University of St Andrews

School of Computer Science

Timespan - Museum Without Walls Mobile Application

Mujtaba Mehdi

120022147

MSc in Advanced Computer Science

23 August 2013

Declaration

“I hereby certify that this dissertation, which is approximately 12957 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree. This project was conducted by me at The University of St Andrews from June 2013 to August 2013 towards fulfilment of the requirements of the University of St Andrews for the degree of MSc under the supervision of Dr. Alan Miller and Dr.Lisa Dow.”

"In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work."

Signature	**Date**
Mujtaba Mehdi	*23/08/2013*

Dedicated to my parents,

Ghulam Mehdi and Hamida Bano.

Acknowledgements

I would like to offer special thanks to Dr.Alan Miller who accepted the responsibility to supervise this project and offered valuable suggestions which guided this project to a successful completion. I would like to thank my second supervisor Dr. Lisa Dow for her guidance and help throughout the project.

I would also like to thank Timespan staff (Anna Vermehren and Jacquie Aitken) who provided me with the opportunity to work on this project. They were very helpful and offered a great support throughout the project. They provided the data and feedback without which this project would not have successfully reached its current state.

I would like to thank my family specially my parents (Ghulam Mehdi and Hamida Bano) and my brother Ather Mehdi who were always there to help me in every possible way. Last but not least I would like to thank the University for accepting me for MSc and my friends at St Andrews who made this year memorable.

Abstract

"Museum Without Walls" is a mobile application aimed at delivering an interactive user experience for a Museum - beyond the traditional instruments of presenting objects of interest using the latest mobile technology. The project is commissioned by *Timespan*, a creative heritage and arts development hub based in Helmsdale, Scotland. The primary aim of this project is to define the various parameters within the environment which a user interacts using a mobile phone - and then utilise the available options to make this a highly accessible and user friendly application. This project not only makes the whole subject of visiting museums more interesting, but also maximises the possibilities of better educating their visitors. The management of the Museum see this project as a great tool to attract new audiences whilst keep the existing ones highly engaged by using mobile technology.

The year 2013 is also highly significant in the history of Scotland as it marks the 200th anniversary of the tragic *Highland Clearances* that occurred in 1813. The several hundred tenants of the Strath of Kildonan were forced to abandon their homes and were dispatched to North America; making way for sheep farming which was a booming trade at the time. The painful ordeal suffered by the tenants is still considered one of the most demanding journeys faced by European emigrants to North America.

Very recently *Timespan* came out with their iOS application which delivers a virtual tour of the Highland Clearances and provides a highly educational experience to the visitors who would like to learn about the history of the *Strath of Kildonan*. The project discussed in this report is a continuation of the same application. However it aims to extend the reach of the application from Apple iOS platform to other mobile platforms. The major highlights of this App include: *The Kildonan Trail*, a GPS triggered geo-referenced historical map and a user interactive book called *The Clearances Story*. The application has not only been fully conceived, but the Android version of it has already been ported successfully.

Table of Contents

List of Figures

List of Tables

Chapter 1 | **Introduction**

It has become an undeniable fact that mobile technology has become an imperative need for all cultural and social organisations to utilise the new opportunities available today. Today a vast number of mobile applications are serving millions of customers and benefiting businesses every minute as we speak. The growth of mobile technology in the recent years has been quite dramatic and exciting, whilst opening up doors for the technology experts and businesses to present themselves in front of their audiences on a truly global scale. It does not matter how big or small an organisation may be, but the opportunities are equally available for the creative and entrepreneurial minds to capitalise on. In this case, the project aims to stretch the scope of business and education that a Museum has to offer beyond its physical walls.

This report presents a case of an interesting application development process for *Timespan* and discusses the technical and logical challenges involved along the way. This project has been an offshoot and extension of the initial iOS application which is available to download from Apple App Store. The project has been already delivered to Timespan in association with the *University of St Andrews*.

