Indira Gandhi National Open University
School of Computer and Information Sciences

Commonwealth of Learning

INTRODUCTION TO MOBILE ARCHITECTURE

Software Development Tools

4

Introduction to Mobile Architecture

School of Computer and Information Sciences
Indira Gandhi National Open University

Commonwealth *of* Learning

Block

4

SOFTWARE DEVELOPMENT TOOLS

UNIT 12
Native Development Tools **5**

UNIT 13
Cross Platform Development Tools **23**

UNIT 14
Publishing Tools and Developer Program **39**

UNIT 15
Monetization **53**

INTRODUCTION TO MOBILE ARCHITECTURE

This course has been developed as part of the collaborative OER Course Development for ICT Skills initiative of the Commonwealth of Learning (COL). COL is an intergovernmental organisation created by Commonwealth Heads of Government to promote the development and sharing of open learning and distance education knowledge, resources and technologies.

Indira Gandhi National Open University (IGNOU), established by an Act of Parliament of India in 1985, has continuously striven to build an inclusive knowledge society through inclusive education. It has tried to increase the Gross Enrollment Ratio (GER) by offering high-quality teaching through the Open and Distance Learning (ODL) mode.

Course Team

Author:

Mr. Shailesh Kumar Shiva Kumar, Research Scholar (SOCIS, IGNOU) and Senior Technology Architect (Infosys Limited) , Bangalore, India

Content Editor: Dr.P.Venkata Suresh, SOCIS, IGNOU, New Delhi, India

Content Design Team

Dr. P.V.Suresh,
Associate Professor & Director,
SOCIS, IGNOU, New Delhi

Mr. Amit Kumar Saxena,
Chief Technology Officer,
Grasp IO Inc, Bengaluru

Dr. S. Arumuga Perumal,
Associate Professor & Head of the
Department, S.T. Hindu College,
Nagercoil, Tamil Nadu

Prof.Parma Nand Astya,
Dean, School of Computing Science &
Engineering, Galgotias University,
Greater Noida

Mr. Shashi Bhushan Sharma,
Associate Professor, SOCIS,
IGNOU, New Delhi

Mr. Akshay Kumar,
Associate Professor, SOCIS,
IGNOU, New Delhi

Dr. Naveen Kumar,
Associate Professor, SOCIS,
IGNOU, New Delhi

Mr. M. P. Mishra,
Assistant Professor, SOCIS,
IGNOU, New Delhi

Dr. Sudhansh Sharma,
Assistant Professor, SOCIS,
IGNOU, New Delhi

Language Editors: Mr. P. Lakshmi Kantham , Faculty in English(Retd.), VSR & NVR College, Tenali, India

CRC Preparation: Tessa Media & Computers, New Delhi

Published by:

Indira Gandhi National Open University, Maidan Garhi, New Delhi-110068, India.

The course has been developed with the support of the Commonwealth of Learning, Canada.

Acknowledgements

Indira Gandhi National Open University wishes to thank those below for their contribution to development of this Course Material:

Staff of School of Computer and Information Sciences, Administration, Finance & Accounts Division and Electronic Media Production Centre, IGNOU, New Delhi, India and all External Experts. Special thanks to Dr.Sanjaya Mishra, Education Specialist(eLearning), Commonwealth of Learning, Canada for his support and guidance through out.

BLOCK 4 INTRODUCTION

This is the fourth and final block of the Course. This block introduces the learner to tools that can be used to develop software for mobile apps.

The process of software development has been rapidly changing since years. Initially, programmers used to write code. Then, we had 4GLs which can be used to run queries which mean that underlying code was already available. Also, we had tools which can be used to generate code if we drag and drop the elements that are part of our application.

With the advent of smart phones, several tools were developed for professionals to write apps. There are different types of tools. Some tools are developed by the developers of the concerned operating system. Then, there are tools that are developed by third party vendors that can be used to develop mobile apps for a particular mobile operating system.

In this block, our focus will be on the tools that can be used to develop software for mobile operating systems, namely, iOS, Android and Windows. Some tools were native to the operating system. Some are platform independent.

This block consists of four units and is organized as follows:

Unit 12 deals native development tools. It covers Android studio and Eclipse IDE for Android, Xcode and Swift for iOS, and C# and XAML for Windows.

Unit 13 deals with cross platform mobile app development tools. It covers Xamarin and PhoneGap.

Unit 14 deals with app publishing tools and Developer programs. It covers Google's Play Store, Apple's App Store and Window's Store.

Unit 15 deals with monetization. It deals with monetization of Android apps, iOS apps and Windows based mobile apps.

UNIT 12 NATIVE DEVELOPMENT TOOLS

Structure

12.0 Introduction

12.1 Objectives

12.2 Development Tools for Android

- 12.2.1 Android Studio
- 12.2.2 Eclipse IDE

12.3 Development Tools for iOS

- 12.3.1 Xcode
- 12.3.1 Swift

12.4 Development Tools for Windows Based Mobiles

- 12.4.1 C#
- 12.4.2 XAML

12.5 Summary

12.6 Solutions/Answers

12.7 Further Readings

12.0 INTRODUCTION

Developing mobile apps needs a good supporting eco system. Development tools and Integrated Development Environment (IDE) form an integral part of the support eco system. Development tools and IDEs provide all necessary services, editors, resources to quickly and efficiently develop the mobile app. They support the full lifecycle of the mobile app starting from development, design, testing, debugging and deployment.

In this unit, we will discuss various development tools for key mobile platforms. The learner needs to note that the system requirements etc. may change from time to time with different releases and hence needs to follow the instructions given on the portals while downloading and installing various software mentioned in this unit. Also, in the case of programming , latest syntax will apply.

12.1 OBJECTIVES

After going through this unit, you should be able to understand

- key concepts of Android Studio and Eclipse IDE for Android;
- migration from Eclipse IDE to Android Studio;
- concepts of Xcode IDE and Swift programming language for iOS apps development; and
- C# and Extensible Application Markup Language (XAML) for Windows mobile.

12.2 DEVELOPMENT TOOLS FOR ANDROID

In this section we will look at Android Studio and Eclipse IDE which can be used for development of Android based mobile apps.

12.2.1 Android Studio

Google provides Android studio for developing Android based mobile apps. Android studio uses Gradle build system. The following are the main features of the Android Studio:

- Build , debug and release variants are available.
- Code templates are provided.
- Drag and drop themes are available.
- Lint tool for performance optimization purposes.
- User interface design to preview layout for multiple screens.
- Provides data binding with model entities and UI elements.
- Android studio provides source control integration with many systems such as GIT, Apache Subversion (SVN), and Concurrent Versions System (CVS).
- Provides built-in cloud messaging.

Android monitor can be used along with Android Studio for performance monitoring and testing.

The following are various steps in setting up and running Android Studio:

a) Setting up Android Studio

Android Studio setup needs 4GB of RAM, 500GB disk space with JDK 7 or above. Detailed installation steps are given below:

i) Install Android studio from https://developer.android.com/sdk/index.html

ii) Update the SDK bundle using the Android SDK manager (Figure 12.1).

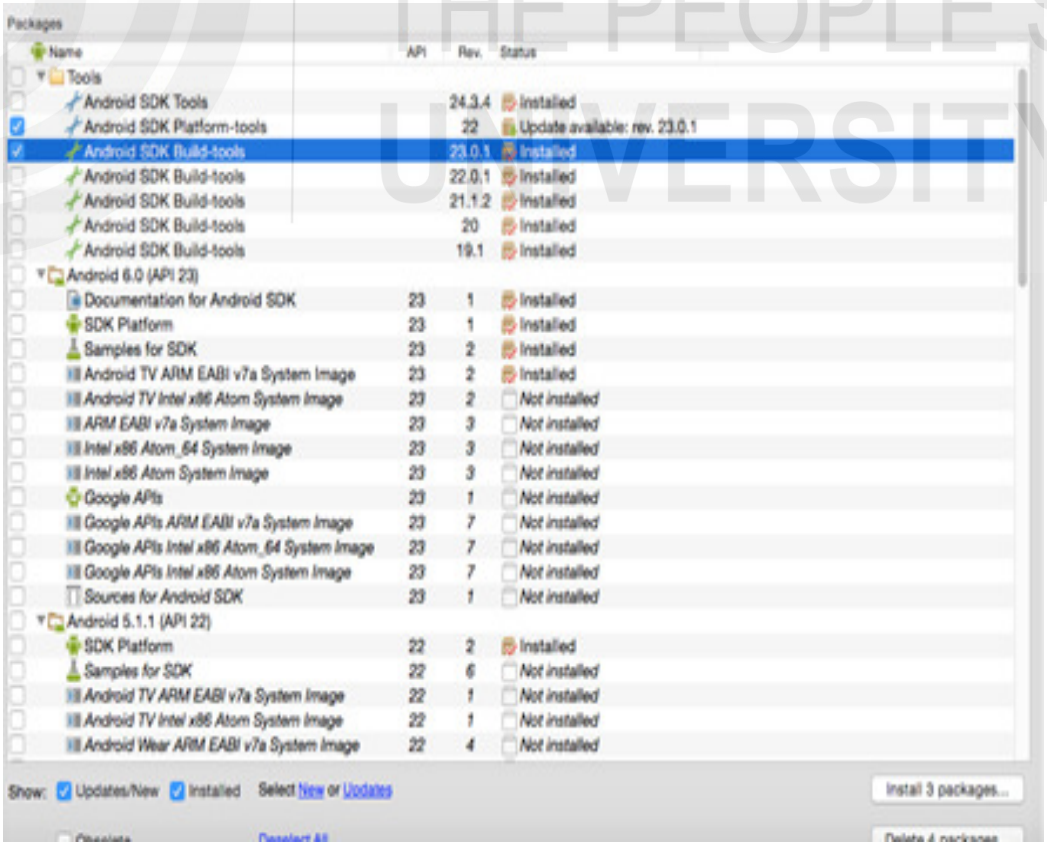

Figure 12.1 : Android SDK manager

iii) Download Gradle from https://gradle.org/gradle-download/ and configure Gradle in Android Studio at Android Studio → Preferences → Build, Execution, Deployment → Build tools → Gradle → select Use local gradle distribution → Gradle home → <Path to gradle file> (Figure 12.2).

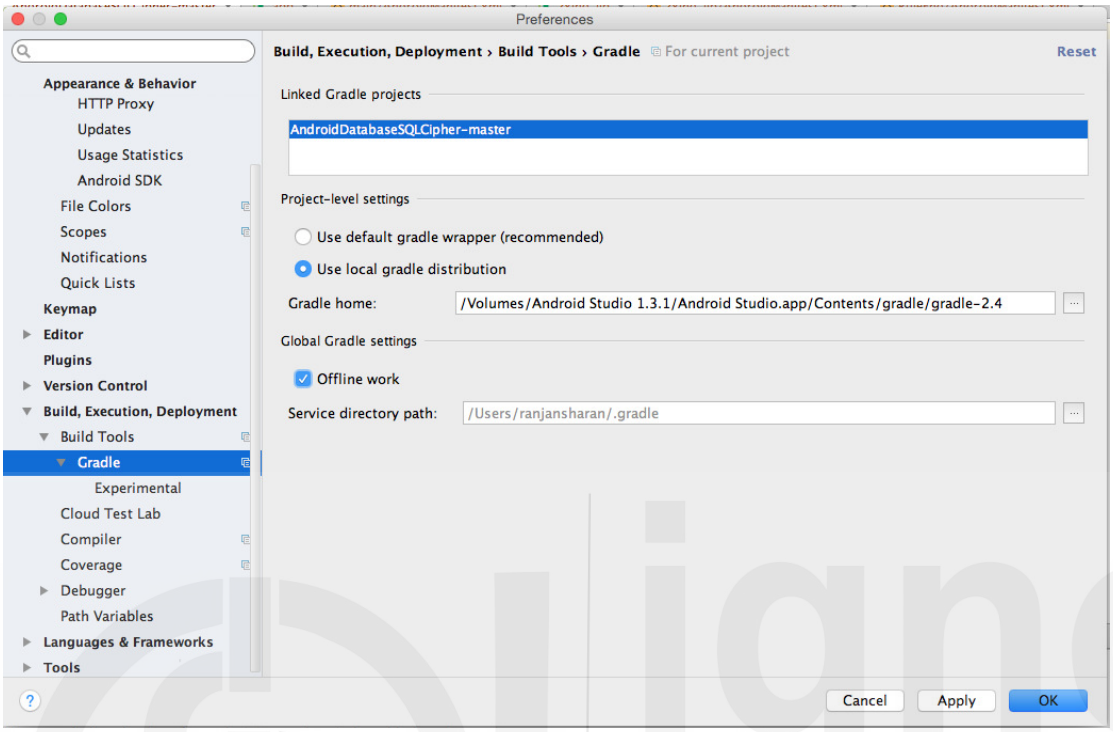

Figure 12.2 : Gradle in Android Studio

b) Running the Hello World project in Android Studio.

Let us look at steps for building the first app using Android studio. Before we start we need to check if JDK 7 is installed and if JDK 7 and Android Studio are configured. The following are the steps and screenshots for creating the first app mentioned above:

i) In Android Studio, go to File → New Project and create the application (Figure 12.3).

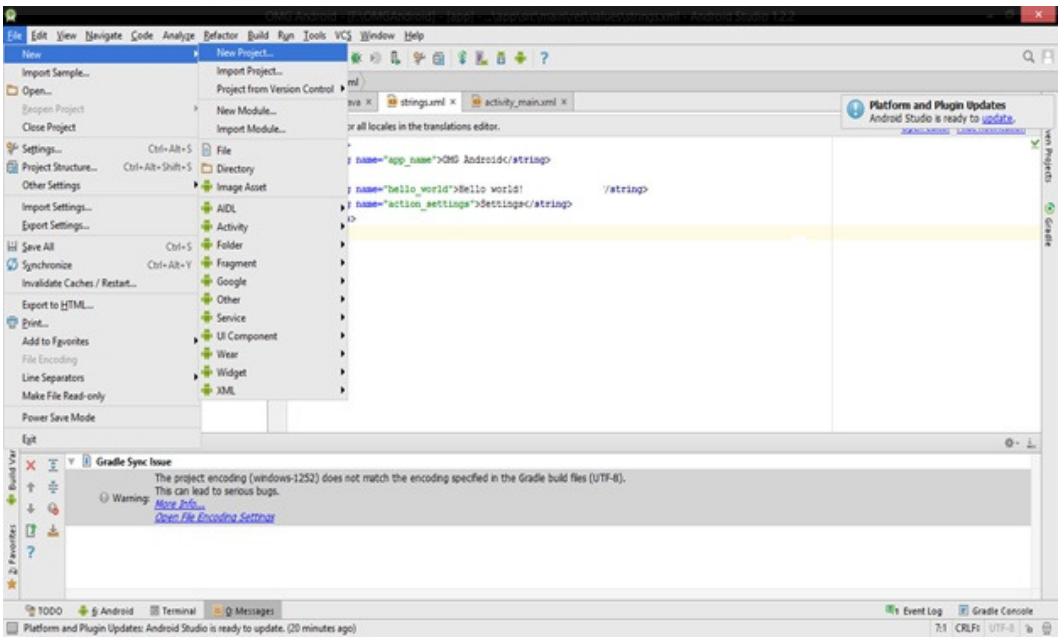

Figure 12.3 : Creating a new project in Android Studio

ii) Select the device and environment (Figure 12.4).

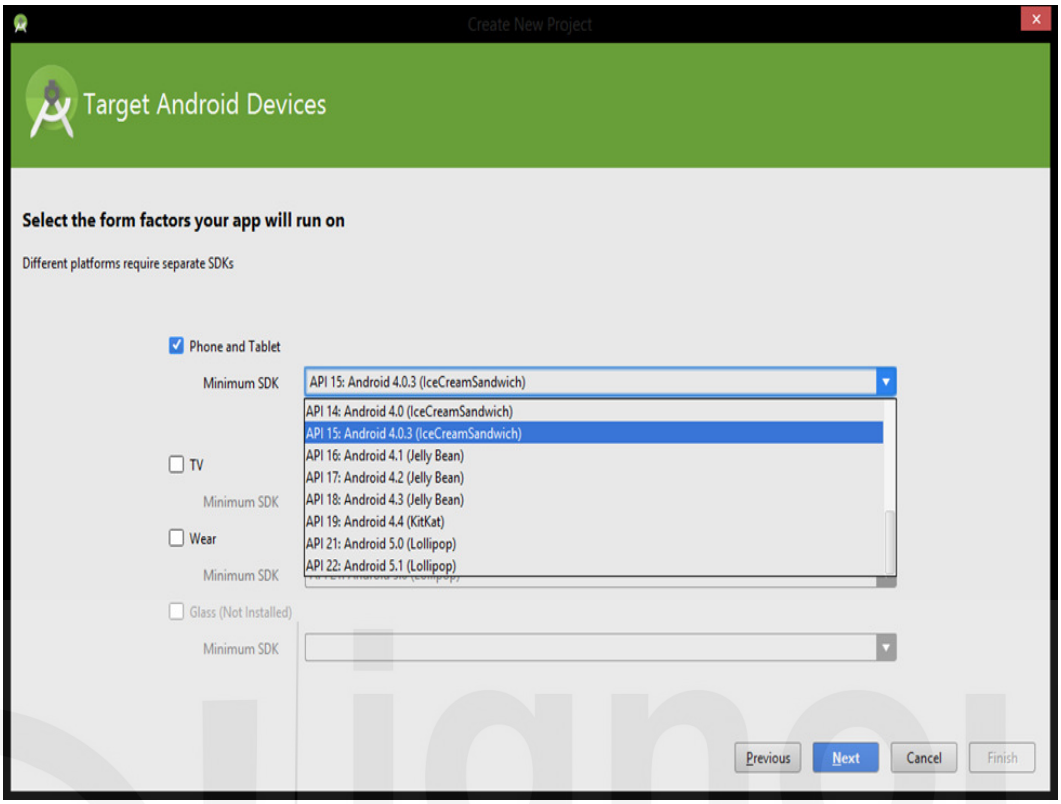

Figure 12.4 : Device selection in Android Studio

iii) Select the activity and rename it as required (Figure 12.5).

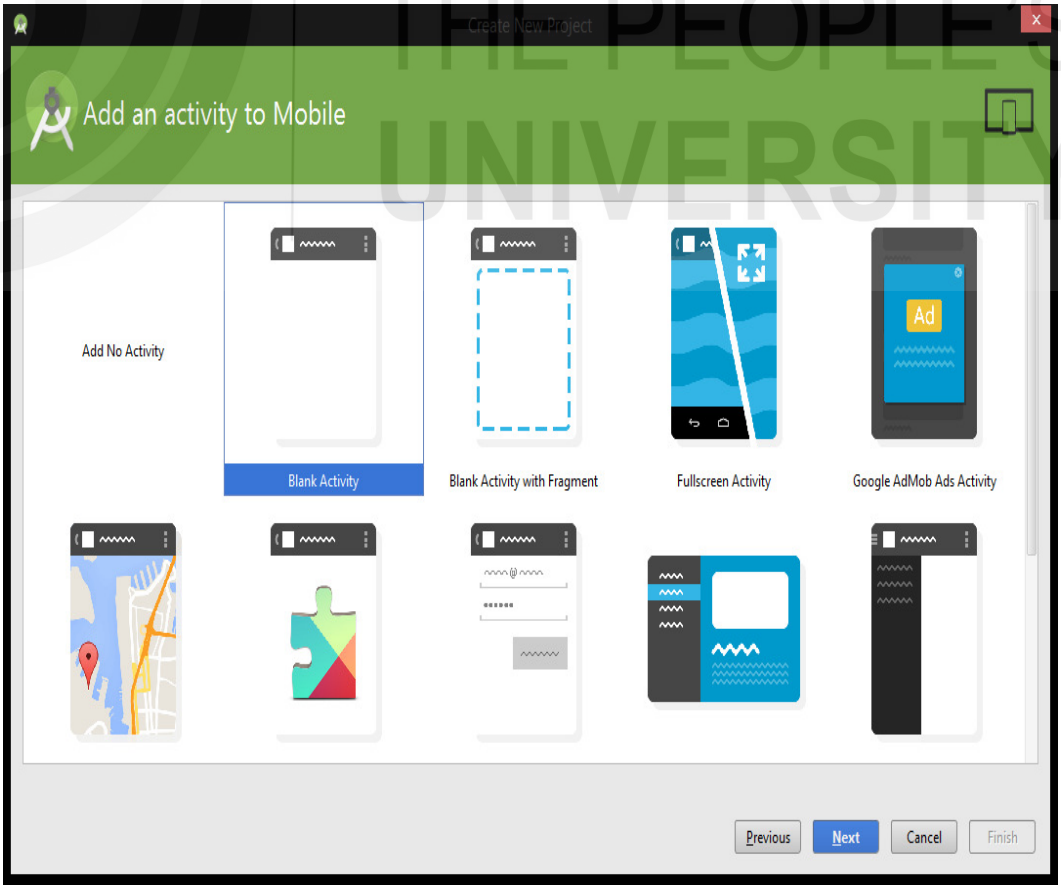

Figure 12.5 : Adding activities in selected device

iv) Build the project and select the hardware device in Android Virtual Device (AVD) manager (Figure 12.6).

