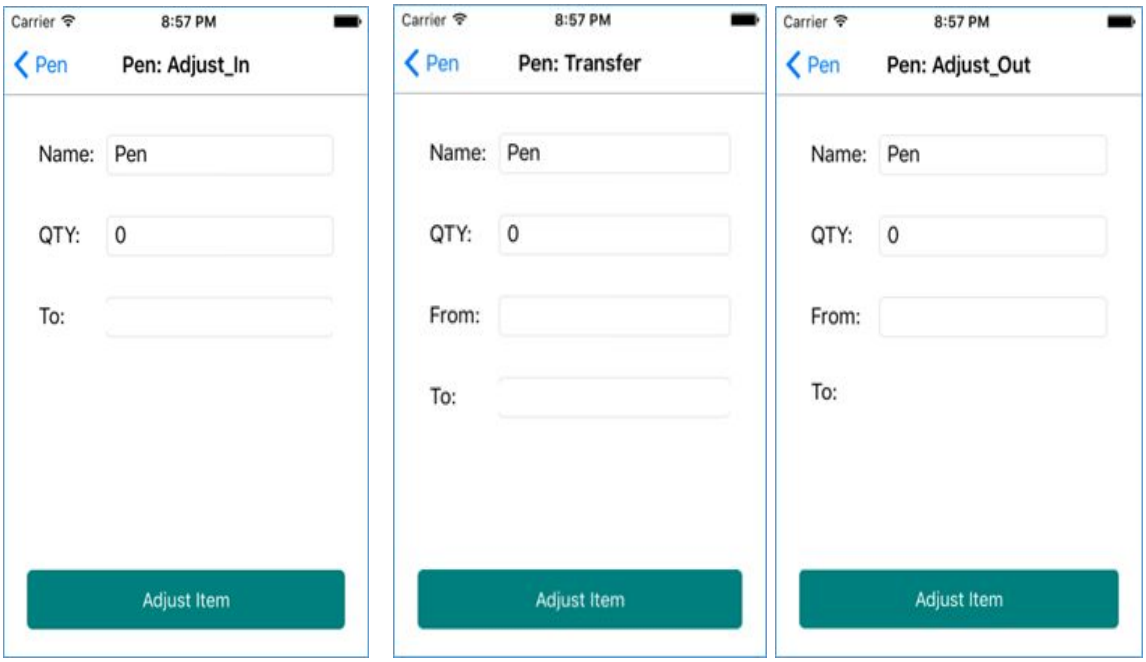

Figure 9: Different Adjustment Pages

3.3 Data Models:

On the back end the application needs 3 data models:

1- Inventory Model: identifies the item and all its properties :

- Id -string unique
- Name-string
- Cost-double
- Vendor-string
- Weight-double

2- Warehouse Model: identifies the warehouse and its properties:

- Id -string
- Name-string
- Location-string

3- Inventory-Warehouse Model: identifies the relationship between item model and warehouse model:

- Id-string
- Item id-string
- warehouse id-string
- quantity-integer

4. Application Implementation

The application is implemented using the Portable Class Library strategy as described in section 1. The code is divided into two main parts:

- Shared Code: a shared project that contains the common code to build the user interface, and manage the data models and storage across all mobile platforms.
- Platform Specific Code: This part of the project contains platform-specific UI customization (e.g., background color). This is also where platform-specific code is placed that Xamarin currently does not abstract. For example, code that plays a sound is currently platform-specific.