1.1 Background

The purpose of this App is to create a virtual tour of The Highland Clearances and to create an interactive trip for the actual visitor in the Highlands area of Strath of Kildonan, a northernmost part of Scotland[1]. Timespan is launching this application because the year 2013 marks the Bicentenary of Highland Clearances. The current iOS application developed by Bluemungus[2] is available at the App Store. Timespan wants to port this application to other mobile platforms to extend their use base.

1.2 Scope and Objectives

One of the core requirements in the existing Timespan iOS application is that some features require the user to be physically present in the Strath of Kildonan. Due to the fact that there is no mobile reception in that area, this project aims to develop a mobile application which should work offline without any internet or mobile access. All of the data needs to be present on the device at the time of installation. The application database consists of audio and visual information such as pictures and videos. There are also historical maps to give the visitor information about different events at different locations

Following are features or different screens present in the iOS application:

- Clearances Story Book
- Kildonan Trail Offline Map
- Oral Histories Audio Podcasts
- Longhouse and About static screens

Following are the main objectives of this project:

- To perform the evaluation of iOS Application
- To determine cross platform mobile development methodologies
- To perform the market analysis for different platforms
- To generate the application content
- To support multiple device screens
- To develop and publish the application for other mobile platforms.

1.3 Achievements

An Android application has been successfully developed to be published on the Android Market Google Play Store. It meets all the requirements specified by this project. The client has tested and given the final feedback about the application which can be found in Appendix C. The project accomplishes the following objectives:

- Generated all the content for the application.
- Implemented landscape and portrait mobile screen support.
- Completed the Clearances Story, Kildonan Trail, Oral Histories, Longhouse and About screens.
- The user can perform multiple operations, for example playing audio and navigating the app.
- The application is ready to be published on the Google Play Store.

1.4 Overview of Dissertation

There are total nine chapters in this report. The first chapter is the *Introduction* which gives an overview, background and achievements of this project. The second chapter *Context Survey* discusses related work review, evaluates different mobile development methods, presents market share analysis of smartphones and at the end, points out few Android platform specific issues. The third chapter *Evaluation of the iOS App* evaluates the existing iOS application. Based on the results from second and third chapters, the fourth chapter *Software Requirement Specifications* lists down the application requirements. Chapter five *Software Development Methodology* discusses the design and development process adapted for this project. Chapter six *Android Application Design* and chapter seven *Android Software Development* describes the user interface design, software design and software implementation of this project. Chapter eight *Publishing* explains how to publish the application on Google Play Store. Finally chapter nine *Conclusion* wraps up this project and gives an overview of future work.

Chapter 2 | **Context Survey**

This chapter discusses the literature review of this project. In the first part of this chapter, a review is presented about related mobile apps developed by museums around the world. These apps are available on iOS and Android markets. In the second part of this chapter, a market share analysis is presented to determine the market statistics on different mobile platforms. This project uses public data to present the market analysis. The mobile platform with the highest market share is considered for the development of the mobile application, which is Android for this project.

The third part of this chapter discusses issues and resource optimization for the Android platform. The following issues will be considered; security, performance, usability and battery utilization.

2.1 Related Work

There are several museum applications available on the Android, Apple and other smartphone markets. Following are the reviews of three applications selected from the Apple and Android market stores.