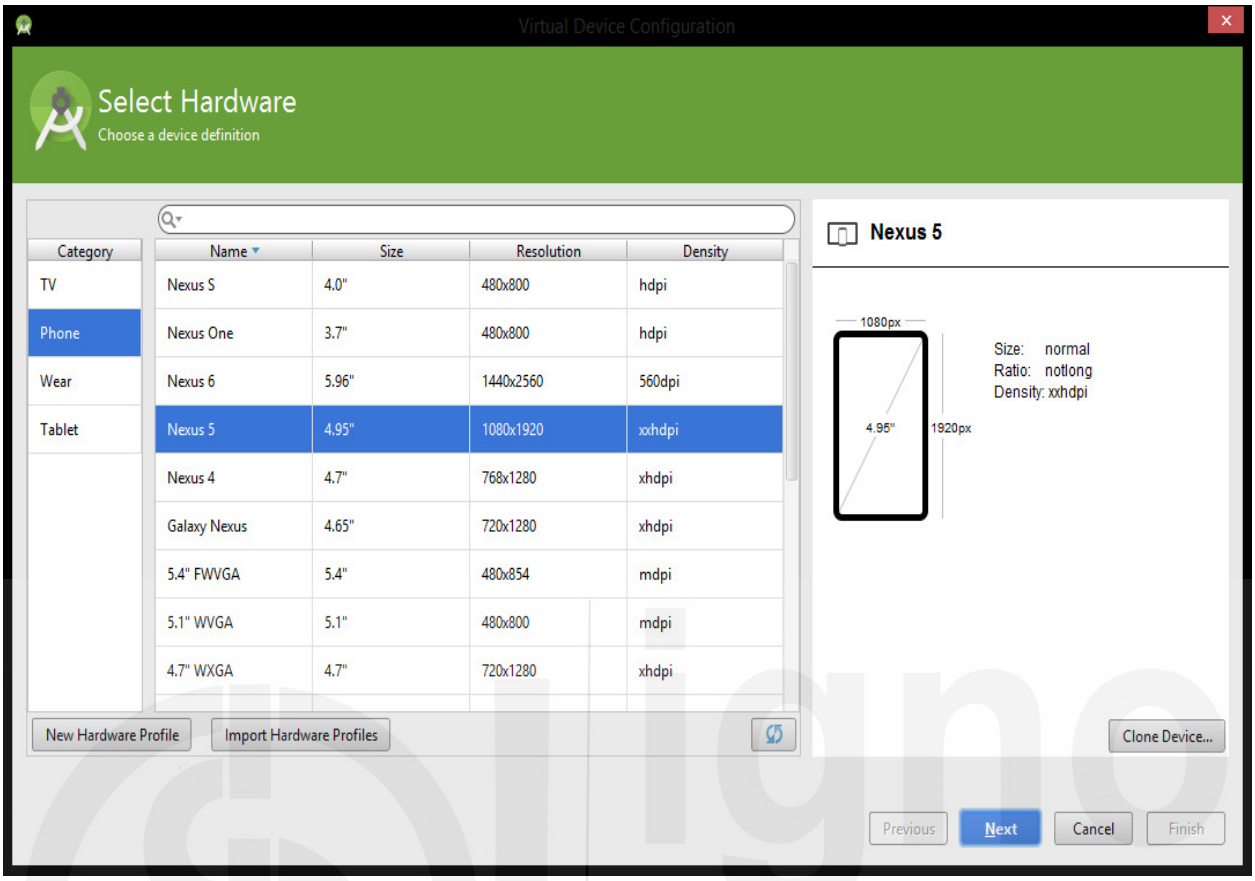

Figure 12.6 : Hardware selection

v) Edit the strings.xml as needed (Figure 12.7).

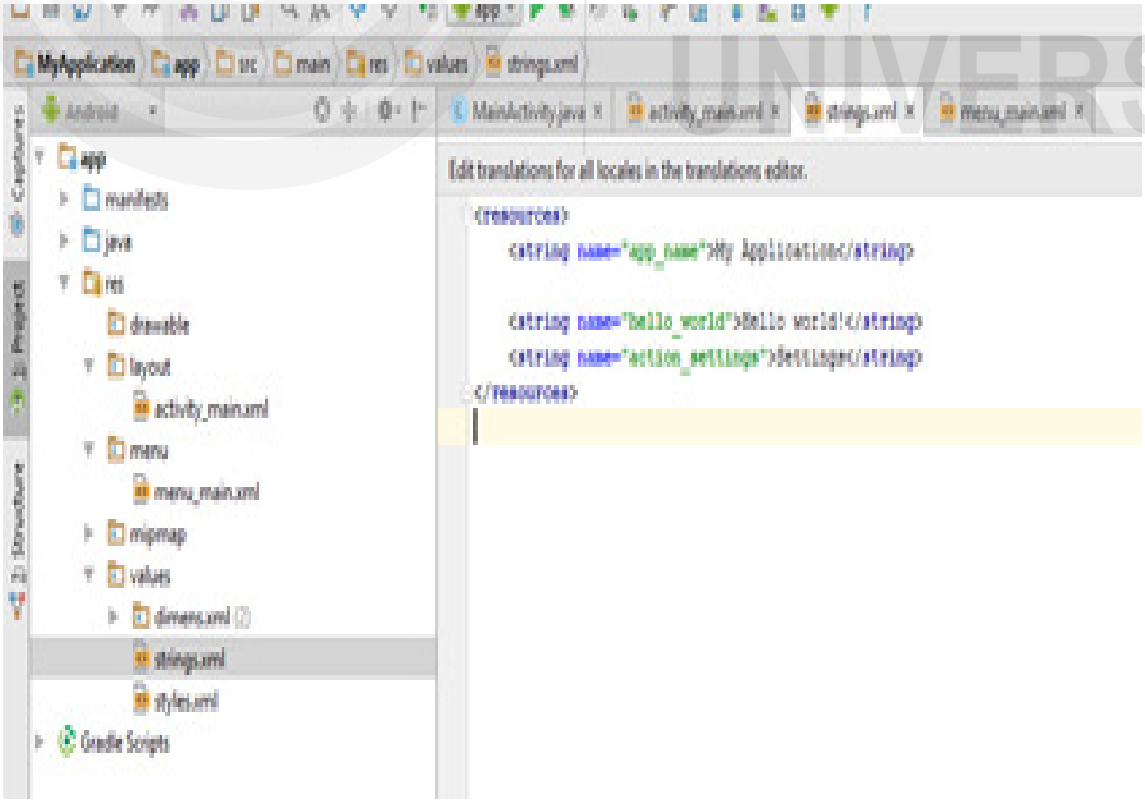

Figure 12.7 : Editor in Android studio

vi) Run the application on the emulator (Figure 12.8).

Figure 12.8 : Output in Emulator

12.2.2 Eclipse IDE

Eclipse is the integrated development environment (IDE) for developing Java based applications. We can build, debug, deploy and test the enterprise Java and core Java applications. Eclipse studio can also be used to develop the Android app by using Android SDK, Android development tools plug-in and Android virtual device.

The following is the process of setting up of Eclipse IDE:

a) Download the Eclipse IDE from http://www.eclipse.org/download

b) Select Eclipse IDE for Java EE Developers and select the appropriate OS.

c) Install the appropriate JRE and JDK needed and configure it in Eclipse IDE.

d) For installing the updates and plug-ins, we can use the menu Help → Software Updates. Design Studio can be installed from the Update option.

Migration from Eclipse to Android Studio

In this section, we will discuss migration from Eclipse to Android Studio. Table 12.1 provides the mapping of components in both Eclipse IDE and Android Studio:

Table 12.1 : Mapping of components for Eclipse IDE and Android studio

Eclipse IDE	Android Studio
Project	Module
Workspace	Project
Library project	Module

A sample Eclipse project is shown in Eclipse Project Explorer in Figure 12.9.

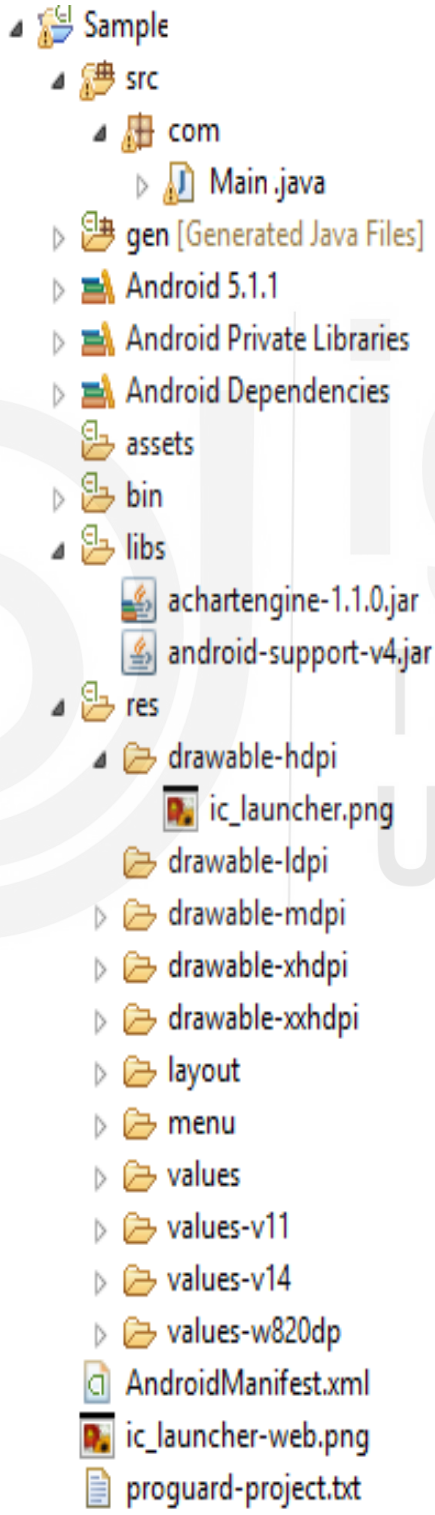

Figure 12.9: Eclipse Project Explorer

The corresponding Android Studio project is shown in figure 12.10.

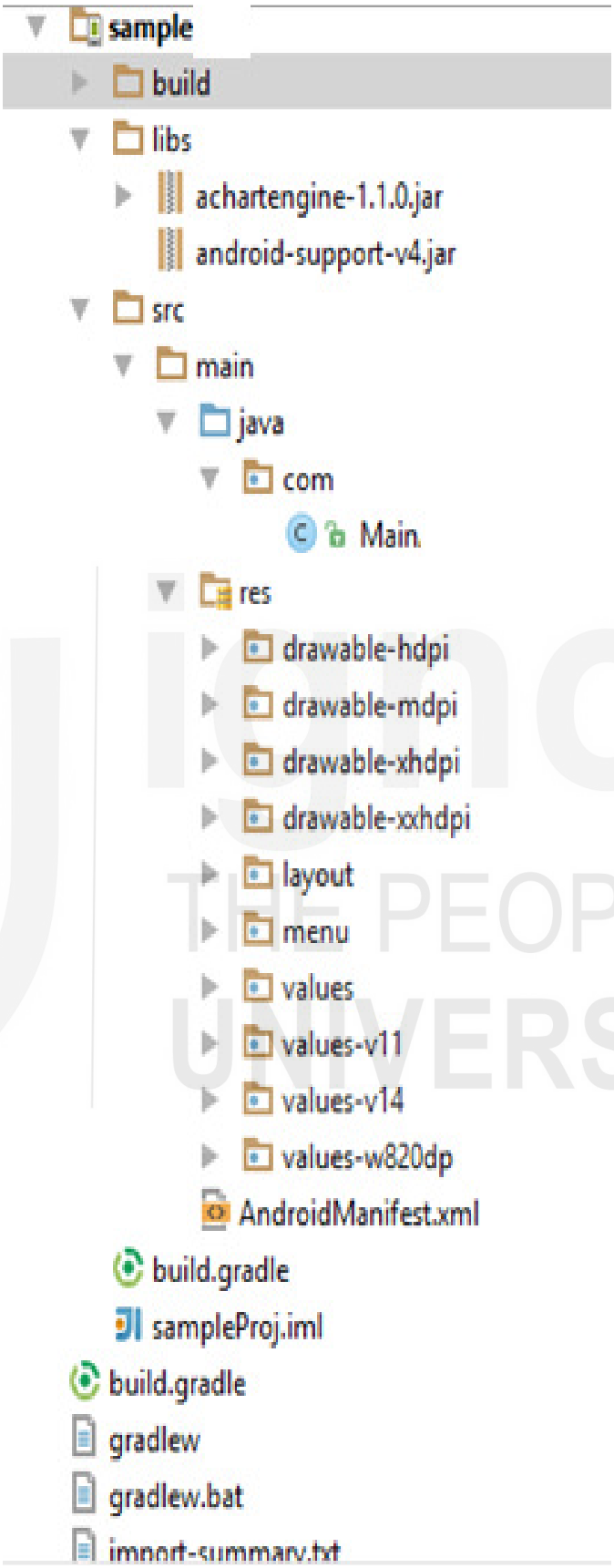

Figure 12.10: Android Studio Project

The following are the steps to migrate from Eclipse project to Android Studio project:

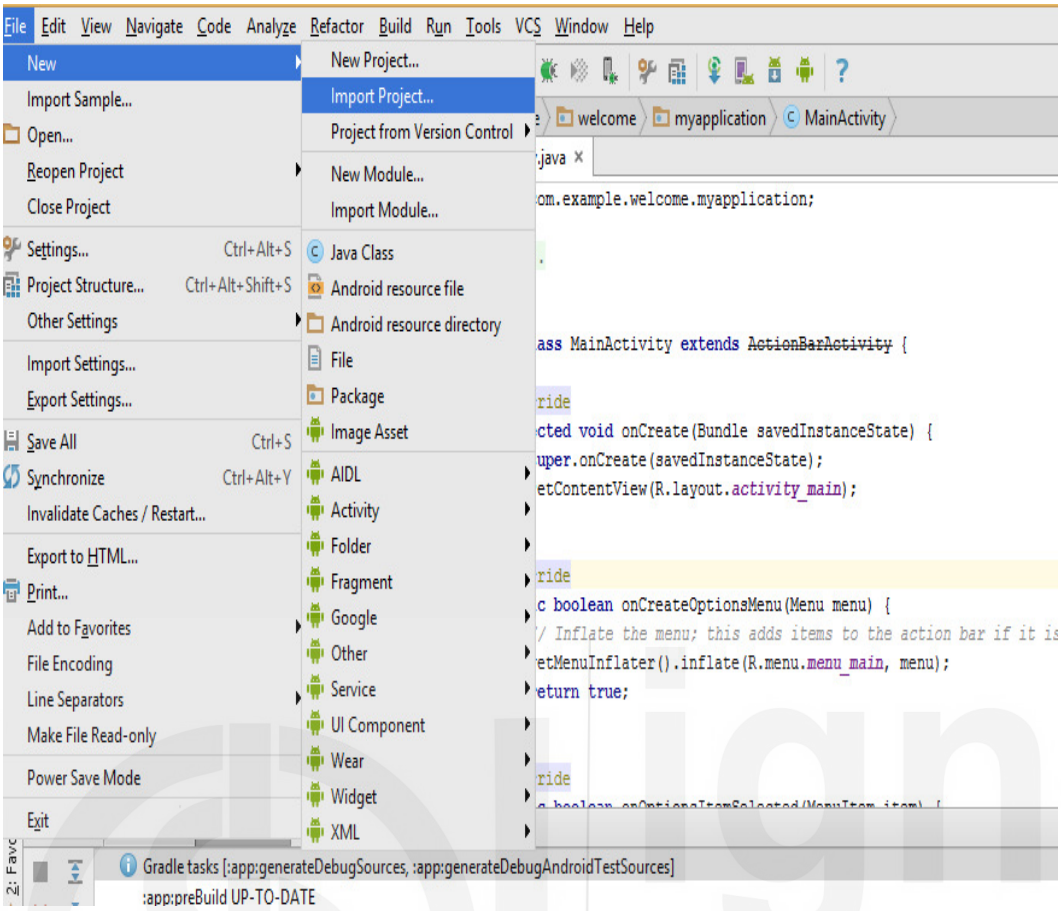

Figure 12.11: Opening a new Project in Android Studio

1) In Android Studio go to File → New → Import (Figure 12.11)
2) Provide Eclipse project path
3) Give target file path
4) Check "Replace jars" and "Replace library sources" (Figure 12.12)

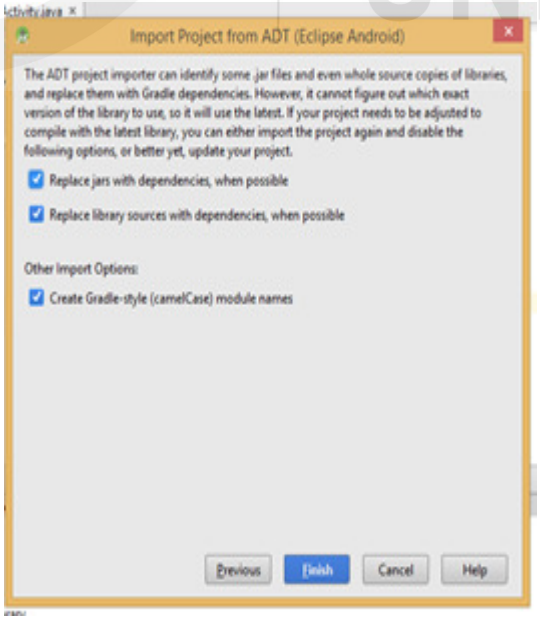

Figure 12.12 : Replace jars and Replace Library Sources

5) Figure 12.13 summarizes import from Eclipse to Android Studio :

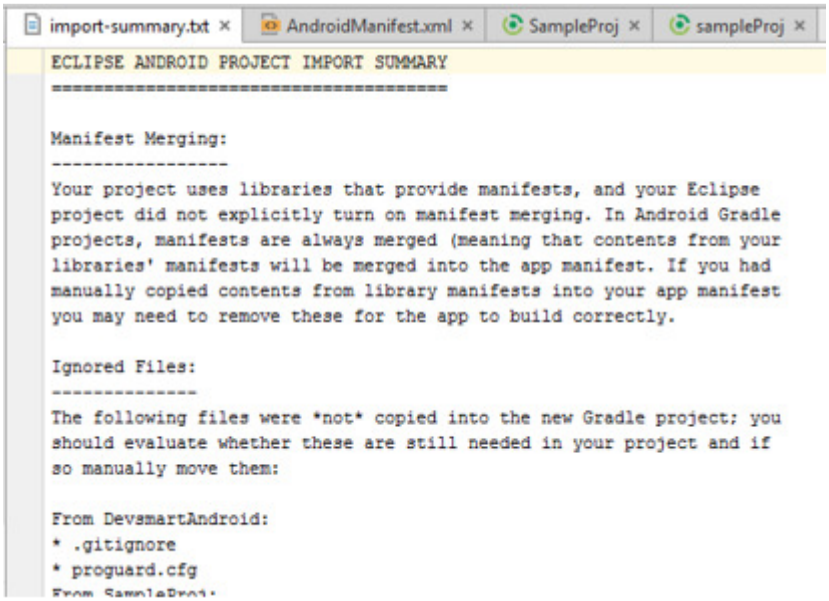

Figure 12.13: Import summary from Eclipse to Android Studio

6) Once the project is setup , we can add JARs and library project using context menus.

7) Once we run the project, the emulator output is shown in Figure 12.14

Figure 12.14 : Output in Emulator

12.3 DEVELOPMENT TOOLS FOR iOS

In this section, we will look at various tools and IDEs needed for development in Apple iOS. Xcode IDE and Swift can be used to develop apps for Apple devices such as iPhone, iPad, Apple Watch, Apple TV, Mac etc.

12.3.1 Xcode

Xcode IDE consists of a set of tools to develop apps for Apple's iOS mobile platform (Figure 12.15). The key features of Xcode are as follows:

- Xcode provides a single interface for all workflow and it is fully integrated with interface builder.
- Xcode provides large number of utility tools for memory leak profiling, allocation management, automation, code signing, code documentation, interface builder etc.
- User interface can be designed by reusing UI objects in the dock.
- It consists of version editor to connect to source control systems such as Git, SVN etc.
- It provides Xcode debugger for code debugging
- Can be used for monitoring and control of the program execution using breakpoints, watch points
- Fix-it can be used for error highlighting and for fixing code errors in real time.
- Xcode tool kit offers UI automation and accessibility testing.
- Xcode also acts as a source management system providing features such as snapshot creation.
- Xcode provides option to save code as a template and provides re-usable code snippet library.

Figure 12.15 : Xcode IDE

The project navigator view of Xcode is shown in figure 12.16.

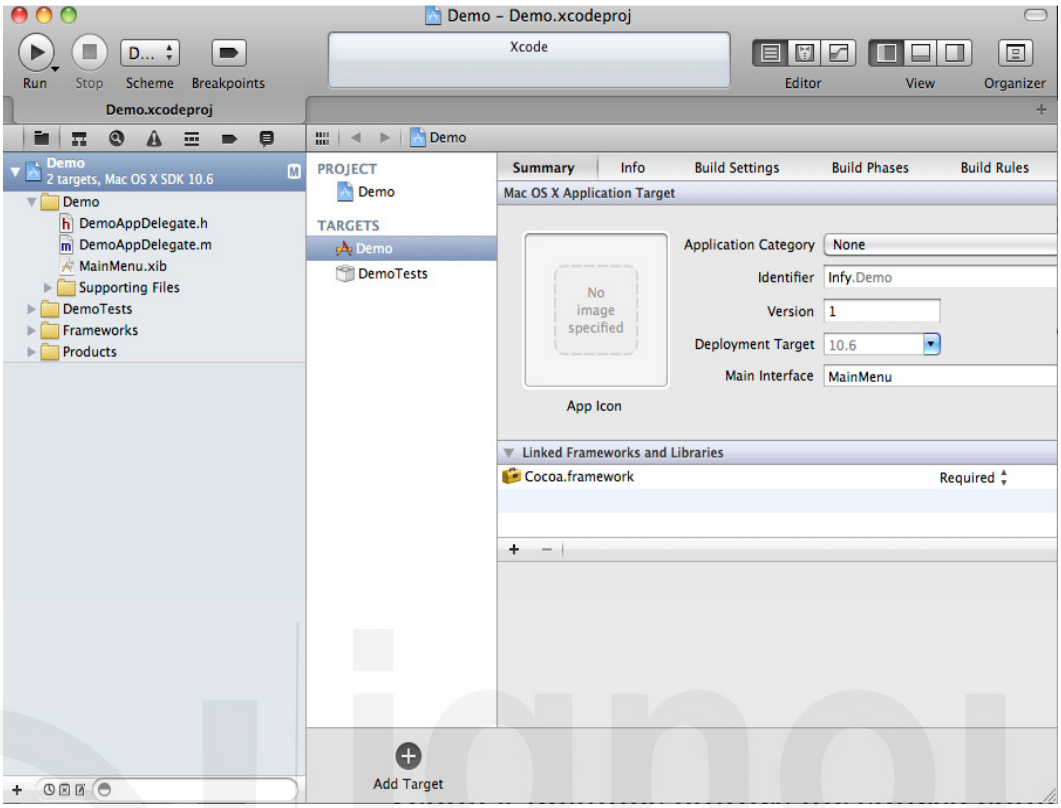

Figure 12.16 : Project navigator of Xcode

Interface builder which can be used for UI designing is shown in figure 12.17. Its possible to make direct connections to source code from interface builder.