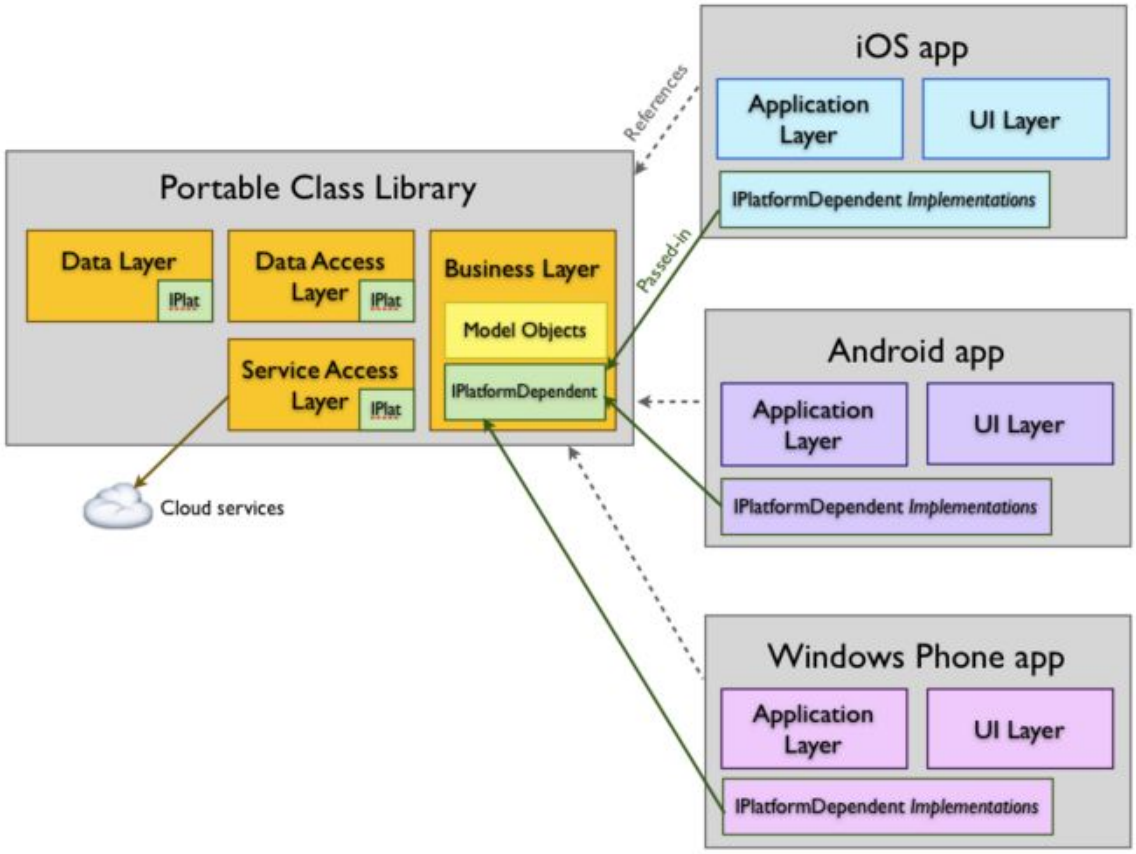

Figure 10: The general structure of a cross platform application

4.4.1 The Shared Code

The shared code library (Figure 10) contains all the database code as well as the data access layer code. Each native application will reference this shared code when it is deployed.

The data layer contains the code that identifies the database models of the application. It contains 3 classes that identify the objects that are used to store data on Parse. Each Parse object contains key-value pairs of JSON-compatible data.

The Data Service Access Layer operates between the data layer and the Parse server. Parse does not provide a unique SDK for different platforms. In order to reduce the need for each class to interact with the Parse server, this layer provides that functionality.

The Data Access Layer encapsulates the methods that are used between the UI and the data. The UI will use methods that encapsulate Parse objects.

The Business Layer consists of classes, one for each screen of our application. Their purpose is to act as a controller for a particular screen.

4.4.2 The Platform Specific Code

Xamarin forms provide two main techniques to customize the UI appearance and behavior and to deal with the need of difference implementation in different platforms.

a. Custom Renderers

Xamarin Forms allow the developer to describe the UI once using a shared set of controls while still rendering a native UI. It provides “Custom Renderers” (Figure 11) which allow the shared code to be overridden and customized in appearance and behavior on each platforms. It was used to customize the back button in the Inventory Adjustment Page in the iOS and Android apps. The default name of the button has been changed to “cancel”, and the action of the button has been overridden so that it displays a confirmation message for the user before it takes them back to the previous page. It has been also used to customize the appearance of Entry and Picker controls in iOS and Android.