- **The Louvre Museum, Paris**

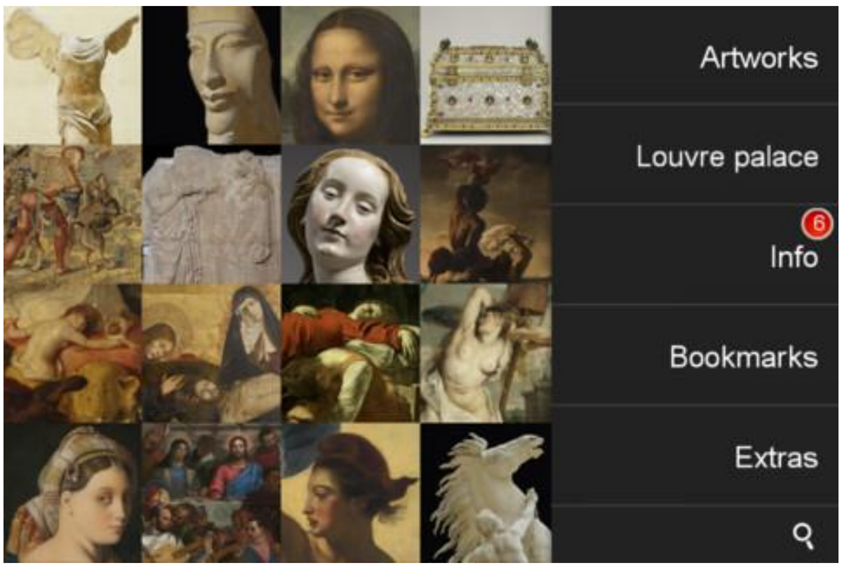


Figure 1 - The Louvre Museum, Paris iOS App[1]

The *Musée du Louvre* iPhone application offers a fairly concise view of the Museum. The virtual tour includes some handy information about the 12th century palace in the form of pictures, video and text. They have utilised some of the popular features of the iOS for example the artwork is presented in iPhone Cover Flow format. The app offers viewing options to see all the paintings as you flip through the collection and individual items can be tapped on to see enlarged views and read more details. The app

[1] Image Credit and iOS App Available at: https://itunes.apple.com/gb/app/musee-du-louvre/id337339103

supports iPhone's pinch gestures to zoom-in to closely look at parts of the museum paintings. Under a location button, it also lets you see the exact location of where the painting resides in the Museum. You can find some detailed and high definition videos about some of the most significant pieces. The videos are narrated in a choice of French, English, German and Japanese languages. It also lets you discover the various parts of the Louvre in a Cover Flow format and some detailed information can be seen under the images. Overall the application is pretty simple and straight forward with a conventional set of features.

➢ **National Portrait Gallery, London**

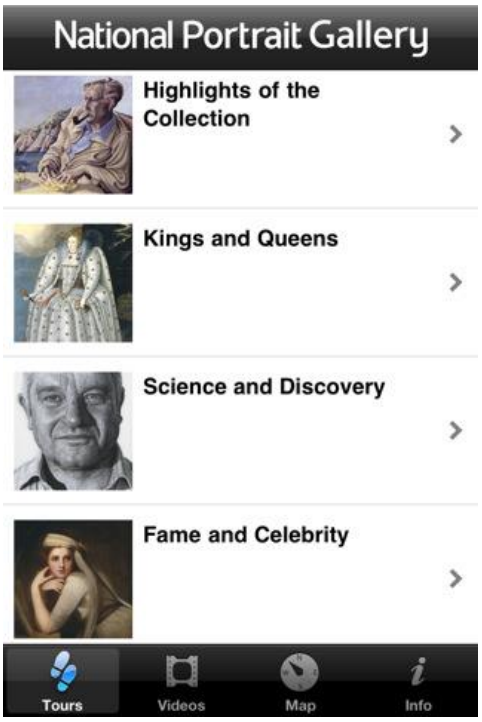


Figure 2 - National Portrait Gallery iOS App[2]

The *National Portrait Gallery* iOS application is very rich in content as it showcases comprehensive galleries and audio commentaries from the curators who dive in to the background and history of every art piece. It also lets you see the several themed sequences of the collections by selecting from Kings, Queens, Science and Discovery to Fame, Celebrity, and Writers. Similar to the Louvre app it also shows you the exact physical location of every art piece in the Museum. The total number of portraits are said to be over 10,000 art pieces. Other features include useful information about opening hours and the shops and facilities around the area.

However, a slight weakness we found in this app is the audio commentaries can be annoying for some as the content is only available in audio format. If a visitor is not carrying their headphones then they are left with two options - either to avoid the app altogether or turn on their speakers loud and disturb

[2]Image Credit and iOS App Available at: https://itunes.apple.com/gb/app/national-portrait-gallery/id411447181

other visitors around them. Another weakness is the sheer size of the app which can take quite a while to download.