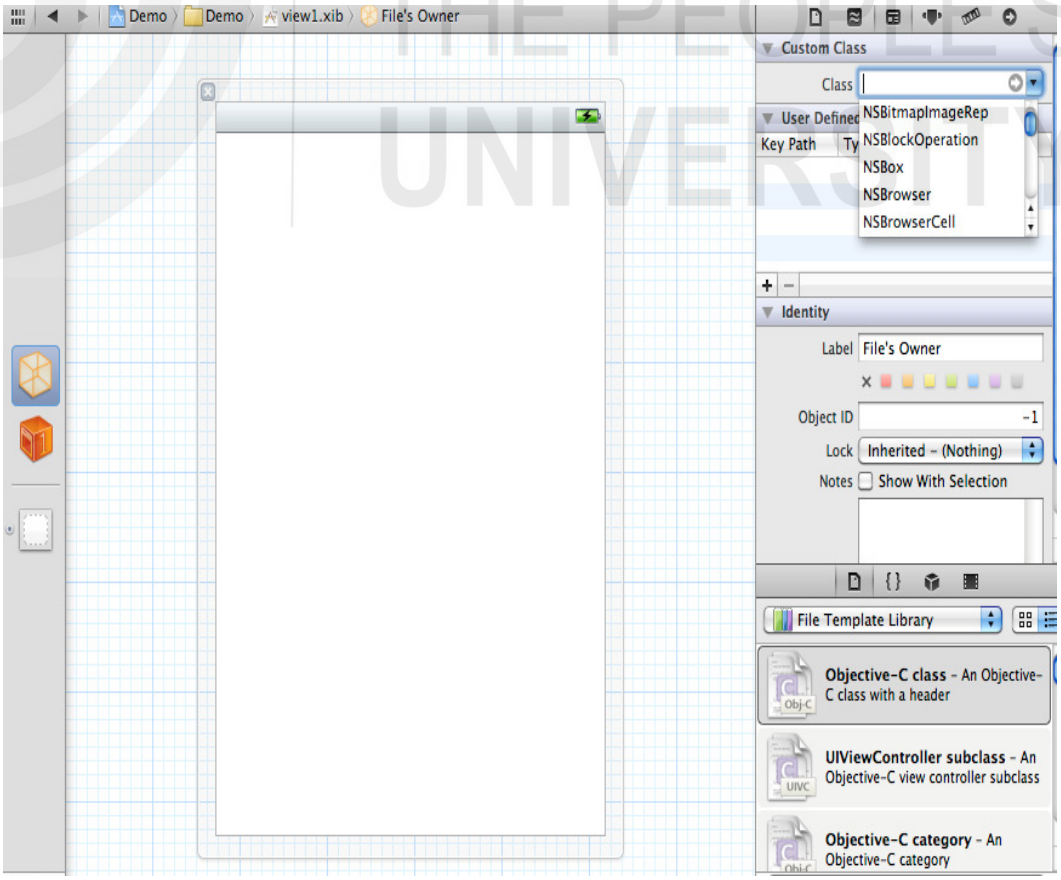

Figure 12.17: Interface Builder

12.3.2 Swift

Swift is the programming language that is supported from iOS 8 onwards and it succeeds objective- C used for earlier Apple development. Swift has a cleaner syntax and is designed to work alongside Objective – C. To use Swift, we need at least Xcode 6 and we we can use Apple's *Playgrounds* feature to try out Swift.

Swift uses Model-view-controller (MVC) programming architecture and uses Apple's Cocoa and CocoaTouch frameworks. Low level virtual machine is used as Swift compiler.

Some of the key advantages of Swift are as follows:

- Enhanced readability: It replaces "@" and "NS" in Objective-C and uses a "." Notation.
- Enhanced maintainability: Swift can be easily maintained as a single file.
- Enhanced safety due to better error handling.
- Enhanced memory management due to automatic reference counting.
- Better performance
- Namespace feature to avoid name collision
- Interoperability with Objective C

The following is a high level description of programming elements of Swift:

Variables

Examples of Swift variables are given below:

```
var myVar:Int8 = 1
var testString = "Hello World"
let someConstant = 12.2
```

We could use "+" operator for string concatenation as well

Arrays

We can define arrays as follows:

```
var noArray = [12,23,2,4,5]
var strArray = ["Ab","Cd","Ef","Gh"]
```

Dictionaries

Swift allows us to save key value pairs in dictionaries. For example,

```
var testDictionary : Dictionary<String,String> = ["1" : "One", "2" : "Two", "3" : "Three"]
```

We can use key as an index to retrieve the value from dictionary

```
var value = testDictionary["1"]
```

In this case, "value" would be "One".

Functions

"Func" can be used to define functions

```
func helloWorld () {
```

```
println("Hello World !")
}
```

Classes

Class keyword can be used to define classes

```
class PersonDetails {

var firstName : String
var lastName : String
var age : Int = 5

init() {
firstName = "John"
lastName = "Smith"
}

func printPersonDetails() {

println("FirstName: \(firstName)")
println("LastName: \(lastName)")
println("Age: \(age)")

}
}
```

☞ Check Your Progress 1

1) _____ can be used along with Android Studio for performance monitoring
2) Android Studio uses _____ build system
3) _____ file contains the key text values in Android app
4) A project in Eclipse IDE would become _____ in Android studio
5) _____ can be used for error highlighting and for fixing code errors in Xcode
6) _____ can be used for UI designing in Xcode
7) ___ qualifier is used in Swift to define constants
8) _____ feature is used in Swift to define key value pairs

12.4 DEVELOPMENT TOOLS FOR WINDOWS BASED MOBILES

Apps for Windows based mobiles apps can be developed using two methods: C++/C#/VB with XAML or by using JavaScript, HTML 5 and CSS3.

12.4.1 C#

C# (pronounced as C Sharp) is a .NET based Object Oriented programming language used for Microsoft mobile app development. Microsoft Visual Studio is the main tool used for development of C# applications.

A simple Hello World program in C# is given below.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace HelloWorld
{
   class Program
   {
      static void Main(string[] args)
      {

      Console.WriteLine("Hello World !");