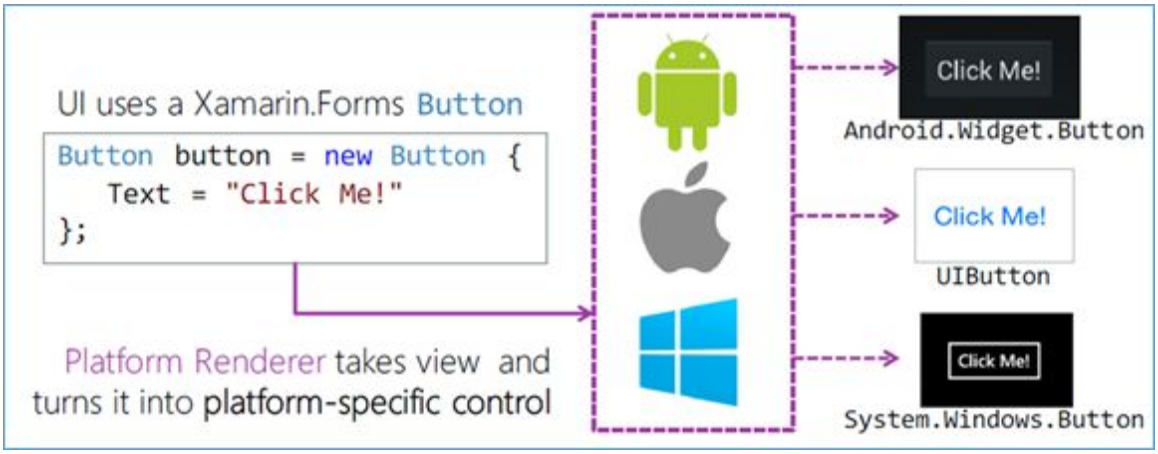

Figure 11: Custom Renderers technique

b. Dependency Service

As another alternative to customization Xamarin forms provides a service locator called the DependencyService (Figure 12). This locator is used to register the implementation of each platform UI and locate it through the abstraction used in the shared code. We used this technique in our app when dealing with many platform-specific features such as dealing with the database and some hardware features including camera, GPS, and playing sound.

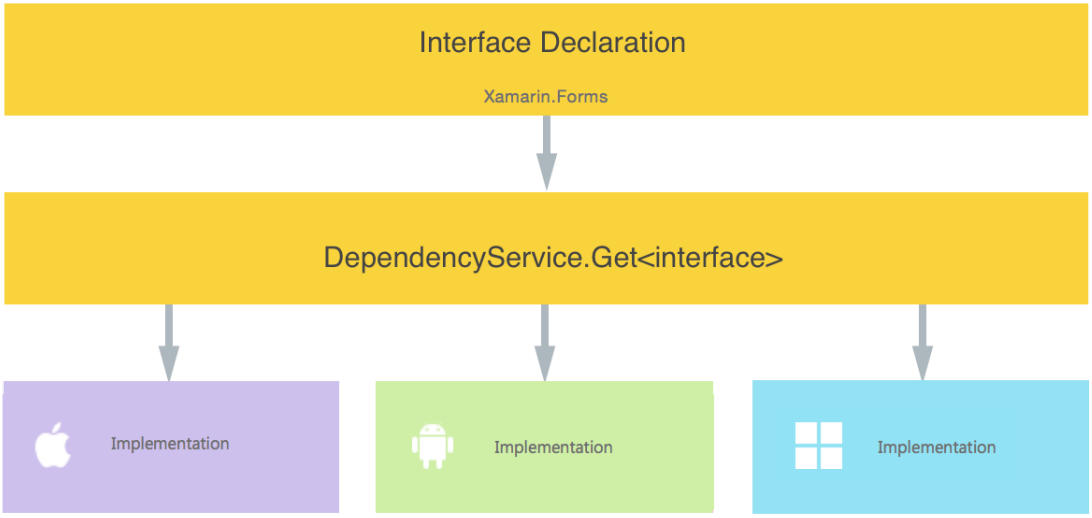

Figure 12: Dependency services

To use the Dependency Service we need two main components:

1- Interface: defines the required functions to implement on each platform.