➢ **Vusiem for British Museum**

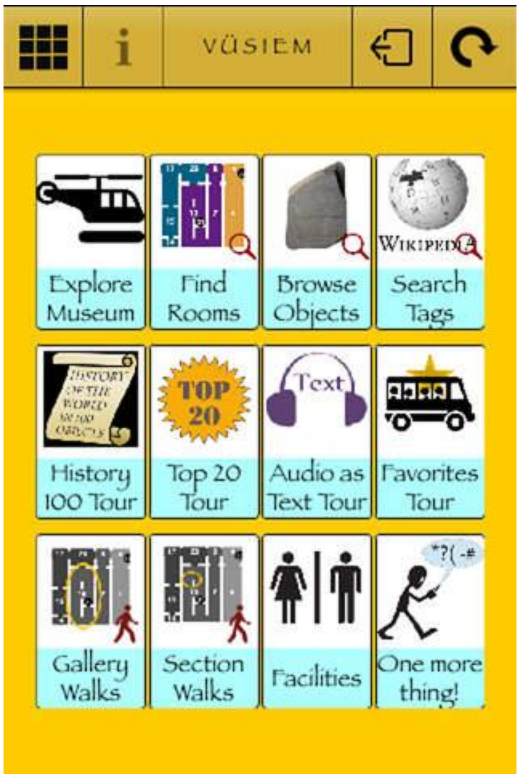


Figure 3 - Vusiem for British Museum iOS& Android App[3]

The *Vusiem* is a mobile app for the *British Museum* which showcases more than 800 important objects that can be seen across the museums in London. It offers a virtual tour along with audio guides that let you explore all the objects in detail. However it does not provide a slideshow view to see flick through the pieces quickly, and similar to the National Portrait Gallery app; it lets you explore the gallery in themed collections. We noticed a major chunk of information is referenced through Wikipedia but we hope there must be some sort of editorial control over; what gets published etc. It also supports seven different languages including English, German, French, Spanish, Russian, Portuguese and Simplified Chinese. The usability is pretty good and it does create a spark of interest when you see a gallery full of curious little objects responsible for shaping the world we live in. Overall it offers all the basic functionalities in a conventional way.

[3] Image Credit and iOS App Available at: https://itunes.apple.com/gb/app/vusiem-for-british-museum/id551275212

Android App: https://play.google.com/store/apps/details?id=air.com.bm.london.vusiem&hl=en_GB

2.2 Evaluation of development methodologies

The related work review provides information about the different applications in the market. To implement the mobile application, there are several development styles and methodologies available.

➢ **Architecture Styles**

The architecture defines the behaviour of the application. Following are the two basic architecture styles in which a mobile application can be developed.

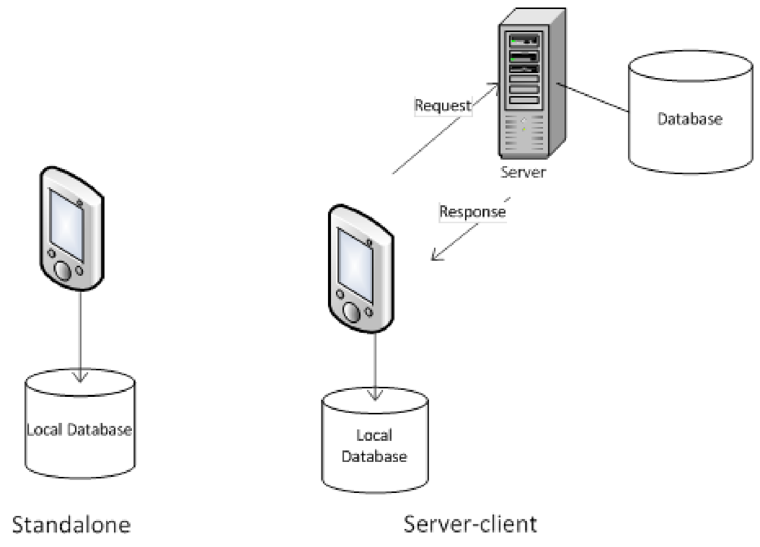


Figure 4 - Standalone and Server-client architecture

2.2..1 Standalone

Standalone or offline applications do not interact with the server after installation[3]. The left part of the Figure 4 presents the standalone architecture. The current iOS application for Timespan follows *Standalone* architecture. The application size is increased because everything needs to be downloaded which may or may not be required by the user. The mobile client performs all of the processing and it does not depend on server calls so the application can work without internet availability. Any update in the application requires the user to download the complete application again.