      }
   }
}
```

Some of the key features of C# are as follows:

- Namespaces: They create easily organizable and reusable compilation units in C#.
- Classes are defined by keyword “class”.
- Variables can be declared using <datatype identifier> syntax. For example *public string pname* declares “pname” as a string type variable.
- Access modifiers include public, private, protected, internal and protected internal.
- Methods form the building block of the C# code.
- Exception handling can be done using try-catch blocks.

12.4.2 XAML

XAML is mainly used for page designs. It provides the flexibility to assign properties for an object. XAML with C# is used when we need dynamic behavior for controls such as focus, hover and when data binding and context setting is required for the app. XAML can also be used to control device services such as GPS, audio, GPRS, Gravity sensor etc.

The developer can set the look and feel properties for the object in XAML. Main properties that can be represented in XAML are as follows:

- **x:Key** uniquely identifies elements that are created and referenced as resources which exist within a **ResourceDictionary**
- Style.TargetType : Gets or sets the type for which this style is intended.
- Setter: This is a class that applies a property value

A resource that can be referenced in XAML can either be static or dynamic. Differences between the static and dynamic resources are given in table 12.2.

Table 12.2 : Static vs Dynamic Resources

Static Resource	Dynamic Resource
Load time resolution of key wherein the reference is resolved and replaced at load time	Compile time key resolution wherein the reference is resolved at compile time but replaced at run time.
Forward reference not supported	Forward reference supported
Can be used when resources are predefined and not dependent on user input	Can be used when the proper is dependent on the user.

To create an app using C# and XAML we need the following components:

- XAML component which defines the page layout, page controls, visual design, controls nesting and such.
- Associated back end page code that takes care of data binding and handles user responses.

Microsoft Visual Studio helps in rapid application development for the Windows apps by providing various features such as IntelliSense, end-to-end packaging and deployment, and debugging support. Figure 12.18 shows a screenshot of "Desipad" Windows app (https://www.microsoft.com/en-us/store/p/desipad/9wzdncrdccgd) designed using Metro UI developed using MS Visual Studio. The app uses Hub and Grid Windows Store template.

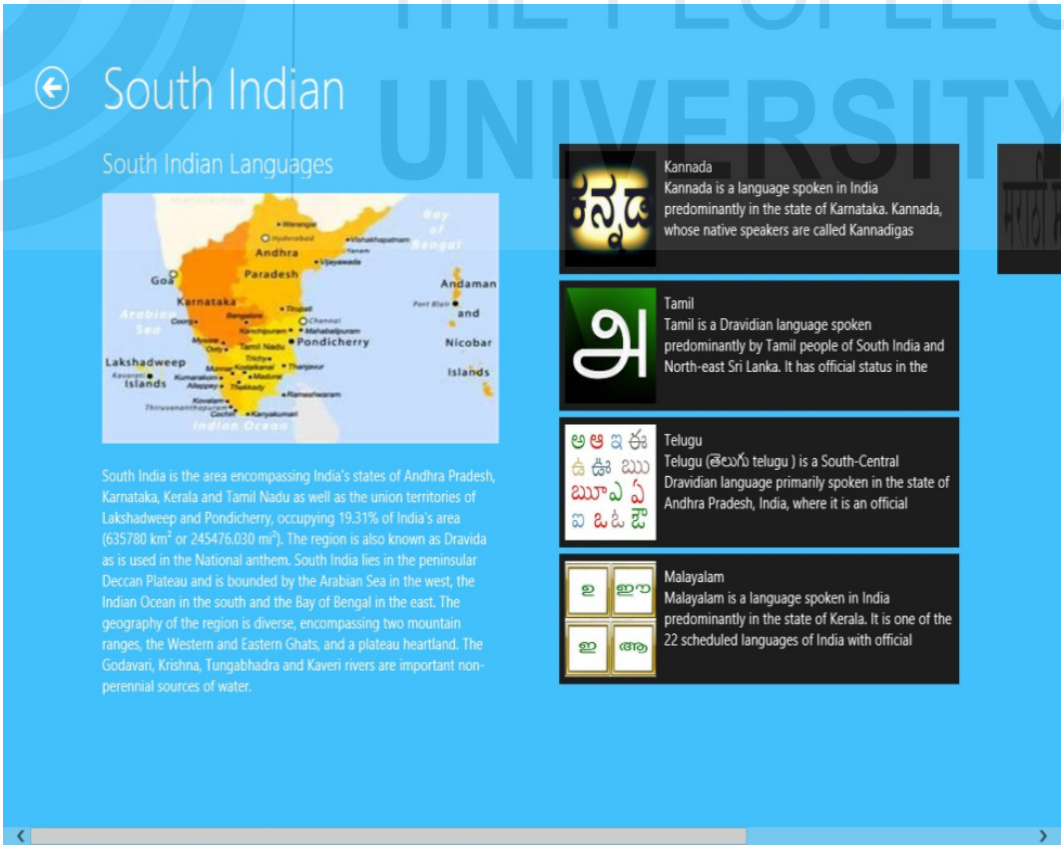

Figure 12.18 : Desipad Windows app

To create a new Windows app in Visual Studio, we need to create a new project and select the Windows store template with preferred application name.

☞ Check Your Progress 2

1) The two options to develop mobile apps in Windows platform are _____ and _____ .

2) ____ represents style name in XAML.

3) _____ resource has load time resolution in XAML .

4) _____ defines the page layout, page controls in Windows app.

12.5 SUMMARY

In this unit, we started discussing various development tools for mobile platforms. We started looking at Android tools such as Android studio, Eclipse and migration from Eclipse to Android studio. For iOS platform, Xcode and Swift programming language were introduced. Finally, for Windows platform, C# and XAML were introduced.

12.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Android monitor
2) Gradle
3) Strings.xml
4) module
5) Fix-it
6) Interface builder
7) Let
8) dictionaries

Check Your Progress 2

1) C++/C#/VB with XAML and JavaScript, HTML 5 and CSS3
2) x:Key
3) static resource
4) XAML component

12.7 FURTHER READINGS

References

- https://www.airpair.com/android/android-studio-vs-eclipse
- http://android-developers.blogspot.in/2014/12/android-studio-10.html
- http://en.wikipedia.org/wiki/Eclipse_(software)

- http://developer.android.com/tools/studio/index.html
- http://tools.android.com/tech-docs/new-build-system/user-guide
- http://developer.apple.com/
- https://developer.apple.com/swift/blog
- https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language
- http://programmingincsharp.com/
- http://msdn.microsoft.com/en-us/

UNIT 13 CROSS PLATFORM DEVELOPMENT TOOLS

Structure

13.0 Introduction

13.1 Objectives

13.2 Various Approaches for Cross Platform Mobile App Development

- 13.2.1 Web Based Cross Platform Mobile App Development
- 13.2.2 Hybrid Approach for Cross Platform Mobile App Development
- 13.2.3 Single Language Approach for Cross Platform Mobile App Development
- 13.2.4 Mobile Middleware Approach for Cross Platform Development

13.3 Xamarin

- 13.3.1 Xamarin.Android
- 13.3.2 Xamarin.iOS

13.4 Phone Gap

- 13.4.1 Developing an Android App in Phone Gap
- 13.4.2 Developing an iOS App in Phone Gap

13.5 Advantages and Disadvantages of Cross Platform Mobile App Development Tools

13.6 Summary

13.7 Solutions/Answers

13.8 Further Readings

13.0 INTRODUCTION

Mobile apps are rapidly becoming the primary channels to deliver superior end user experiences. Organizations are choosing native mobile apps to implement their mobile-first strategy. Mobile apps are the de-facto customer engagement platforms which directly impact the organization's revenue.

There are various mobile platforms such as Apple iOS, Google's Android, Windows phone etc. Additionally there are variety of mobile devices differing in form factors, capabilities, resolution and such. As such the development platform for these mobile systems are also different: Apple iOS uses Swift/Objective-C, Android uses Java and Windows uses C#/XAML. The app development standards and patterns are also different: iOS uses MVC pattern whereas Windows app uses Model-View-Viewmodel (MVVM) pattern. Hence in order to develop apps for all mobile platforms we need understanding of diverse languages, standards and patterns. This also impacts the skill set and resources needed for such development projects.

In order to save time, effort and reuse the code across platforms we use cross platform development tools such as PhoneGap, Titanium and Xamarin. These cross platform tools would help us to reuse the code across mobile platforms. Cross platform tools implement the "Write Once Run Anywhere" concept.

13.1 OBJECTIVES

After going through this unit, you should be able to understand:

- various approaches for cross platform mobile app development and its applicable scenarios;
- key concepts of Xamarin for Android and iOS development;
- key concepts of PhoneGap for Android and iOS development; and
- the advantages and disadvantages of cross-platform mobile app development tools.

13.2 VARIOUS APPROACHES FOR CROSS PLATFORM MOBILE APP DEVELOPMENT

Native mobile apps provide superior user experience and leverages native and device-specific features. They provide the best performance and are more suited for game kind of applications. Native apps are distributed through app stores. However, the main drawback of native app development is that in order to develop apps for various platforms, we need developers with varied skill sets. This would lead to escalation in price, effort and impacts time to market. Total cost of ownership (TCO) and the overall license cost would be high for developing apps to various platforms natively.

In order to address this problem, there are various tools and techniques to develop cross platform mobile apps. In this section, we will briefly discuss key cross platform mobile app development techniques.

13.2.1 Web Based Cross Platform Mobile App Development

The following are salient features of Web based cross platform mobile app development:

- Developers will develop web applications that can run in mobile web browsers. This typically follows Model-View-Controller (MVC) architecture;
- Javascript based widgets and UI components would communicate with back end server components (such as servlets, JSP) through light weight services (such as REST based services);
- Backend server components would interact with enterprise interfaces such as database, content management systems, internal applications etc.;
- Due to emergence of HTML 5 standards, it is possible to have limited access to some of the device features through this approach;
- All modern mobile platforms such as Android, iOS, Blackberry, Windows phone support a full-fledged browser that can be leveraged for this approach;
- The main technologies are HTML 5, CSS, JavaScript with RWD and backend technologies are Java, and .Net;
- The key Javascript frameworks that can be used in this approach are jQueryMobile, Sencha Touch, and Dojo Mobile; and

- TCO is minimal as we need to use standard web technologies.

We can use this approach when we would like to reuse the web channels for mobile apps. We can also use this approach when we quickly want to launch the application across various mobile platforms.

If the application needs native device features and device hardware, we cannot use this approach. Visually rich applications like gaming apps are not suited for this kind of approach.

13.2.2 Hybrid Approach for Cross Platform Mobile App Development

The following are salient features of hybrid approach for cross platform mobile app development:

- Developers will develop mobile apps in standard web technologies such as HTML 5 and JavaScript;
- JavaScript APIs expose some of the native device features such as camera, contacts etc. through a bridge;
- App developers need the knowledge of Javascript; and
- Adobe PhoneGap, Apache Cordova are some of the frameworks that provide this approach.

We can use this approach when we would like to use HTML 5 based web applications on mobile and provide a good native device access. The code can be shared across multiple mobile platforms.

If the application needs high quality rich UI experience with high performance, then, this approach should not be followed.

13.2.3 Single Language Approach for Cross Platform Mobile App Development

The following are salient features of single language approach for cross platform mobile app development:

- App is developed in single language;
- We could get native look and feel user experience;
- High performance;
- APIs will be provided to access the native device features; and
- Some tools and frameworks that can be used for this approach are Xamarin, Appcelerator Titanium etc.

We can use this approach when we would like to use reuse the code across various platforms and want native UI and performance.

If the application needs high quality rich UI experience with high performance this approach is not suited.

13.2.4 Mobile Middleware Approach for Cross Platform Development

The following are salient features for Mobile middleware approach for cross platform development:

- A dedicated Mobile Enterprise Application platform (MEAP) acts as mobile middleware;
- The middleware provides adaptors to interact with various mobile platforms;
- The middleware provides various features such as integration, authentication, security, governance, and mobile management features etc.;
- The middleware also interacts with back end systems such as database, ERP, web applications, portals etc. and
- Systems such as IBM Worklight is used for this approach.

We can use this approach when we want to support various types of platforms and have a centralized mobile management platform for all mobility needs.

If the enterprise wants to have reduced total cost of ownership (TCO) or use open source technologies, then, this is not applicable.

13.3 XAMARIN

Xamarin is a cross-platform with an inbuilt IDE and SDK based on .net platform. Xamarin can be used to develop apps for Android, iOS and Windows. Xamarin SDK uses Mono Touch for iOS and Mono Droid for Android for building the mobile apps for respective platforms. Xamarin. Android is the Android variant that is used for accessing native UI features, device features etc.; Similarly Xamarin.iOS is the iOS variant.

Xamarin studio is the IDE that can be used to develop and deploy mobile apps and is available for Windows and Mac platforms. Xamarin can be used as mobile middleware to provide native user interface. Xamarin is also compatible with MS Visual studio. Code debugging can be performed using MS Visual Studio integration of Xamarin and XCode is needed for debugging for iOS.

Using Xamarin we can write the business logic code in C# and access platform specific APIs and user interfaces. The code will be compiled using Xamarin compiler that generates just-in-time compile binaries for Android and ahead-of-time native ARM binaries for iOS. Using this method, it is possible to share and reuse the business logic across various mobile platforms.

The following are some of the features of Xamarin:

- Presentation layer consisting of screens, controls and widgets.
- Application Layer that contains platform specific implementation details.
- Business Layer consisting of business logic.
- Data Layer for handling persistence and data related transactions.
- Service Access layer that provide data access services.

The main advantages of Xamarin are as follows:

- It provides a native look and feel.
- Easily integrates with MS Visual Studio. Enables a C# developer to develop the apps for iOS and Android.
- Good performance.

Xamarin.Android and Xamarin.iOS provide framework for invoking platform specific features using C# calls. Using these frameworks, developers can develop full-fledged mobile apps using .net framework and C#.

13.3.1 Xamarin.Android

The following are high level steps to setup Xamarin.Android:

1) Android SDK (https://developer.android.com/studio/index.html) and NDK (https://developer.android.com/ndk/downloads/index.html) are used with MS Visual Studio for developing Android apps.

2) Install Android APIs from SDK manager (Figure 13.1).

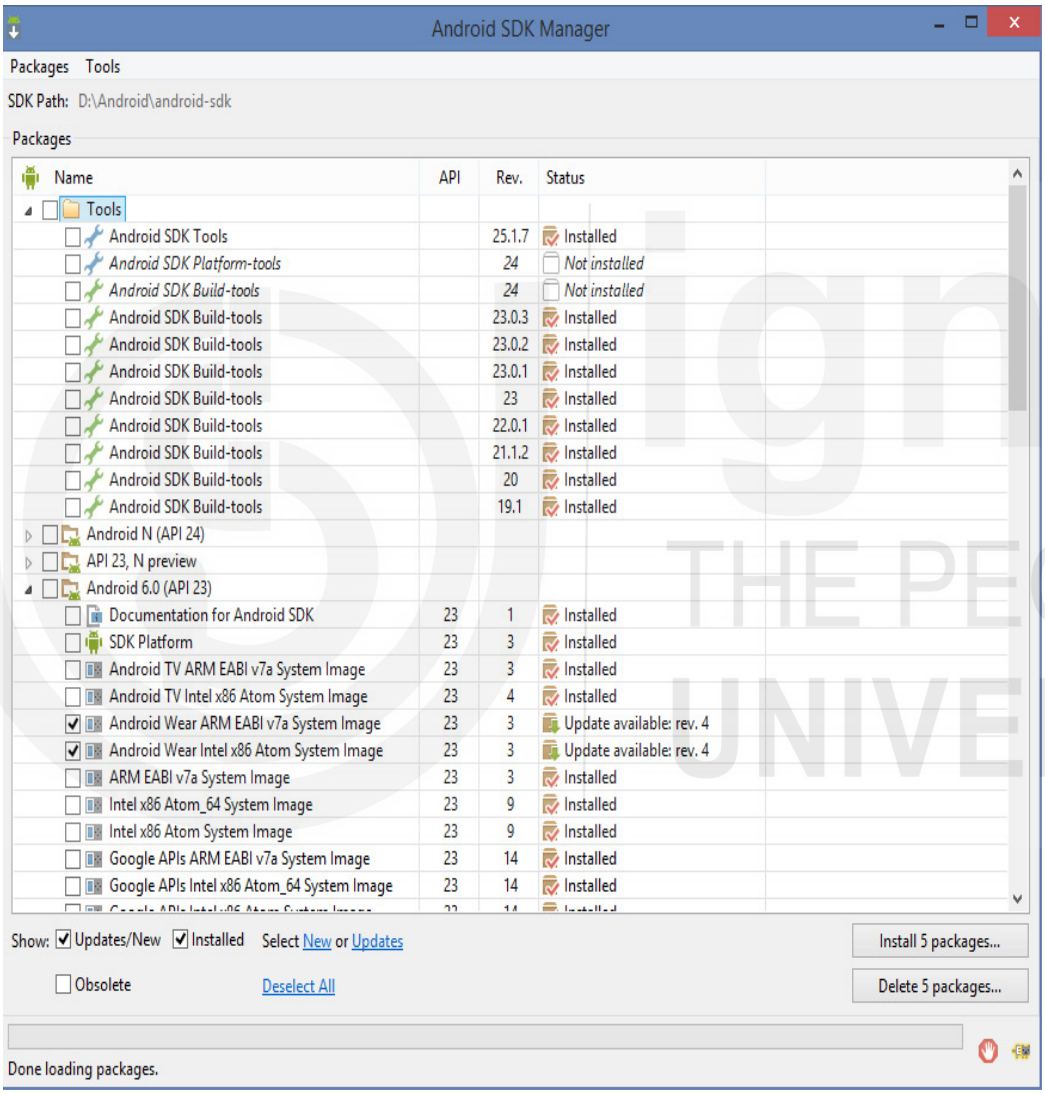

Figure 13.1 : Installation of Android API from SDK manager

13.3.2 Xamarin.iOS

C# and Objective-C can be used for development in this framework. It connects native iOS and C#. Xamarin can be used to develop iOS apps. High level steps to develop a sample app listing the values in Apple iPhone app using Xamarin are given below:

1) Open Xamarin for iOS (Figure 13.2)

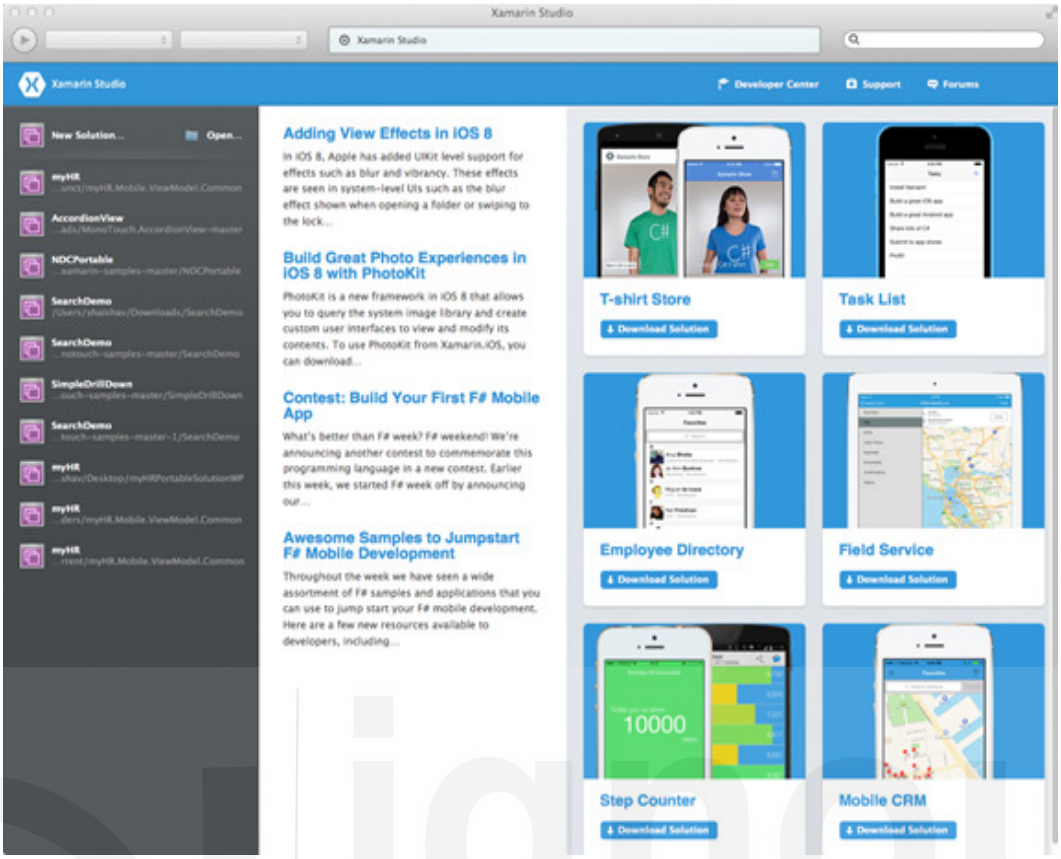

Figure 13.2 : Xamarin for iOS

2) Select the app type. For example, in the following screenshot, we have selected "Single View Application" (Figure 13.3).

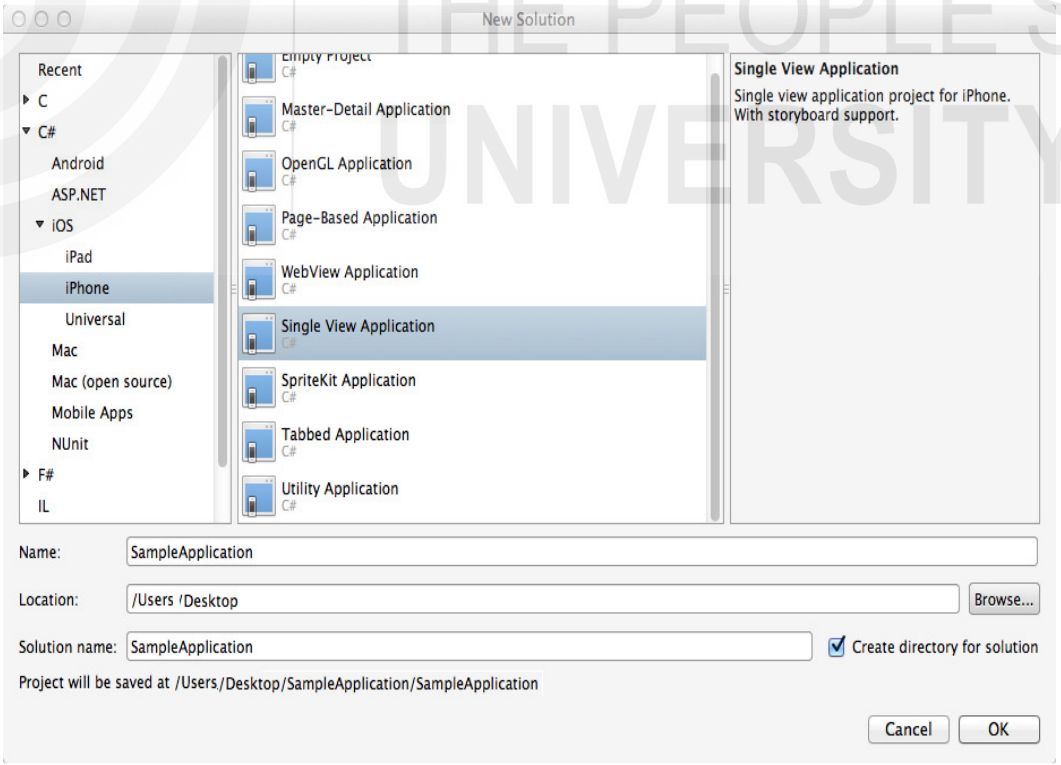

Figure 13.3: App selection in Xamarin

3) Use ViewController.cs to open the Xcode based application (Figure 13.4)

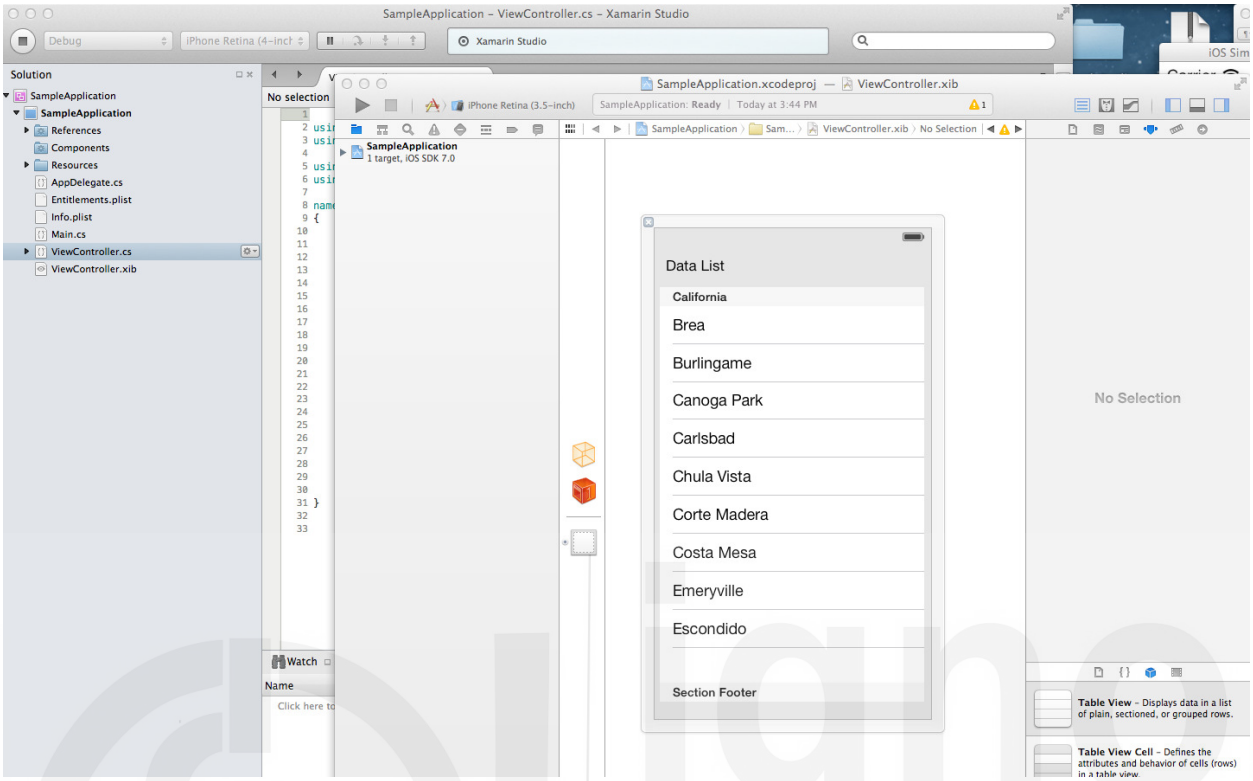

Figure 13.4: View Controller in Xamarin

4) Add needed elements. In this sample application we have added UILabel and UITableView. In ViewController.cs, add data list to UITableView (Figure 13.5 and Figure 13.6).

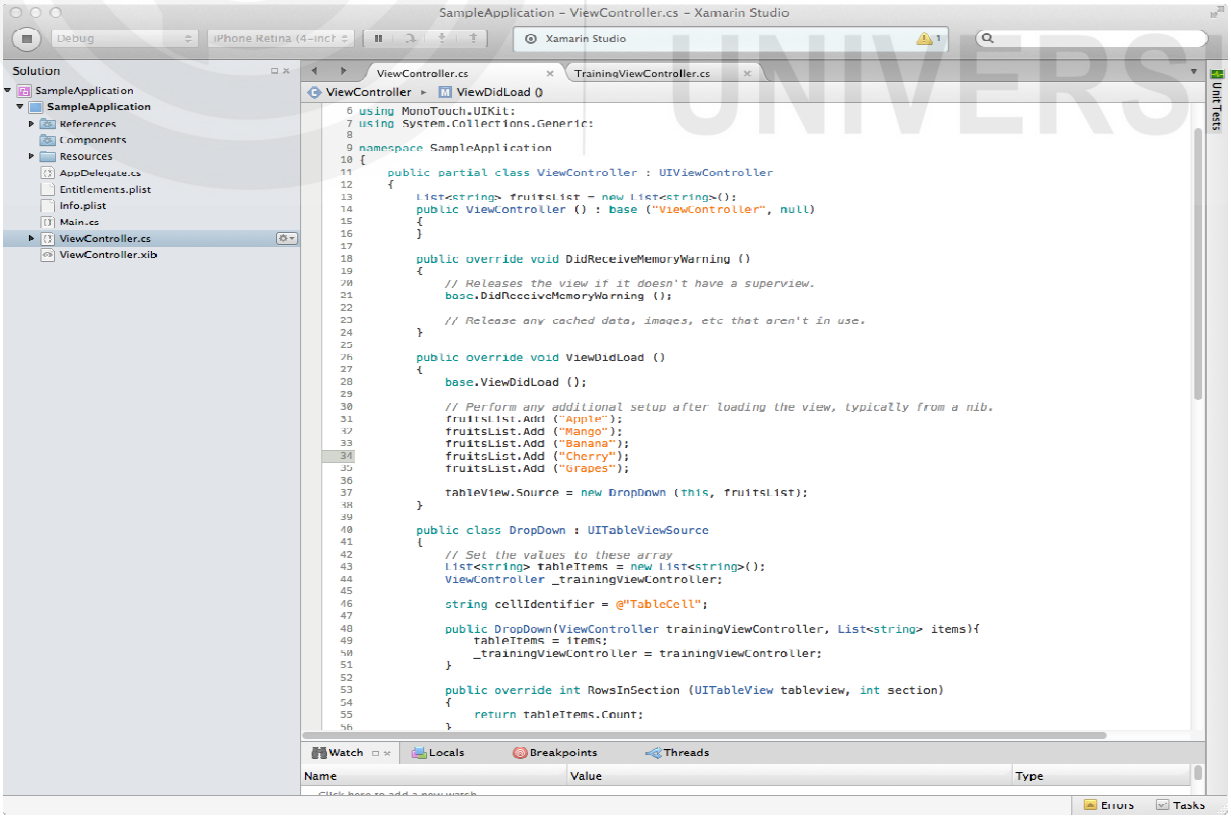

Figure 13.5: App Code in Xamarin

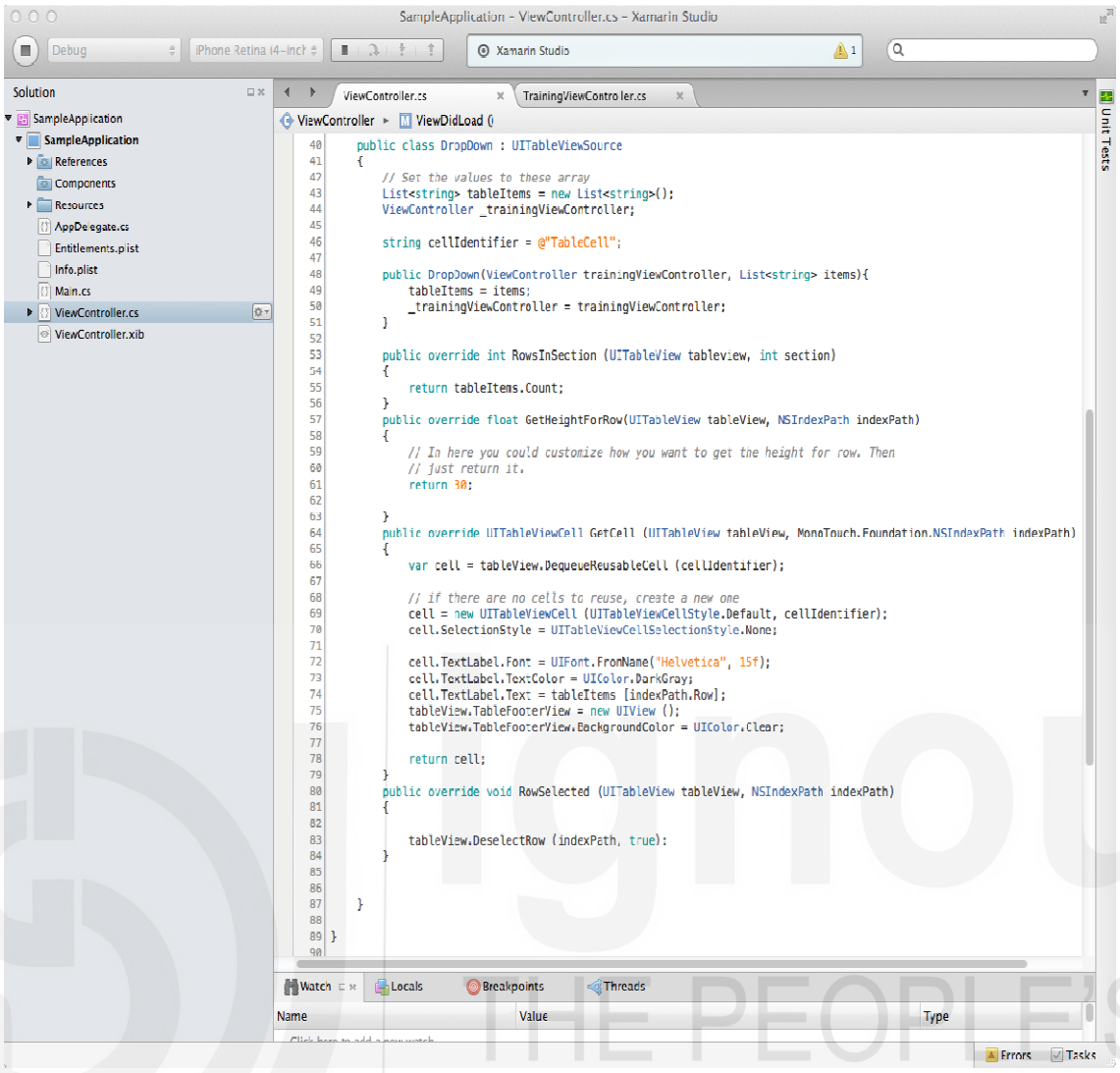

Figure 13.6: App code in Xamarin

5) When we run the application in iOS simulator, the output is displayed as shown in Figure 13.7

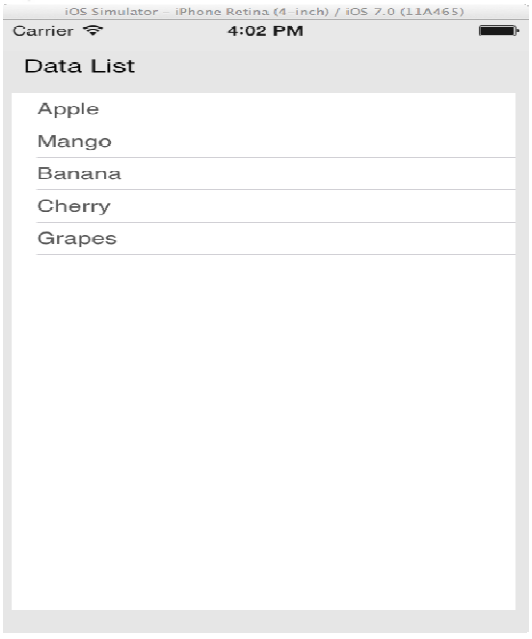

Figure 13.7 : iOS Simulator

13.4 PHONEGAP

HTML5, JavaScript and CSS are most popular technologies and standards to develop cross-platform mobile apps. PhoneGap is one of the most popular cross-platform development tools that uses HTML 5 compatible browser in the native app and provides JavaScript functions to access native platform capabilities. PhoneGap provides HTML 5 browser and a hybrid framework exposing APIs to access native mobile features. PhoneGap builds and runs an application for a device. PhoneGap is the original distributor of Apache Cordova. As HTML 5, JavaScript and CSS are standard technologies, it would be possible to easily develop and reuse the code and deploy it to various platforms using this technology. The trade-off is the performance overhead that occurs in this process.

PhoneGap provides APIs to access native mobile features such as camera, GPS, accelerometer, contacts, sensors etc. The basic architecture of PhoneGap is shown in Figure 13.8.

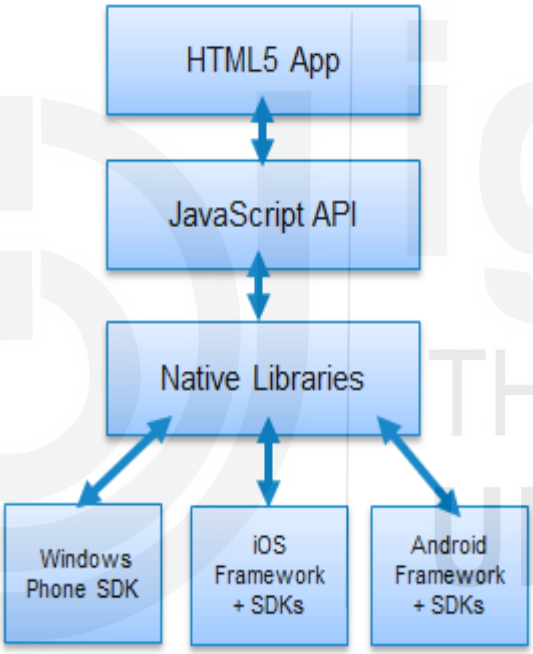

Figure 13.8: Architecture of PhoneGap

13.4.1 Developing an Android App in PhoneGap

The following are steps involved in developing a simple Hello World Android App in PhoneGap:

1) Download the PhoneGap libraries

2) Create a new Android project with empty activity (Figure 13.9). Then, its followed by a new Android application (Figure 13.10).

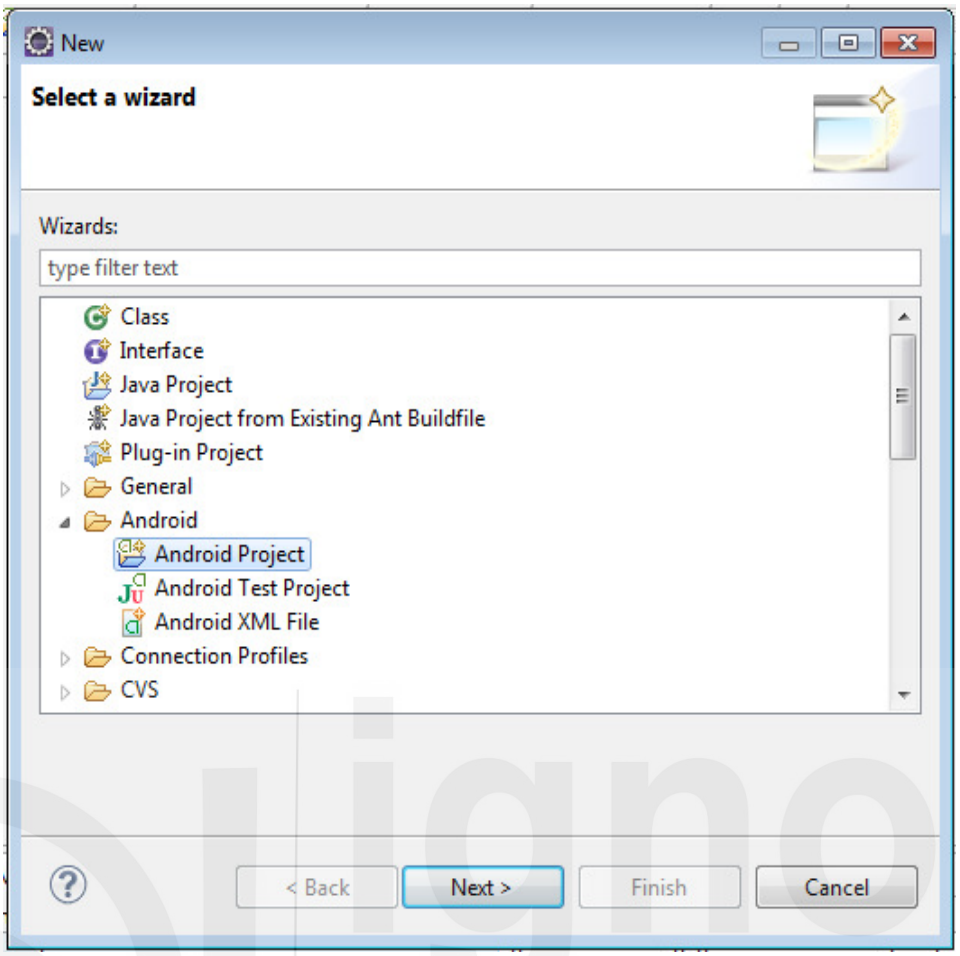

Figure 13.9 : Creation of a new Android project

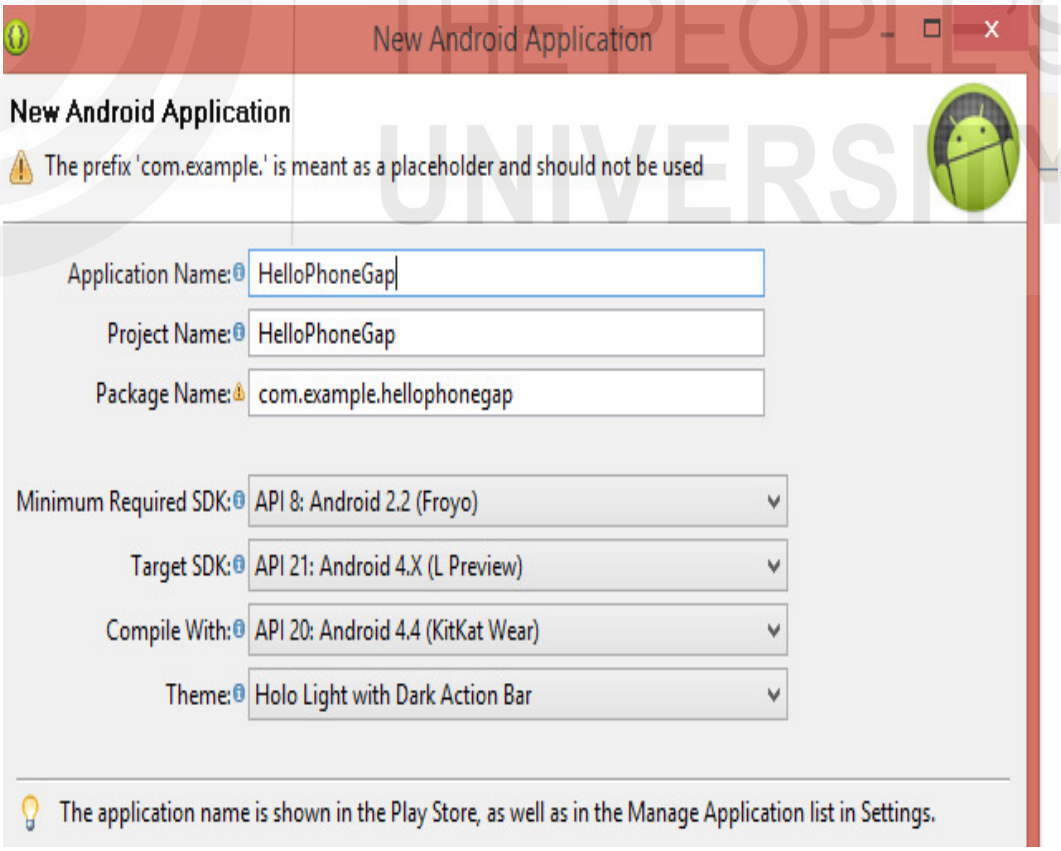

Figure 13.10 : New Android Application

3) Create “www” folder within assets and we need to create all HTML, JS, and CSS in this folder.

4) Move the “cordova.jar” from PhoneGap libraries into the project’s lib folder and “cordova.js” into the newly created “www” folder. Copy xml folder to “res” folder.

5) Create and edit the index.html file in “www” folder. Add the reference of the Cordova.js in the index html

<script type="text/javascript" charset="utf-8" src="cordova-2.6.0.js"></script>

6) Edit the activity file and load the html file as shown in Figure 13.11

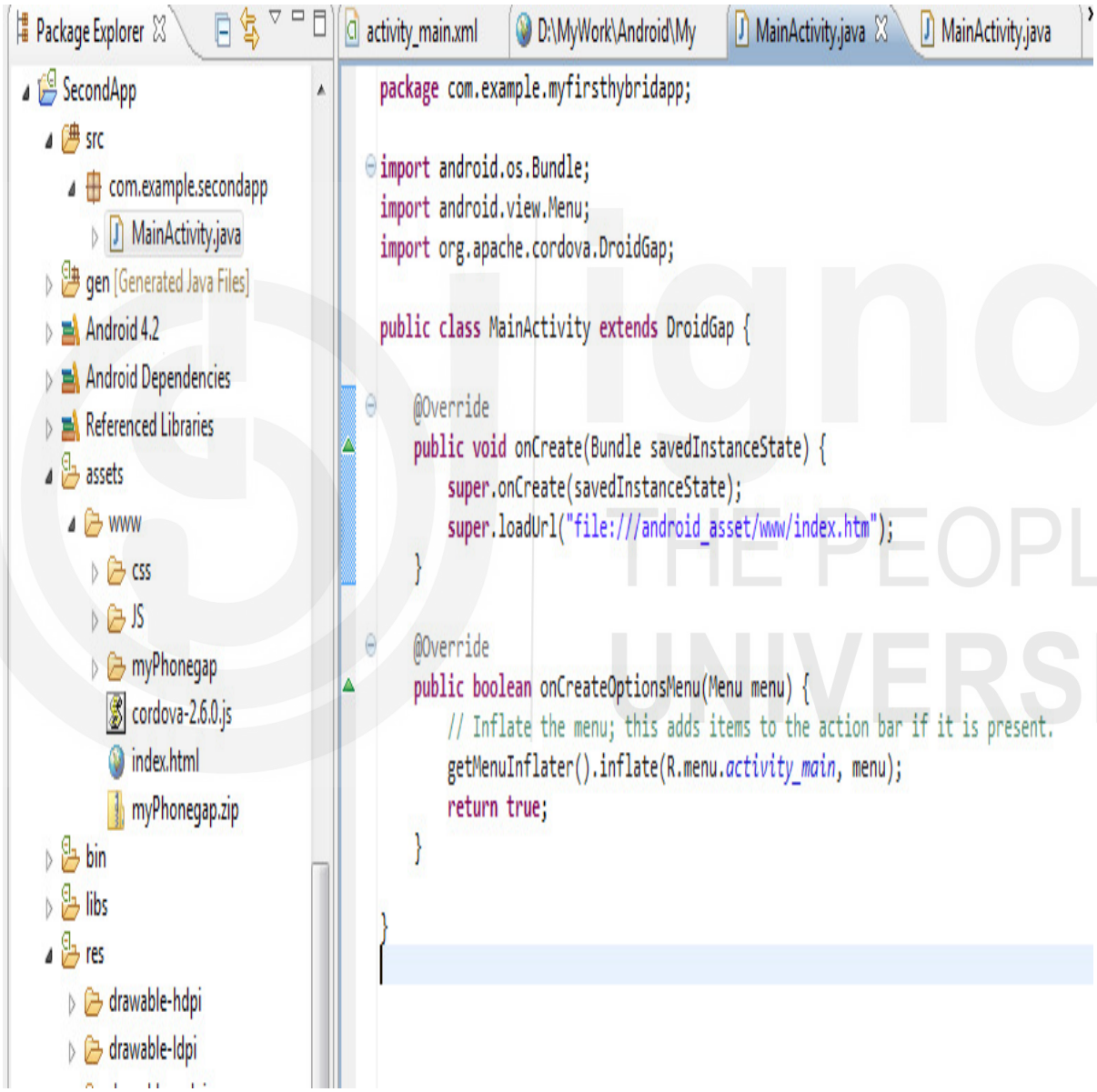

Figure 13.11 : Activity file for Android app

7) Add needed metadata in “AndroidManifest.xml” and provide suitable permissions (Figure 13.12).

YourAppNameActivity.java | *AndroidManifest.xml