2-Platform Implementation: each platform should have its own class which implement the functions defined in the interface.

Each Platform Implementation class must use meta-data attribute to register with DependencyService so that it can be found at the run time.

5. Supporting Technology

5.1 Parse

Parse is a back-end service that provides many cloud and mobile applications with services such as data storage, push notifications, and social media integration. It allows the developer to focus more on the front-end, and forget about server code or maintenance. Unfortunately, their cloud hosting services will be fully retired in January 2017. Instead, they are going to release the Parse server as open source software, and allow their back-end solution to be hosted individually by application developers.

5.2 Visual Studio

Xamarin forms provides a plug-in that can be installed in visual studio for mobile development, and it provides also its own Xamarin studio that can be used in both Windows and iOS environments. although the use of visual studio plug-in would be more expensive since it requires a business licence, it is more beneficial to use it than the Xamarin studio because you can take advantage of many of its plug-in that would facilitate and accelerate your development, also since it is a famous and old development tool ,you can get faster support answer and less bugs.

6. Testing and Results

Xamarin has three different options for testing: using on screen emulator, a real phone connected to the computer through USB cable, or through Xamarin Test Cloud. Each approach has its advantage and disadvantages. Xamarin Test Cloud provides an automatic mobile application testing with thousands options of mobile devices, however

the service price is probably too expensive (minimum cost $12k per year) for an independent developer.

On the other hand, while the emulator option can be considered as the easiest and cheapest way for testing, a real device can be much useful when it come to test touch interaction and response time, camera, GPS testing, and/or Android testing.

The Windows Phone emulator has different screen resolutions and runs smoothly. Touch can be used on the emulator screen if it has a touch screen. However if testing on a real device is needed, an unlocked phone is required.

The iOS emulator Xamarin uses the default one which comes with XCode.

For Android, there are two option: using the one which comes with the Java SDK, or the Android player which is developed by Xamarin. Android emulators are not efficient, therefore using an actual Android device for testing is much better.

7. Personal Reflection

Using Xamarin I was able to create a mobile application for Android, iOS, and Windows. During the project I met a lot of problems especially when I was implementing hardware-specific features (such as GPS) for a specific platform. Although the Xamarin community is growing, it is still not comparable to the communities or libraries of native platforms. It also lacks several libraries and features that the native platforms provide, and it could take too much time to implement simple features.

However, despite all these difficulties, I was able to answer the project question: Xamarin forms is an excellent tool for mobile development. It uses a single language and code for

different platforms which make it easier and faster to develop an application especially when using C# language features such as LINQ and Visual Studio development tools. In addition, it produces a fully binary compatible application that targets the required platform and looks and acts like the native one.

Xamarin is a good complement to native development, and can help make development faster, more efficient, and easy. Perhaps developers devalued it because of lack of knowledge and patience.

Looking ahead, Xamarin has been opened up to more developers since Microsoft announced that Xamarin will be free in every edition of Visual Studio, including Visual Studio Community Edition. Xamarin forms has a great future in the mobile development market that will give programmers the opportunity of not repeating their work. It might become the standard solution for cross platform mobile development.

References

[1] Creating Mobile Apps with Xamarin.Forms ,Microsoft Press, 2015.

[2] Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms By Daniel Hermes: Apress Edition, 2015

[3] Xamarin Cross-platform Application Development - Second Edition
By Jonathan Peppers : PACKT Edition ,2015

[4] Xamarin forms introduction.
https://developer.xamarin.com/guides/xamarin-forms/getting-started/ . Access 28-03-2016

[5] Xamarin iOS introduction. https://developer.xamarin.com/guides/ios/getting_started/ . Access 28-03-2015

[6] Xamarin Android introduction
.https://developer.xamarin.com/guides/android/getting_started/ . Access 28-03-2015

[7] .NET and Xamarin Guide for Parse. https://parse.com/docs/dotnet/guide
Access 28-03-2015