```
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".YourAppNameActivity"
                  android:label="@string/app_name"
                   android:configChanges="orientation|keyboardHidden|keyboard|screenSize|locale">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Figure 13.13 : Android app output in emulator

8) On running the application in emulator, output is shown in Figure 13.13

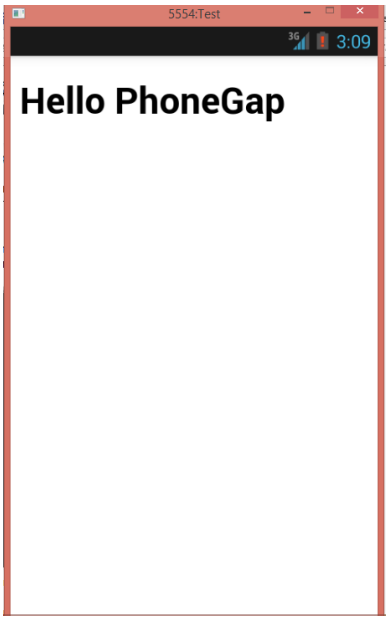

Figure 13.13 : Android app output in emulator

13.4.2 Developing an iOS App in PhoneGap

The following are steps for developing an iOS app in PhoneGap:

1) Download the PhoneGap library from official website (http://phonegap.com/)

2) Create an iOS project in lib/iOS folder of the extracted content (Figure 13.14).

3) Use the shell command to create a simple Cordova iOS project.

4) Open the project using the <projectname>.xcodeproj file.

5) After running the app, the basic application is rendered as shown in Figure 13.15 in simulator

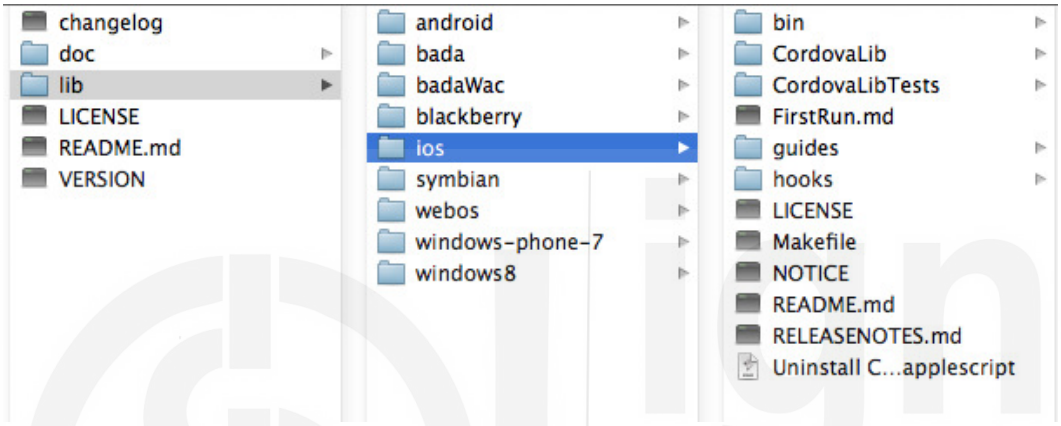

Figure 13.14 : iOS project in lib/iOS folder

Figure 13.15 : Output of iOS app in simulator

☞ **Check Your Progress 1**

1) Xamarin uses __ platform

2) Xamarin SDK uses _____ for iOS and _____ for Android

3) ____ contains platform specific implementation details in a typical app

4) Android APIs can be installed from the _____ in Xamarin

5) _____ opens the XCode application in Xamarin studio for iOS

13.5 ADVANTAGES AND DISADVANTAGES OF CROSS PLATFORM MOBILE APP DEVELOPMENT TOOLS

As discussed earlier, there is a huge diversity in the mobile platform space. There are various mobile devices with different form factors and device capabilities. Various mobile platforms need specialized developers in varied skill sets. Cross platforms play a key role in addressing this challenge.

The following are some of the advantages of cross platform mobile app development tools

Given below are various advantages of cross platform mobile app development tools:

- Cross-platform app development tools provide native look and feel and we can reach wider audience quickly.
- We can use HTML standards (such as in PhoneGap) or a single source code (such as C# in Xamarin) and reuse/share it for various mobile platforms. It would be efficient to manage the mobile apps due to single code base.
- Due to single code base, the cost would be reduced with minimal learning curve.
- Unit testing and bug fixing would be easier due to single code base.
- Updates, maintenance and provisioning of apps to various mobile platforms would be easy.
- Cross platform tools provide uniform look and feel across mobile platforms.
- Time to market for mobile apps will be reduced.

The following are some of the disadvantages of cross platform app development tools:

- Some of the unique/specific device and platform features cannot be exploited within cross platform tools.
- Development of advanced graphics such as 3D graphics is a challenge in these apps.
- Designing a truly exceptional user experience on all mobile platforms would be challenging as each mobile platform has its own unique screen layout, and elements.

- Integrating with native cloud services, notifications, settings, and storage services would be a challenge.
- Mobile apps developed with cross-platform tools will not be able to leverage all security features of the native platform and hence are relatively more vulnerable to security attacks.
- Performance would be challenge sometimes leading to slow rendering issues.

☞ Check Your Progress 2

1) HTML, CSS, Javascript are copied to _____ folder while creating a PhoneGap app.
2) _____ file contains the permissions in Android app developed using PhoneGap.
3) _____ leads to minimized cost, effort and learning curve in apps developed from cross platform tools.

13.6 SUMMARY

In this unit, we started discussing Xamarin and PhoneGap tools. We looked at detailed steps for developing Android apps and iOS apps in Xamarin. We also looked at key features and advantages of Xamarin. We then looked at steps for developing a simple Android app and an iOS app using PhoneGap. Lastly, we saw the key advantages and disadvantages of cross platform app development tools.

13.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) .net
2) MonoTouch, MonoDroid
3) Application Layer
4) SDK manager
5) ViewController.cs

Check Your Progress 2

1) www
2) AndroidManifest.xml
3) single code base

13.8 FURTHER READINGS

References

1) http://developer.xamarin.com
2) http://stackoverflow.com/questions/tagged/monodroid+or+monotouch+or+xamarin?sort=active
3) http://blog.xamarin.com
4) https://developer.android.com
5) https://university.xamarin.com/resources/working-with-android-emulator
6) http://phonegap.com/

UNIT 14 PUBLISHING TOOLS AND DEVELOPER PROGRAM

Structure

14.0 Introduction

14.1 Objectives

14.2 Google Play Store

14.2.1 Prerequisites for Publishing Android Apps to Google Play Store

14.3 Apple App Store

14.3.1 Prerequisites for Publishing Apps to Apple App Store

14.4 Windows Store

14.4.1 Pre-requisites for Publishing Apps to Windows Store

14.5 Summary

14.6 Solutions/Answers

14.7 Further Readings

14.0 INTRODUCTION

The official channel for distribution of mobile apps is the marketplace stores. Each vendor has a specific app store wherein certified mobile apps are available. The app stores provide the much needed governance for the apps ensuring their certification compliance, security compliance, content policy compliance and other regulations. The app stores is the single interface for app developers to post and market their apps.

The app stores also provide intuitive interface for the end users to easily search, discover the apps through search UI, categorization, recommendation, sorting by rating/popularity and other criteria.

In this unit, we will look at the details of app stores for Android, iOS and Windows apps.

14.1 OBJECTIVES

After going through this unit, you should be able to understand

- key prerequisites and steps for publishing Android apps to Google Play store;
- key prerequisites for publishing and detailed steps for publishing iOS apps to Apple app store; and
- key prerequisites for publishing, best practices and detailed steps for publishing Windows apps to Windows store.

14.2 GOOGLE PLAY STORE

Google play store is the marketplace for Android mobile apps. Once the Android app is developed, tested and ready to be published, we need to publish it to the Google play store so that users can download and use it.

Google also offers other media such as digital music, video, books, magazines, TV programs through play store. Google play store forms a single unified distribution channel for Google's digital offerings.

Google play store has over 2.7 million Android apps as on 2017.

14.2.1 Prerequisites for Publishing Android Apps to Google Play Store

The following are the essential prerequisites for publishing Android apps to Google play store:

- A fully developed Android app as per the specification.
- A .apk file which packages the app.
- Eclipse IDE connected to Internet.
- A valid credit card to pay the Google developer license fee and register for Google Developer program at https://console.developers.google.com/
- Two representative screenshots of the app.

The following are the steps that need to be followed for publishing Android app from Eclipse IDE to Google play store:

1) Open the Android app in Eclipse

2) Right click on the app and select Export and "Export Android Application" option (Figure 14.1)

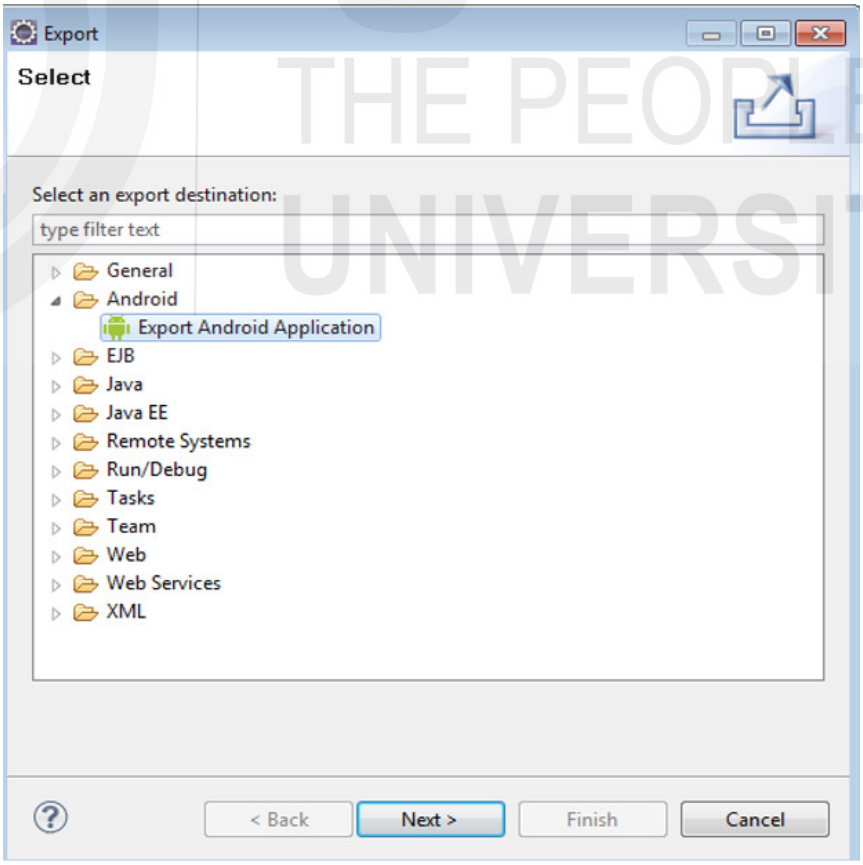

Figure 14.1 : Export Android application

3) Eclipse IDE will check for errors and if there are no errors we can continue to the next step.

4) In the next step we need to use the key store values (create keystore if not done already). This includes storing key store values to local file and entering password (Figure 14.2 and Figure 14.3).

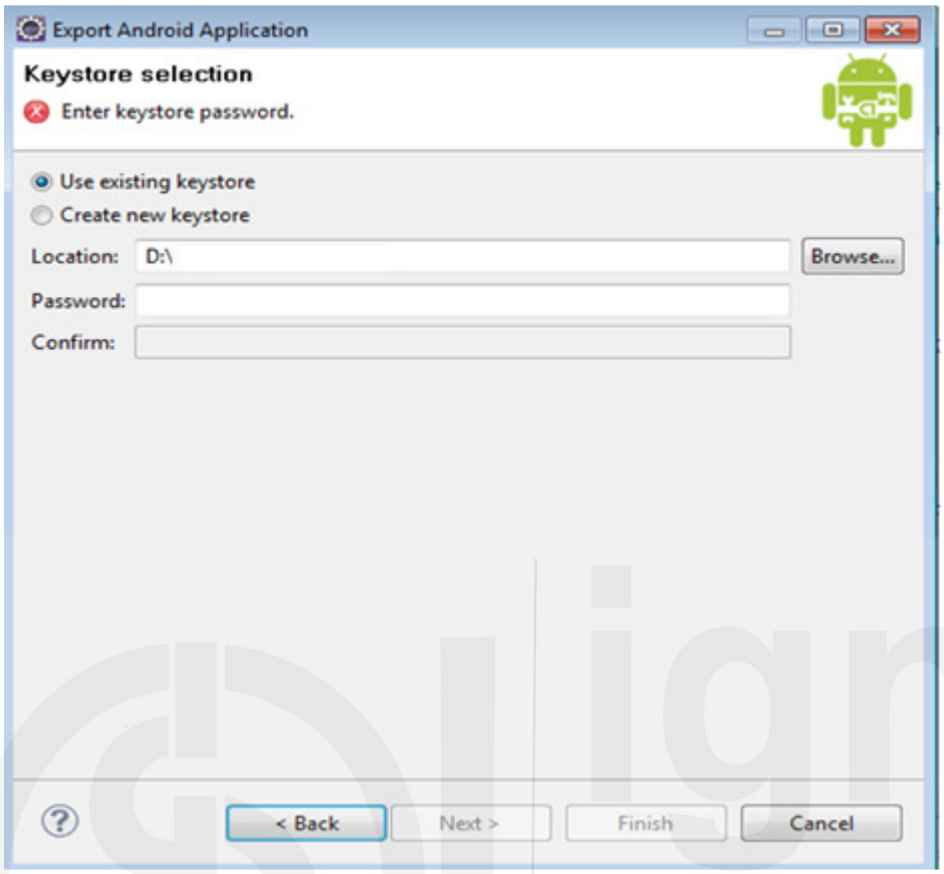

Figure 14.2 : Selection of key store

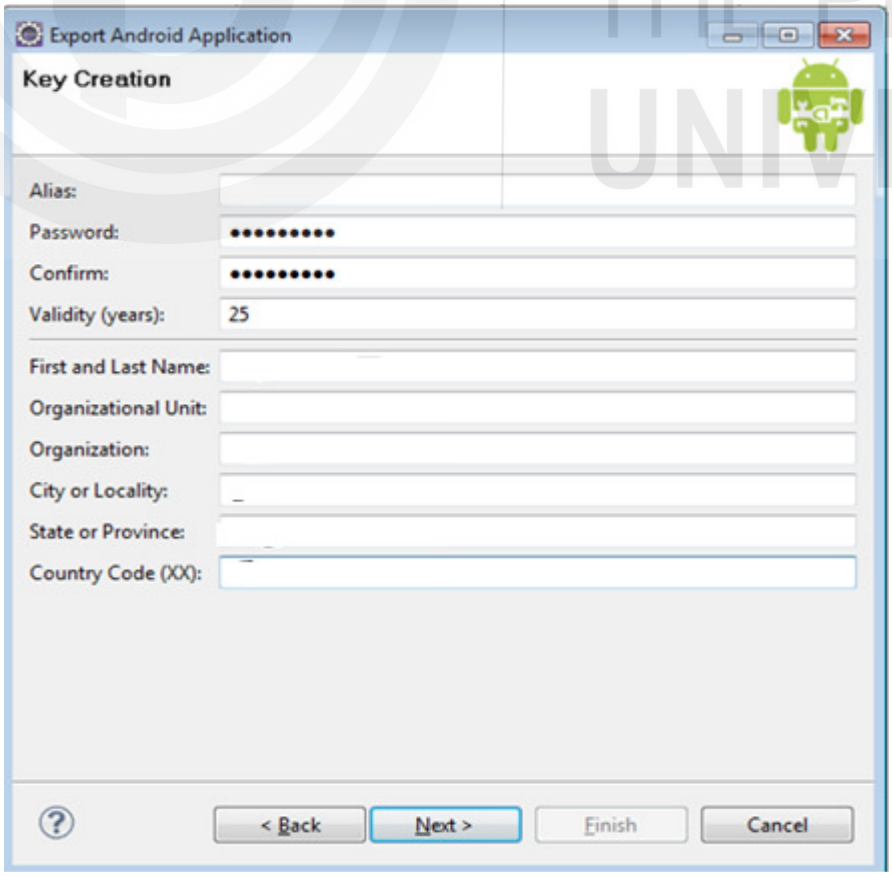

Figure 14.3 : Password creation

5) Exporting the .apk file to a valid destination.

6) Open Google Play Store (https://play.google.com/apps/publish) and log into Google developer console.

7) Select “Add new application” and select the application (Figure 14.4)

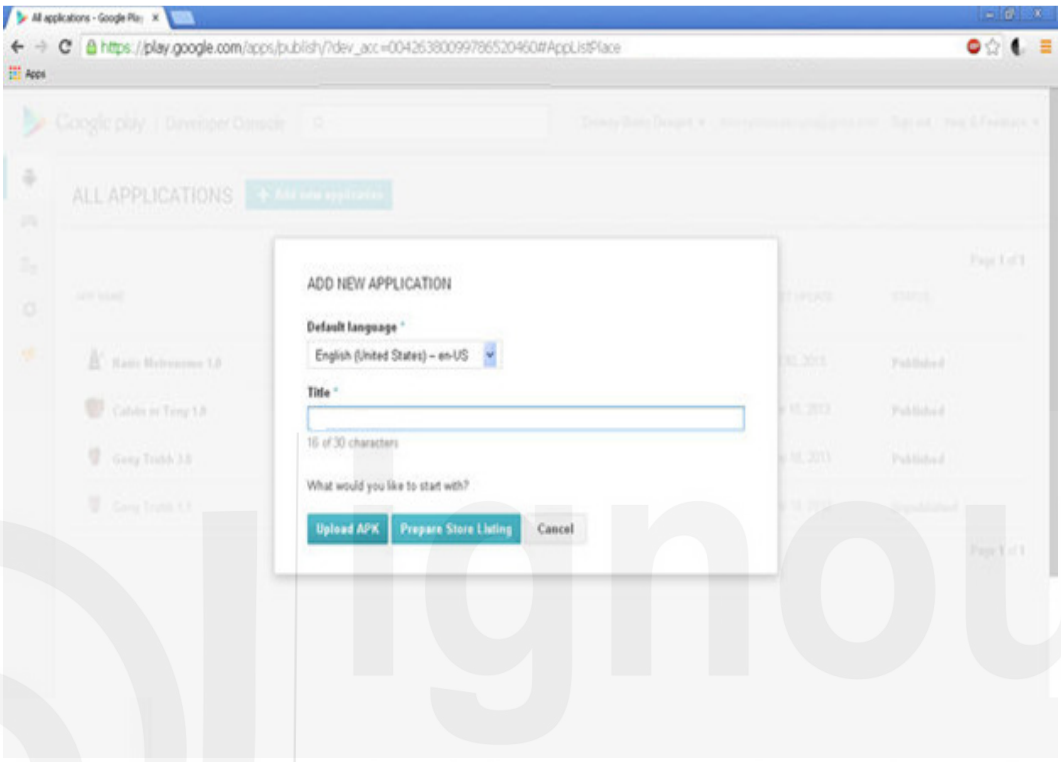

Figure 14.4 : Adding New Application

8) Upload .apk file to production (Figure 14.5)

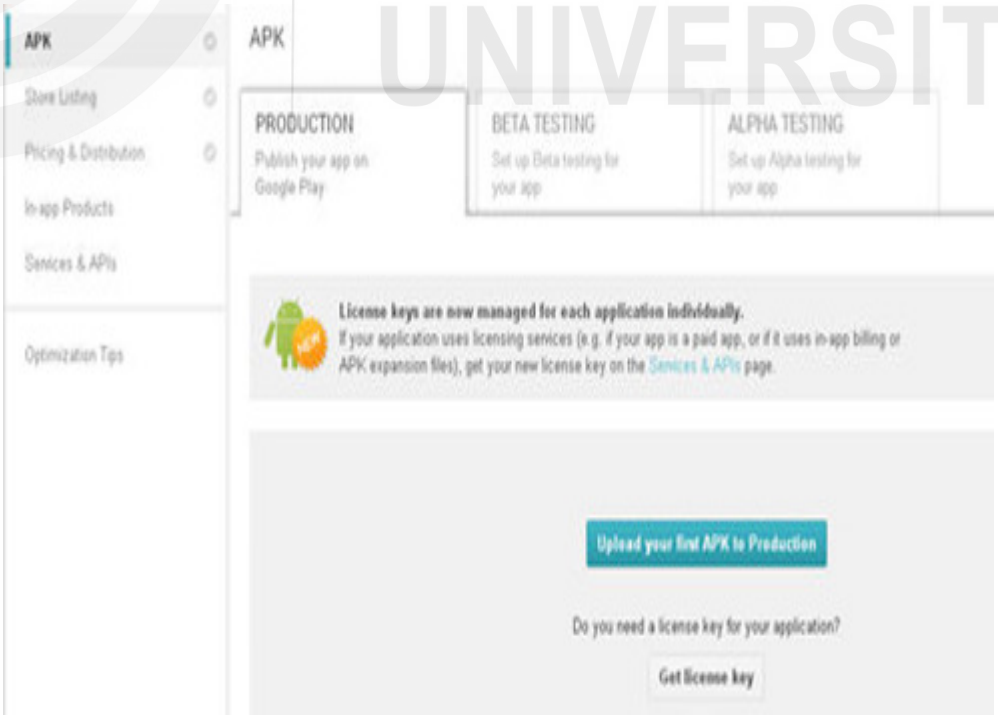

Figure 14.5: Upload .apk file

9) Once the upload is successful we need to add other details such as app description, screenshots, app icon, app categorization, contact details, privacy policy, maturity level details and such. We need to provide Website, email, and phone where customers can contact developer.

10) In the "Pricing and distribution" section we can specify if the app is free or paid, distribution countries, compliance to content guidelines and US export laws and such. For content ratings, guidelines can be referred at (https://support.google.com/googleplay/answer/6209544?hl=en-IN) (Figure 14.6).

Figure 14.6 : Pricing and Distribution

11) After all the steps are completed, we can select "Publish this app" which publishes the app to Google play store and it will be live in few hours.

14.3 APPLE APP STORE

App Store is the official app distribution platform developed by Apple. It hosts all apps for Apple devices such as iPhone, iPad, Apple TV, Apple watch etc. As on 2017 Apple app store has more than 2.2 million apps.

14.3.1 Prerequisites for Publishing Apps to Apple App Store

The following are the prerequisites for publishing apps to Apple app store:

- Register at iTunes connect at https://itunesconnect.apple.com
- Register at provisioning portal at https://developer.apple.com/iphone

The following are high level steps for distributing iOS apps to Apple app store:

1) Register as iOS and Apple Developer
2) Create profile, certificate and register devices
3) Submit app for approval

Its possible to register as Apple Developer for free. Then, you can enroll for paid iOS developer program. Its possible to enroll as individual or as a company.

The following is the process for creating a profile, certificate and register devices:

All apps downloaded from Apple app store have signed Apple certificate. Apple devices verify the signature before executing the app.

Apple provides iOS provisioning portal for performing these activities. The portal creates "profiles", "code signing entities" that enable Xcode to sign the apps so that Apple devices can securely verify the apps' authenticity. There are mainly two kinds of profile: development profile (that restrict the devices on which the app runs) and distribution profile (wherein Apple has to sign the apps). Details of distribution certificate are available at https://developer.apple.com/ios/manage/distribution/index.action

Provisioning portal is used for creating certification profiles and for registering the devices.

The following are high level steps involved in this stage:

1) Create and download the certificates for development and distribution.
2) Using the Unique Device Identifier (UDID), register the Apple device.
3) Create the device-specific unique App Id.
4) Create development and distribution profile using the profile name, App id, certificate and device.

The following are the steps to submit app to the app store:

1) Open the URL https://itunesconnect.apple.com and sign digital contracts.
2) Fill up contact details, bank information and tax forms.
3) Provide following details for the app (Figure 14.7).
    a) App's Name, Description, app image icon (512px by 512px), at least one screenshot
    b) Company name
    c) Rights
    d) Availability date
    e) Price (paid or free)

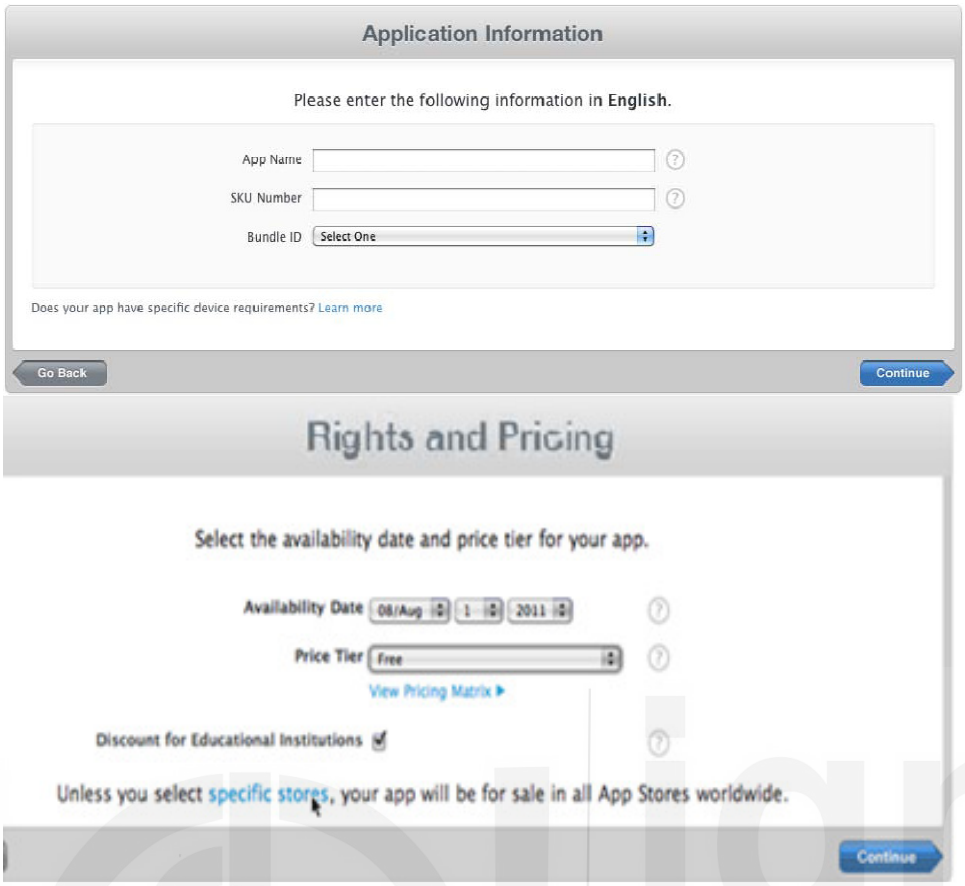

Figure 14.7 : Rights & Pricing

2) Provide the app's version information that includes app description, keywords, primary description, secondary description, copyright, support email address, and support URL, marketing URL, review notes and such (Figure 14.8).

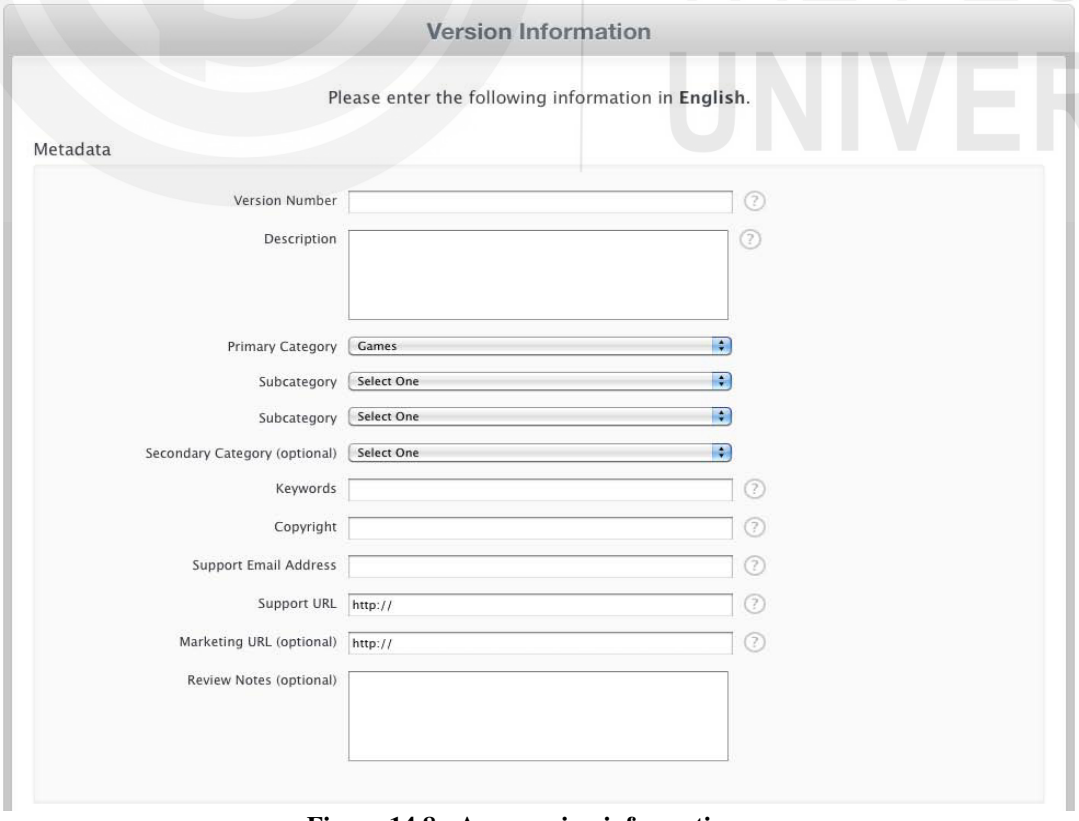

Figure 14.8 : App version information

3) Provide the app rating details as shown in Figure 14.9

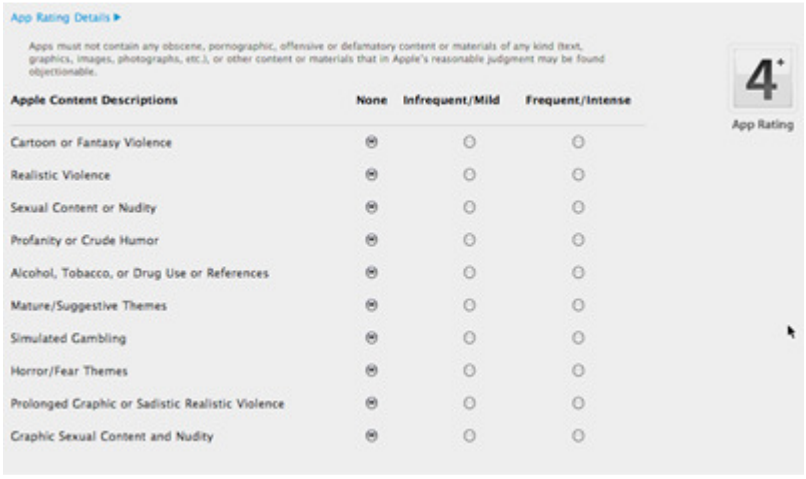

Figure 14.9 : App rating information

4) Provide the End User License Agreement (EULA) information (Figure 14.10)

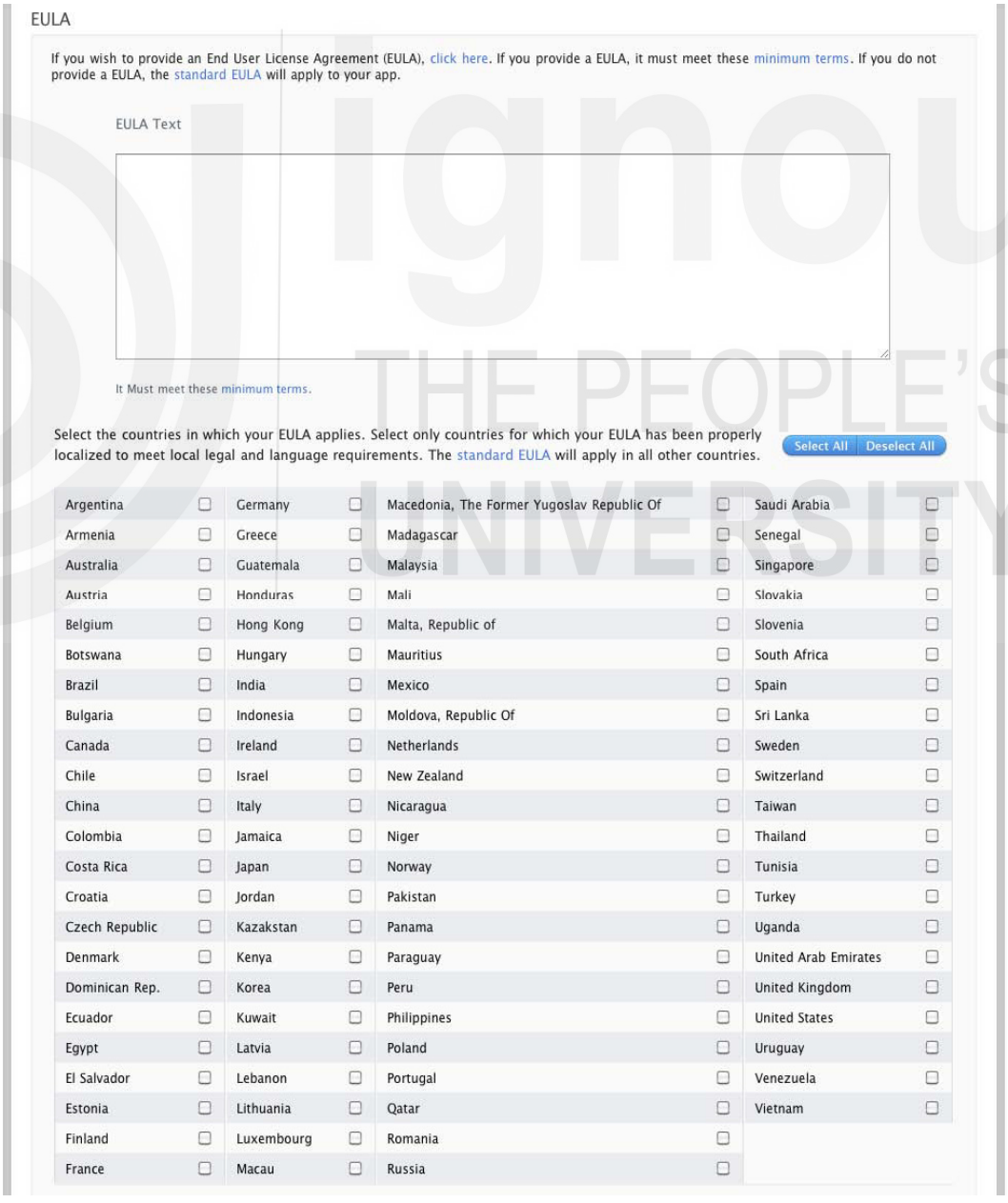

Figure 14.10: App EULA information

5) Provide various app screenshots (Figure 14.11).

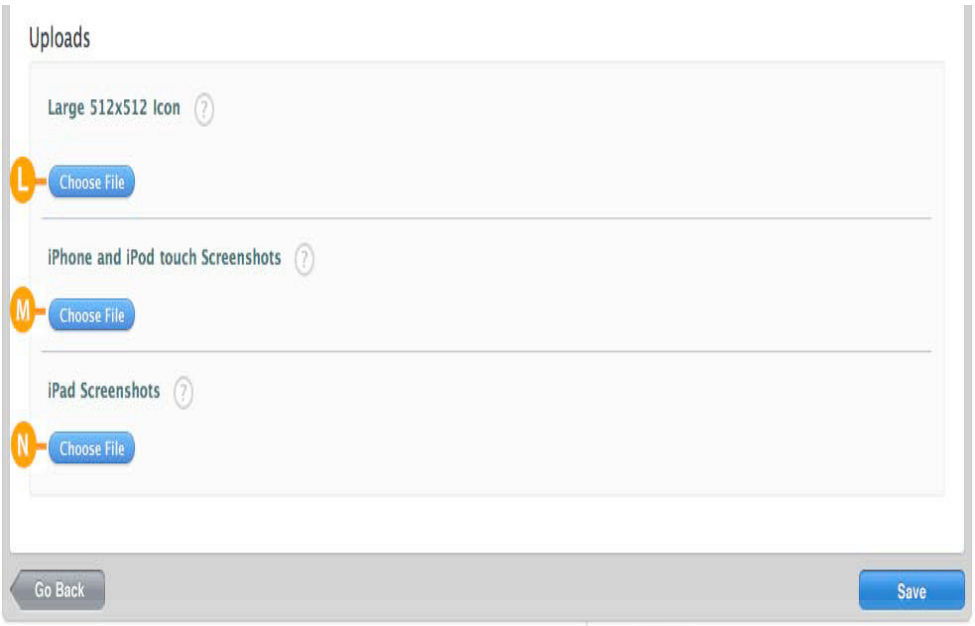

Figure 14.11 : App screenshots

6) After all the details are filled, app can be uploaded.

7) Apple reviews the app and upon successful review, it will be available in the app store.

☛ Check Your Progress 1

1) _____ file is the packaged Android app file that can be deployed to Google Play store.

2) _____section in Google play store provides fields for specifying the price details of the Android app.

3) _____ information of the app is used for registering the Apple device.

14.4 WINDOWS STORE

Windows store is the official distribution channel for certified windows apps. Microsoft revamped the UI and brought the concept of apps from Windows 8 onwards to Microsoft devices. Developers can deploy, distribute and sell apps from Windows store. Users can discover apps belonging to various categories from Windows store. Visual Studio is used for developing Windows apps.

Windows 8 apps use various intuitive features such as full screen page layout, touch enabled features, communication with other apps for seamless user experience. Most of the apps uses tiles layout which can be used for pushing updates and live alerts.

Windows apps can be created either by any of the following groups of technologies:

- HTML 5, CSS 3 and JavaScript
- C#/Visual Basic/C++ and XAML

14.4.1 Pre-requisites for Publishing Apps to Windows Store

The following are the prerequisites for publishing apps to Windows store:

- Windows developer account should be created at https://appdev.microsoft.com/StorePortals/en-us/Account/Signup/Start
- License agreements should be accepted and the subscription fee should be paid.

The following are some of the best practices that would help Windows app to get certification:

- It should adhere to the Microsoft's UX guidelines. MS Visual Studio provides in built templates that conform to the specified UX guidelines.
- It should adhere to Microsoft's content policy. The advertisement should not interfere with end user experience.
- It should publish its data and content privacy policy.
- The launch time of 5 seconds or lesser and suspend time of 2 seconds or lesser should be adhered to.
- It functions should be accessible when the resolution is 1024 x 768 or above.
- It should provide technical support information.
- Windows App Certification Kit can be used to test the compliance level of the app.

The following are for publishing apps to Windows store:

1) Develop the app using any of the technologies mentioned earlier. A sample app project is shown in Figure 14.12.

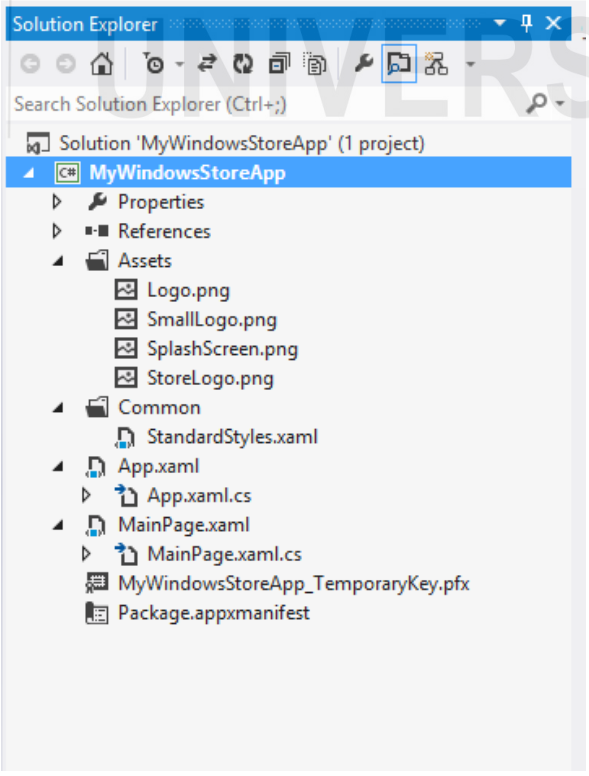

Figure 14.12: Windows app project

2) Prepare the app for publication. Provide all the app details such as app description, display name, app capabilities (such as access requirements for contacts, gallery, GeoLocator etc.), version, declarations, title images, logos (including small logo, wide logo, store logo, badge logo) as shown in Figure 14.13.

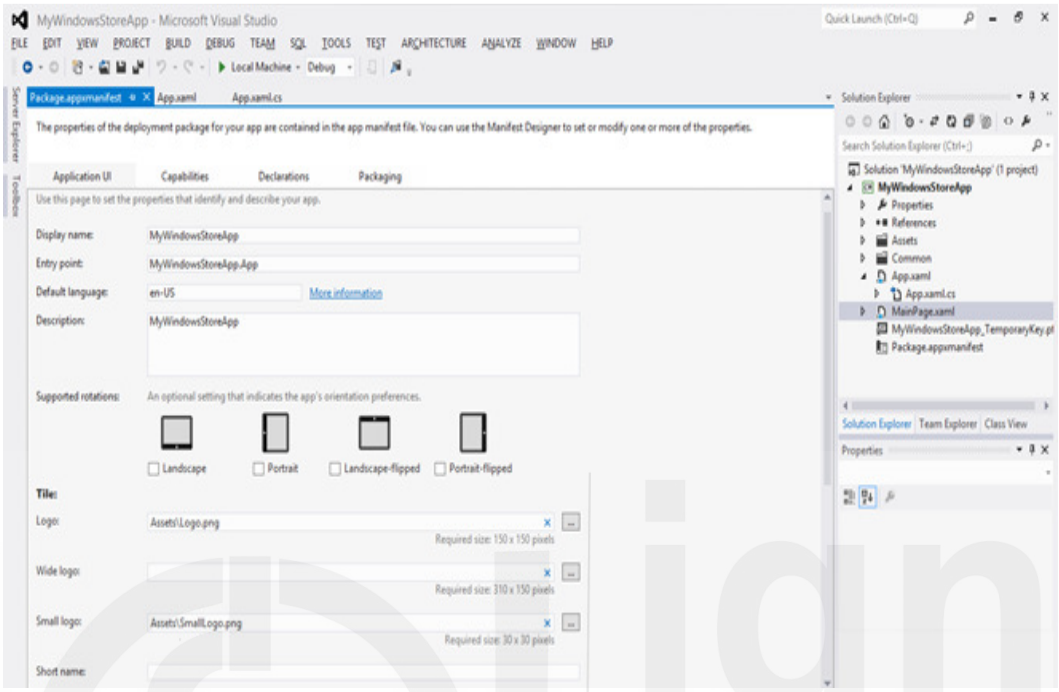

Figure 14.13: Configuring Windows app

3) Create the app package using project → store → create app package. While packaging the app, select the option to package the app for Windows store as shown in Figure 14.14.

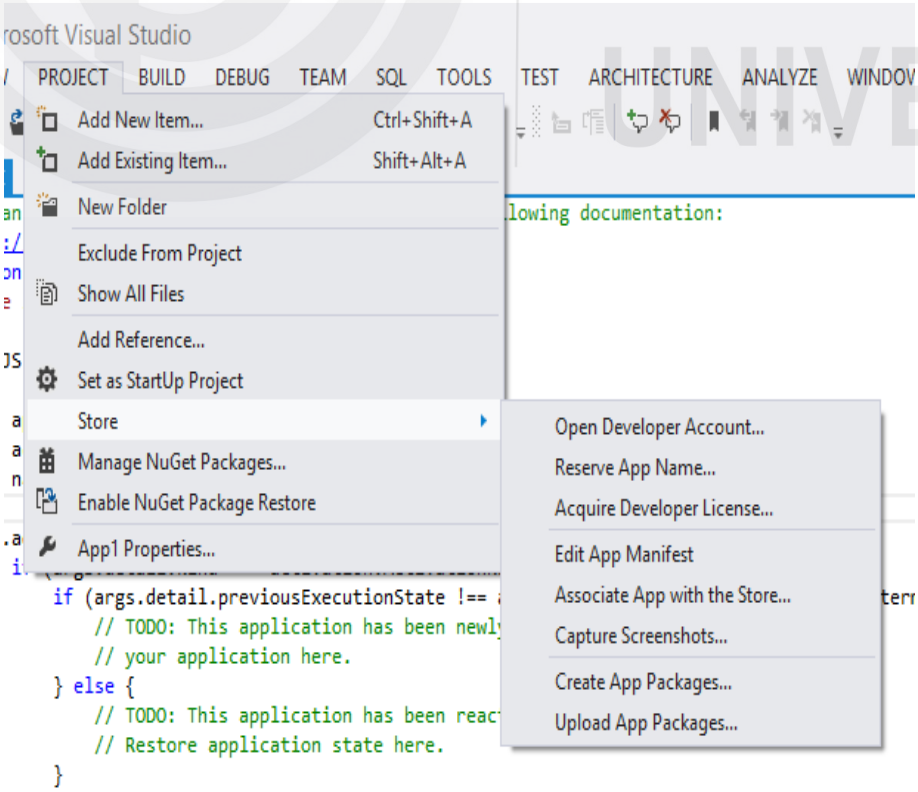

Figure 14.14: Creating a detailed app profile

4) Execute App certification SDK to test the technical compliance of the app to Microsoft guidelines.

5) Create a detailed app profile through project → store → Reserve app name or through online at http://go.microsoft.com/fwlink/?LinkId=256672&clcid=0x409

   In this step, specify the unique name for the app, age rating, app description, payment details (free or paid), distribution region and other details (Figure 14.14).

6) Upload the app package through project → store → upload app package or through URL http://go.microsoft.com/fwlink/?LinkId=256671&clcid=0x409. This submits the app to Microsoft for certification.

7) Developer can check the details such as downloads, ratings, reviews, conversion, usage details, payment etc. at http://go.microsoft.com/fwlink/p/?LinkId=223974.

After the app is submitted to the Windows store, developers can view the certification status of the app at http://go.microsoft.com/fwlink/p/?LinkId=223974.

The high level app certification steps are specified in Figure 14.15.

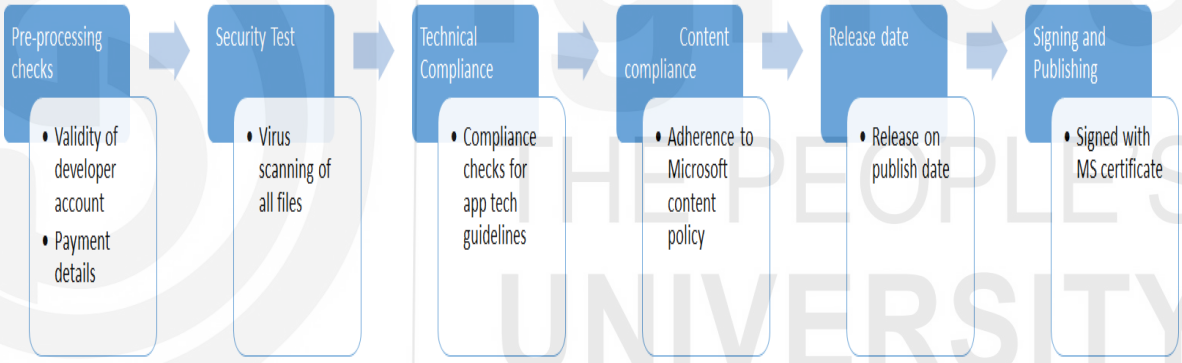

Figure 14.15: Windows app certification steps

☞ Check Your Progress 2

1) The two development technologies for windows app are _____ and _____ .

2) _____ can be used to test the technical compliance for the windows app.

3) Content compliance check includes checking the adherence of app content to _____ .

14.5 SUMMARY

In this unit, we discussed various mobile app stores that can be used for deployment and distribution of mobile apps. The pre-requisites and publishing steps for Android apps in Google Play store, iOS apps in Apple app store and

Windows apps in Windows store are explained. We also discussed some best practices for Windows app development so that it gets certification.

14.6 SOLUTIONS/ANSWERS

Check Your Progress 1

1) .apk
2) Pricing and distribution
3) UDID

Check Your Progress 2

1) HTML 5, CSS 3 and JavaScript and C#/Visual Basic/C++ and XAML
2) App certification SDK
3) Microsoft content policy

14.7 FURTHER READINGS

References

1) http://developer.android.com/
2) http://developer.android.com/guide/publishing/publishing.html#market
3) http://developer.android.com/guide/publishing/app-signing.html
4) http://developer.android.com/guide/publishing/preparing.html
5) http://developer.apple.com/
6) https://developer.apple.com/support/ios
7) https://itunesconnect.apple.com
8) https://developer.apple.com/iphone
9) http://www.windowsstore.com/developers/windows
10) http://msdn.microsoft.com/library/windows/apps/br211361.aspx
11) http://msdn.microsoft.com/en-us/library/windows/apps/hh974581.aspx
12) http://msdn.microsoft.com/en-us/library/windows/apps/hh974578.aspx
13) http://msdn.microsoft.com/en-us/library/windows/apps/jj657972.aspx
14) http://msdn.microsoft.com/en-us/library/windows/apps/hh694062.aspx
15) http://msdn.microsoft.com/en-US/windows/apps/br229512
16) http://msdn.microsoft.com/en-us/library/windows/apps/hh921583.aspx
17) http://msdn.microsoft.com/en-us/library/windows/apps/hh694083.aspx

UNIT 15 MONETIZATION

Structure

15.0 Introduction

15.1 Objectives

15.2 Monetizing Android Apps

- 15.2.1 Freemium Distribution Model
- 15.2.2 Paid Distribution Model
- 15.2.3 Affiliate Partnership Model
- 15.2.4 Commission Model
- 15.2.5 White Labeling Model
- 15.2.6 Publishing to Amazon App Store

15.4 Monetizing iOS Apps

15.5 Monetizing Windows Apps

15.5 Choosing Right Mobile Ad Strategy

- 15.5.1 Defining the Success Metrics
- 15.5.2 Mobile Ad Selection Based on Business Strategy
- 15.5.3 Monitor Ads

15.6 Summary

15.7 Solutions/Answers

15.8 Further Readings

15.0 INTRODUCTION

One of the key success factors for mobile apps may be overall revenue generated from the app apart from number of downloads etc. As app development needs significant investment and planning, monetization and revenue generation plans form part of fundamental aspects of mobile app strategy. We need to carefully explore all avenues of revenue generation and chose the most optimal one that aligns with overall business strategy.

In this unit,we will look at various avenues for monetization through mobile apps. We have detailed various ways to generate revenue from Android apps, iOS apps and Windows apps.

15.1 OBJECTIVES

After going through this unit, you should be able to understand:

- various monetization ways for Android apps such as free model, paid model, freemium model, affiliate model, commission model, white labeling model and publishing by Amazon app store;
- various ways to monetize iOS apps such as services model, sponsorship model, and Facebook ads model;
- various ways for monetizing Windows apps such as affiliate ads, community ads, and donation model; and

15.2 MONETIZING ANDROID APPS

Android platform provides great opportunity for monetization as it is one of the most popular mobile platforms. The following are some of the popular ways in which Android app developers can derive revenue out of apps:

15.2.1 Free Distribution Model with Ads

Offering Android apps for free is one of the popular ways to monetize the app. Free version of the apps feature advertisements (ads). About half of the available Android apps feature advertisements. App developers would derive revenue based on the click-through rate (CTR). CTR basically is the ratio of the ads users clicked to total number of displayed ads. For each ad click, the developer would get a specific amount. There are other events wherein the advertiser pays the developer such as ad view, ad click, ad conversion, ad events and such. Size and placement of ads also play an important role in CTR. Full screen ads have high CTR but they could decrease the popularity of the app. Normally full screen ads are displayed in game apps when user jumps from one level to another. Other ways to increase the CTR through ads are given below:

- Include rich media content (such as video content) in the ads
- Display the relevant ads relevant for the user's context and based on user's history of transactions.
- Include interactive content such as questions, teasers in the ad text. Provide the incentive for the user through ads.
- The ads must be non-intrusive and should not interfere or severely impact user app experience.

AdMob from Google and InMobi are some of the popular mobile ad platforms.

15.2.2 Freemium Distribution Model

Freemium is a free app with some premium content containing in-app purchases. In a typical gaming app, though the app is free, some of the key essentials needed for the full gaming experience (such as currency, higher levels, unlocking a feature, extended features, virtual coins etc.) are paid entities. Users need to purchase within the app (termed as in-app purchase) to access the premium content. This is also a popular model to monetize the app. Normally, freemium apps need more programming effort and maintenance effort. The following are some of the key points for designing an effective freemium app:

- Market the app so that it is popular in its category.
- Identify the key essentials of app and provide premium access to it.
- Decide a one-time purchase or a recurring subscription purchase option.

15.2.3 Paid Distribution Model

End users need to pay for these apps while purchasing from Google play store. App developers can fix the price and the distribution region. The following are some of the best practices for designing a successful paid app:

- Market the app in all the distributed regions.

- Release the demo app which is a limited free version of the app to generate interest among the users.
- Design a compelling user experience.
- Start with a free app and gauge the user feedback and response and then launch a paid app once the free app reaches critical mass.
- Promote the app through social networking sites, emails and SMS.

15.2.4 Affiliate Partnership Model

The apps which have lot of content would become affiliates to derive revenues. Affiliate ads drive the user traffic to ad sites. With a strong content strategy, the app can act as a brand marketing channel for the products and services.

15.2.5 Commission Model

If apps form the medium of transaction, app developers would receive a commission for each transaction that happens through that app. The marketplace apps, retail apps, e-commerce apps which bring buyers and sellers to the same platform are typical examples which work in this model.

15.2.6 White Labeling Model

App developers can sell their source code to various companies with similar needs. Companies can reskin and rebrand the app for their own needs after purchasing the app's source code.

15.2.7 Publishing to Amazon App Store

Android apps can also be submitted to Amazon app store. This provides wider reach to Amazon customers and better app governance provided by Amazon. This would further help to gain additional revenue from Android Apps.

The following are high level steps for distributing Android apps from Amazon app store are given below:

1) Create a developer account at https://developer.amazon.com (Figure 15.1).

2) After successful registration and login, provide all the details such as profile information (name, address, company name etc.), customer support email etc. (Figure 15.2).

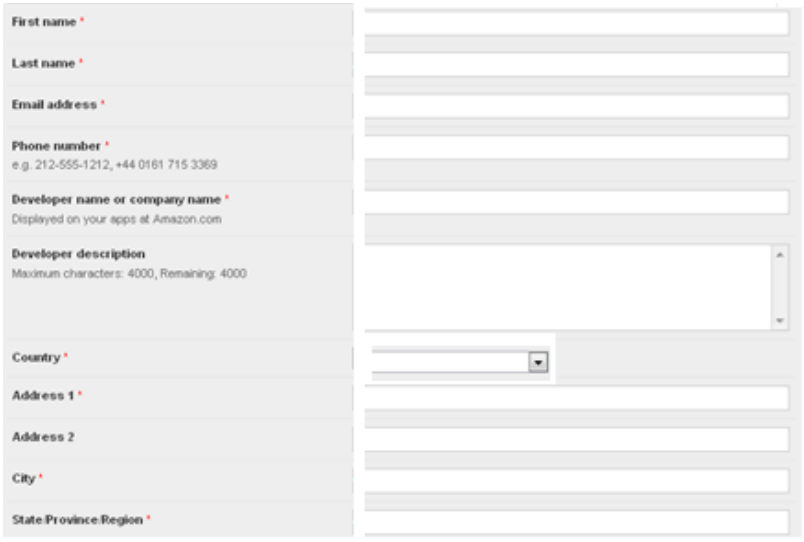

Figure 15.1 : Account creation at http://www.amazon.com

3) Accept developer license agreement and print the agreement.

4) Provide the royalty payment details

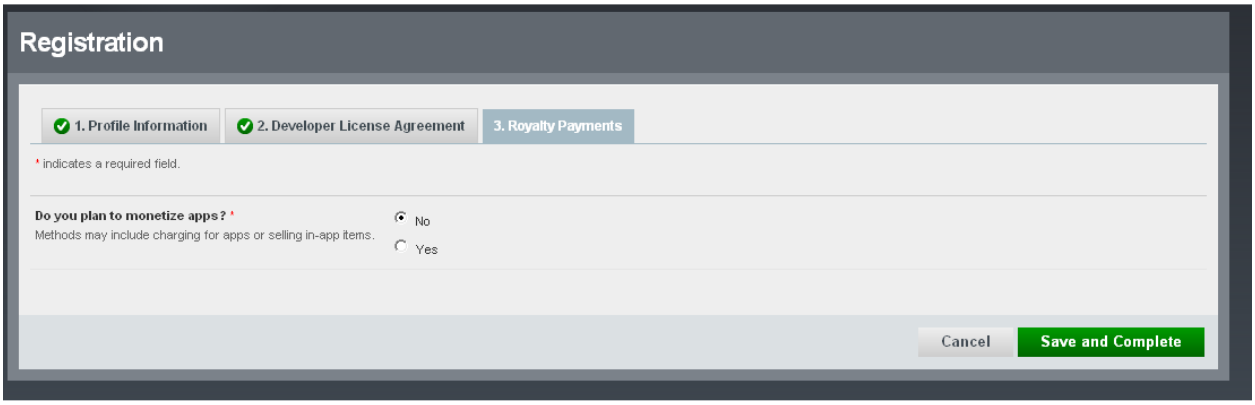

Figure 15.2 : Developer registration at Amazon

5) Add the app details such as app tile, form factor, support details, availability and pricing details etc. (Figure 15.3 and Figure 15.4)

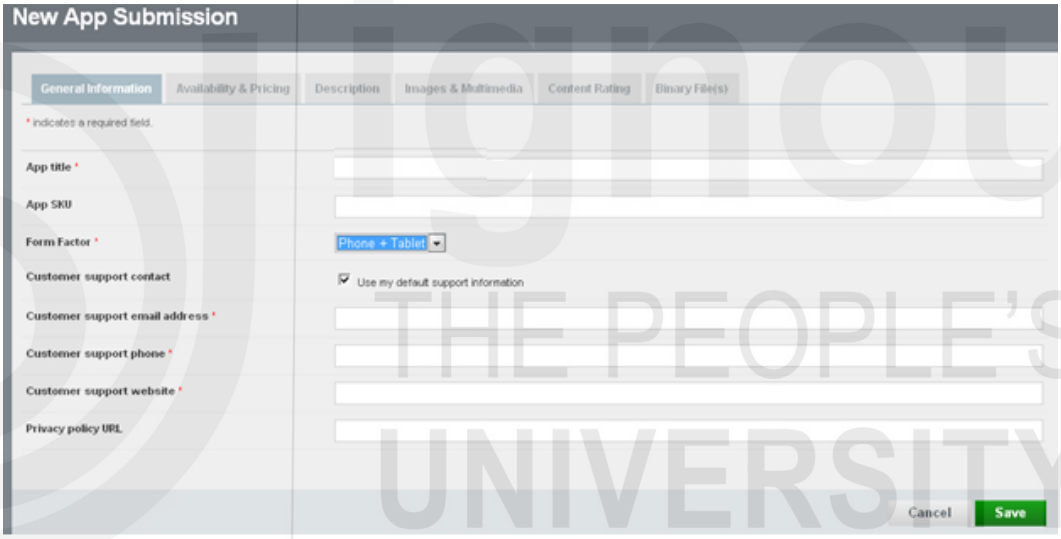

Figure 15.3 : New app submission

Figure 15.4 : Details about app

6) Provide additional app details such as category, language, short description, long description, keywords etc.(Figure 15.5).

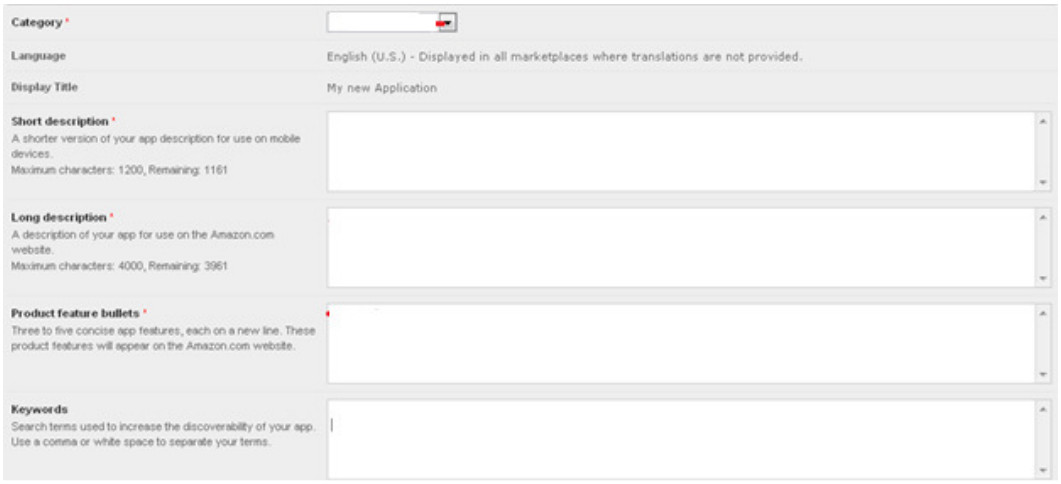

Figure 15.5 : Description of app

7) Provide the app screenshots and content rating and then, submit the app binary.

15.3 MONETIZING IOS APPS

Most of the monetization techniques that were discussed for Android apps such as free distribution model, freemium model, paid module, affiliate model, commission model are applicable for iOS apps as well.

The following are some more monetization techniques for iOS apps:

Apps for mobile-first strategy

iOS apps can be used to implement the mobile-first strategy. Apps can be used to showcase and promote the company's products and services. E-commerce apps can be used to sell the products, services and merchandise.

Revenue through services

If the app has valuable data or business logic, the key features of the app can be exposed as services for third party consumers. Apps can expose analytics functions, demographics data, user interests (subject to privacy policies), and geo-location data as services.

Sponsorship apps

App developers can secure the sponsorship for their apps based on the ad content, themes, events, services and topics. For instance, a movie app can secure the sponsorship for promoting a recently released movie from distributors.

Using Facebook ads

iOS developers can leverage the "Facebook audience network" to monetize their iOS (and Android) apps through Facebook ads. Developers can implement the Facebook's audience network feature in their apps. When the end user is using the app, unless the user has chosen "opt out feature", or if the corresponding user does not exist in Facebook, an ad will be shown.The user information is mapped to the Facebook user information for displaying ads.

Primarily, there are three types of Facebook ads which are explained below:

- **Banner ads**: These ads can be place anywhere in the app. Developers can also set the refresh time for this ad. The size of banner ads is restricted to max height of 50 px for mobile and 90 px for table with configurable width.
- **Interstitial ads**: These are full screen ads normally used for games.
- **Native ads**: Developers get flexibility with these ads in terms of location, look,feel and size. The ad title should contain maximum of 30 characters with cover image of 1200 x 627 px and the body text is restricted to 90 characters.

The following are the detailed steps for implementing Facebook ads in iOS apps:

1) Go to https://developers.facebook.com/products/app-monetization/audience- network/ and apply for the audience network (Figure 15.5)

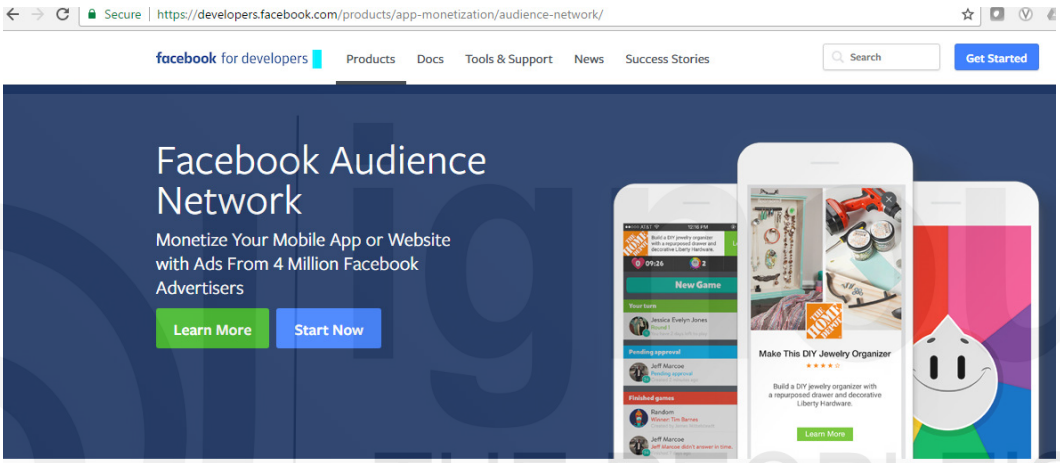

Figure 15.5 : Facebook Audience Network

2) Provide the app details and user details to apply for Facebook audience network (Figure 15.6)

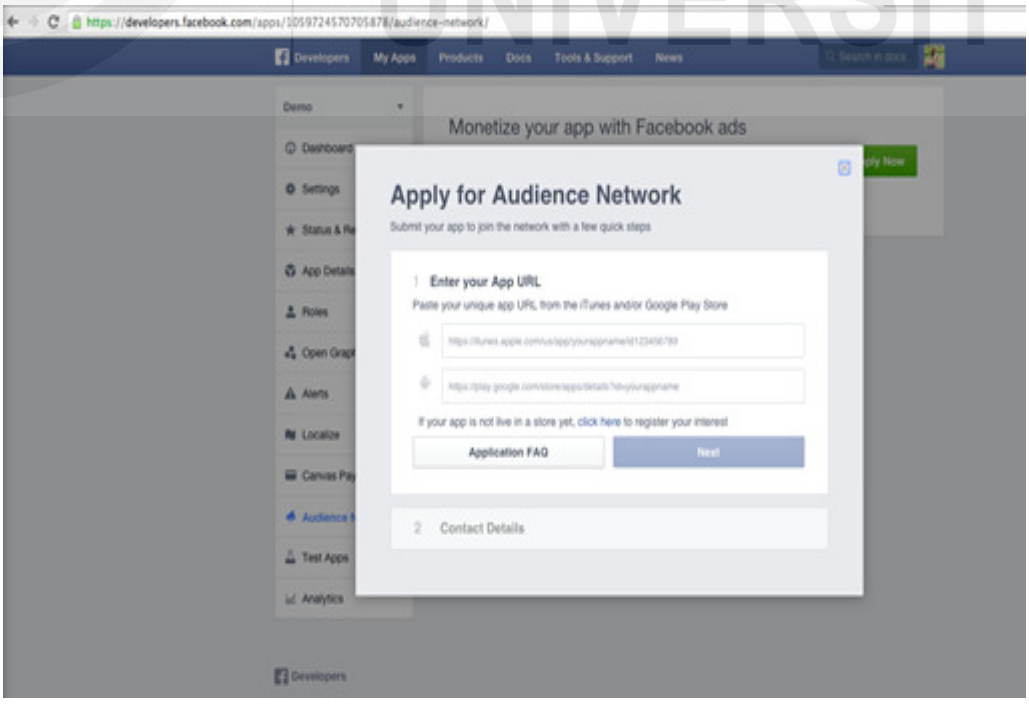

Figure 15.6: Applying for audience network

3) Then, connect with an Audience network specialist (Figure 15.7).

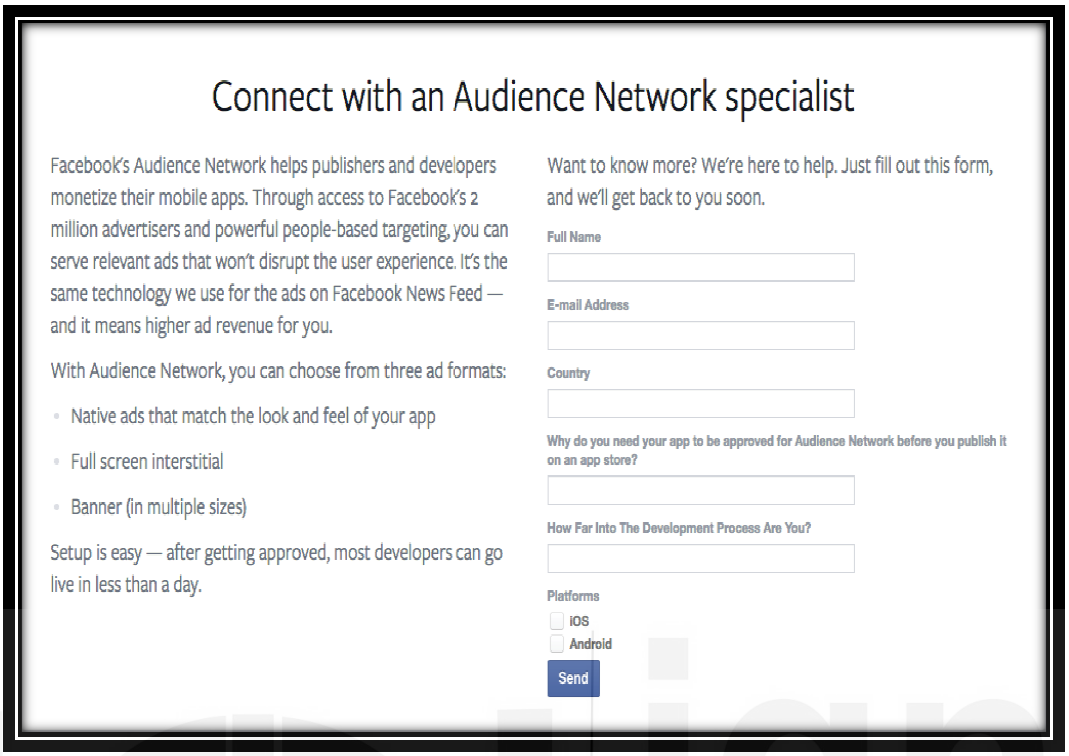

Figure 15.7 : Connecting with Facebook Audience Network Specialist

4) After submitting all the details, the submission is acknowledged as shown in Figure 15.8.

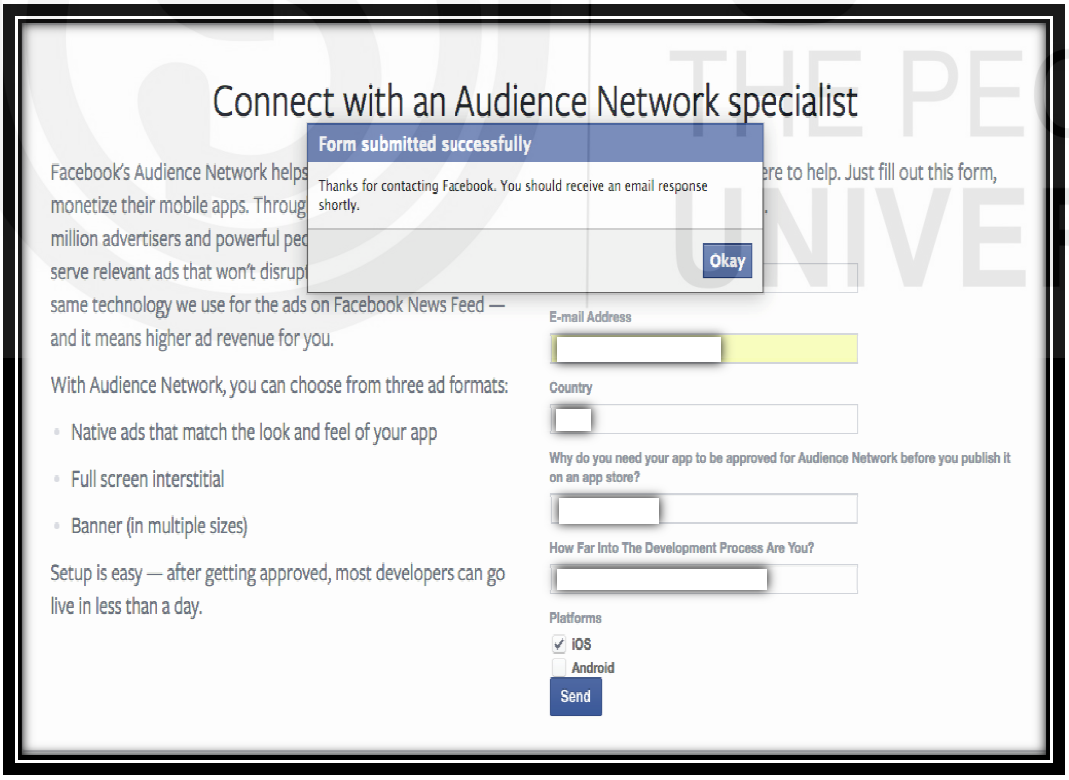

Figure 15.8 : Acknowledging the submission

5) After registration, the app developer can access the audience network section (Figure 15.9).

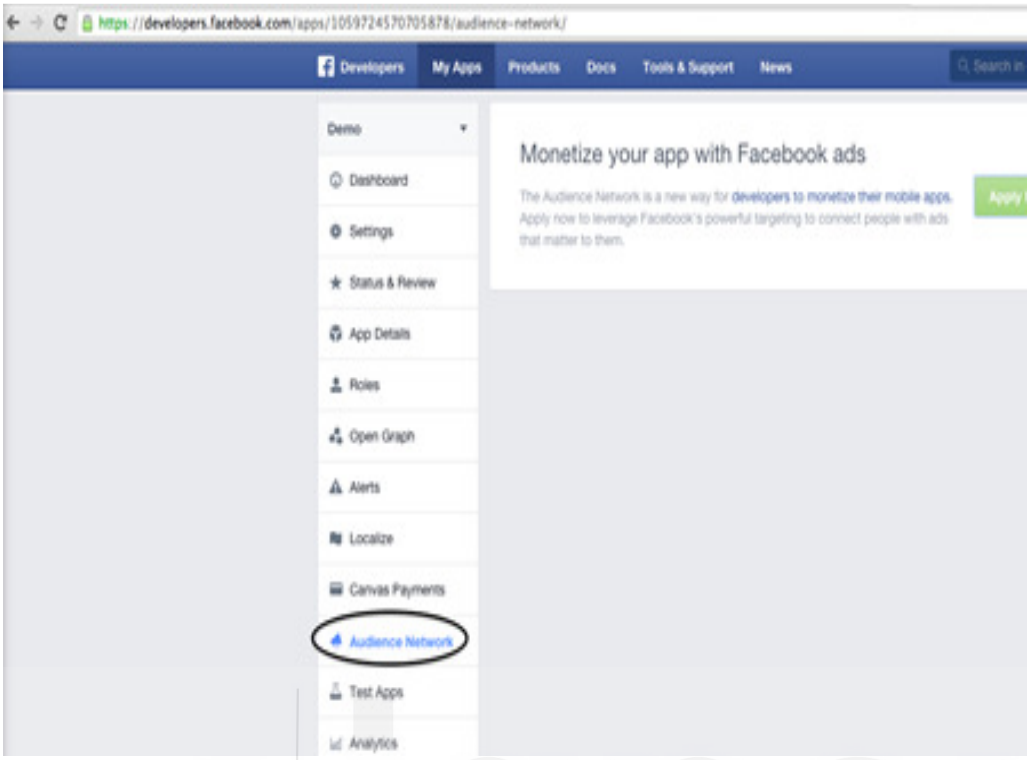

Figure 15.9 : Accessing the Facebook Audience Network

6) Download and include the "Audience network library" which is part of Facebook iOS SDK. The FBAudienceNetwork.framework , AdSupport, StoreKit and CoreMotion can be added to Xcode project (Figure 15.10).

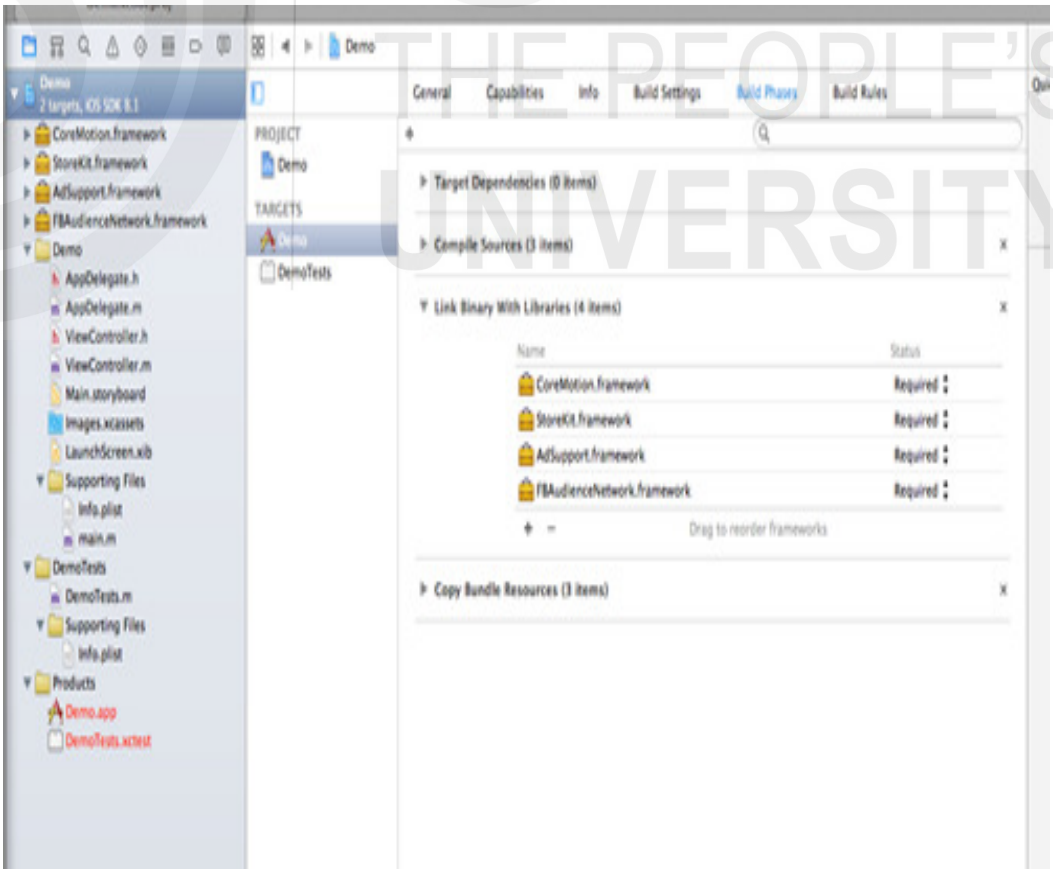

Figure 15.10 : Audience Network Library

7) Use placement id that identifies each Facebook ad unit and add the ad units into app using the APIs.

☞ **Check Your Progress 1**

1) CTR is the ratio of the ads _____ to _____.

2) Marketplace apps, retail apps, e-commerce apps are example of ___ model.

3) Selling app source to other companies is done in _____ model.

4) Exposing key app functionality as _____ is a monetization model.

5) The ads that are placed at top or bottom of the screen are called _____.

6) _____ ads occupy full screens.

7) _____ library in Facebook iOS SDK is used for ads.

15.4 MONETIZING WINDOWS APPS

Paid, free, freemium, white labeling monetization models discussed earlier can be used for Windows apps as well. In addition, the following are other popular monetization techniques for Windows apps:

Affiliate ads from Microsoft

Windows app developers can leverage the Microsoft affiliate ads to show promotions for music, games from Windows store. Based on the conversion rate, commission will be paid to the app developers.

Ads from Community

Windows app developers can select this option to get ads published from other developer communities.

Donation model

Not-for-profit and charity companies normally design apps that seek donations from people. They can post the app to Windows store.

15.5 CHOOSING RIGHT MOBILE AD STRATEGY

As ads play role in app monetization, let us look at some of the key concepts in this regard.

15.5.1 Defining the Success Metrics

The first stage is to define the Key Performance Indicators (KPI) and metrics for mobile ads. Two popular metrics are Click through rate (CTR) and Cost per thousand impressions (CPM).

- CTR is total number of ads clicked by user divided by total number of displayed ads. In this case, the revenue model will be based on CTR.
    - CTR = (ads clicked by user)/ (total ads displayed)

- CPM is the effective cost per 1000 impressions.
    - CPM = (Advertising cost/Number of impressions) X 1000

Once we define the revenue model and the metrics we can accordingly use the ads.

15.5.2 Mobile ad Selection Based on Business Strategy

In this step, we will select the ad that is appropriate for business strategy. Key ad selection criteria are as follows:

- Based on the business needs, we can select the ad type: banner ad, full screen ad or custom ad.
    - Banner ads are normally placed at the top or bottom of the screen
    - Full screen ads (or Interstitial ads) have high CTR rate but carry high risk of user dissatisfaction. This is usually placed at the end of user flow (such as end of game) or when user navigates from one flow to another (such as user going from one game level to another).
    - Ads can be laid out in custom format with custom size and formats based on the business needs. For instance, ads can be placed in list format or in custom panel format.
    - Ads can be pushed to user through notification feature.
    - Form ads can be designed to collect the user information such as email address and contact details.
    - Many product companies are promoting their products through videos that showcase their product.
- Place the ad strategically so that it does not interfere with end user experience and at a location where it is minimally invasive and least distracting for the user experience.
- The content or rich media content within the ad should be relevant and responsive for high CTR.
- Experiment with various ad formats through A/B testing and multi-variant testing to fine tune the ad size and placement.
- Understand the ads that are effective and continuously fine tune the ad size and placement strategy.
- Ensure that ads are compliant with the marketplace policies, content policies and other regulations.

15.5.3 Monitor Ads

The usage and effectiveness of ads should be monitored through analytics dashboard to understand the CTR and CPM metrics. We can also gather other metrics such as clicks, views, and conversion etc.

☞ Check Your Progress 2

1) _____ can be used to show the Windows store promotions.

2) Ads published from other developers can be shown through _____ ads in Windows apps.

3) Cost per impression is _____.

4) _____ ads are pushed to users.

15.6 SUMMARY

In this unit, we started with discussion on various monetization techniques for popular mobile platforms. We looked at key monetization models such as paid model, free model, freemium model, affiliate model, white labeling model and Amazon app store publishing modelfor Android apps. We discussed services model, sponsorship model and Facebook ads model for iOS apps. We studied details of affiliate ads, community ads and donation model for Windows apps model. Finally, we had a deep dive into Mobile ad strategy by looking at key success metrics, ad selection process and monitoring process.

15.7 SOLUTIONS/ANSWERS

Check Your Progress 1

1) users clicked, total number of displayed ads
2) commission model
3) White labeling
4) Services
5) banner ads
6) Interstitial
7) Audience network

Check Your Progress 2

1) Microsoft affiliate ads
2) Community
3) effective cost per 1000 impressions
4) Notification

15.8 FURTHER READINGS

References

1) https://developer.amazon.com/
2) https://developer.amazon.com/help/faq.html
3) http://www.amazonappstoredev.com/

4) https://blogs.windows.com/buildingapps/2017/01/30/monetize-game-app-playtem-ads/#lWTsWQwg9E5xFTht.97

5) https://msdn.microsoft.com/en-us/library/windows/apps/hh202939(v=vs.105).aspx

6) https://docs.microsoft.com/en-us/windows/uwp/publish/monetize-with-ads

7) https://docs.microsoft.com/en-us/windows/uwp/monetize/

8) https://developer.microsoft.com/en-us/store/monetize/ads-in-apps

9) https://developer.microsoft.com/en-us/store/monetize

10) https://developer.apple.com/iad/monetize/Implementing-iAd-in-Your-iOS-Apps.pdf

11) https://developer.apple.com/app-store/business-models/