2.2..2 Server-Client

A server-client architecture implements business logic on the server and user-interaction and information display is performed on the client[4]. The right part of Figure 4 presents server-client architecture. These are data driven applications which have a tiny installation size and increases on request of the user. Internet connectivity is required for web service calls[5] and updates, otherwise only the client-side features are accessible.

➢ Cross Platform Methodologies

The architecture style is not enough to decide development architecture of a mobile application. The next challenge is to determine the development approach. The project requirement of this application is to create a secure[6] cross platform application which uses touchscreen, GPS and utilize audio capabilities. This can be achieved by using the following approaches [3][7]:

2.2..1 Native

Native application is built only for the specific platform and uses operating system features to enhance performance. The market place for every operating system requires a paid developer account and the application is only distributed via the market place. These applications are very reliable and powerful. The downside of this approach is to develop native application for each platform separately.[8].

2.2..2 Hybrid

Hybrid applications work similar to native application but developed using a common language. All of the available hybrid platforms are proprietary and the application is distributed via respective market place of each operating system. They can access all the hardware capabilities of a smartphone - for example camera, audio, video and GPS sensor[9].

2.2..3 Web

A mobile application runs in the browser and does not require a market place to distribute the application. These applications are developed using HTML, CSS and JavaScript. HTML5 can access hardware capabilities of smartphones.[3]

Table 1compares the three development approaches with frameworks, programming language and supported mobile platforms.

Approach	Framework	Language	Platforms Supported
Native	iOS SDK[10], Android SDK[11], BlackBerry SDK[12], Microsoft Silverlight[13]	Objective-C, Java, C#	iOS, Android, BlackBerry, Windows Phone
Hybrid	PhoneGap[14], Titanium[3]	JavaScript	iOS, Android, BlackBerry
Web	Web Technologies	HTML 5[15], CSS3, JavaScript	HTML5/CSS3/JavaScript supported browser

Table 1 - Mobile Development Approaches

There are other approaches available such as *Cabana*[16], *Mobile Mashups*[17], *Unity*[18] and *Cross-Compiling Methods*[19]. The selection for the development approach should be based on Time, Available Resources and Features. Selecting the Web approach has certain challenges such as handling the behaviour of mobile on different devices (including Personal Computers) [20]. Responsive designs are used in Web approach to solve this problem[21].

Apart from determining the architecture style and cross platform methodologies, the development of an application also highly depends on the market share of specific platforms. An application is only useful if it works on the user's mobile device. The next part presents a market share analysis to determine the number of users for each mobile platform.

2.3 Market Share Analysis

Figure 5- Nokia, Samsung, iPhone and BlackBerry Smartphones[4]

This part performs an analysis on the market share of smartphones. Every modern smartphone runs on an operating system with high computing capability. These smartphones feature multimedia audio, video, camera, touchsreen, Wi-Fi and GPS sensors[22]. Examples of operating systems which runs on these smartphones include Apple iOS[8], Google Android[23], BlackBerry OS[24] and Windows Phone[25]. These operating systems are equipped with browsers capable of displaying standard web pages. Application market place for each platform allows developers to build and distribute applications.

To determine the market share, public survey and research data is used for analysis. There are several ways to do the market analysis such as different vendors, device types, device models and operating systems.

[4]Image Credit: http://www.tecnologianice.com/2011/11/android-vs-iphone-vs-windows-phone.html

According to IDC (*International Data Corporation*) report for 2012 smartphone OS, Android has 69% and iOS has 19% market share[26].Figure 6 shows the distribution of smartphone OS in 2012.

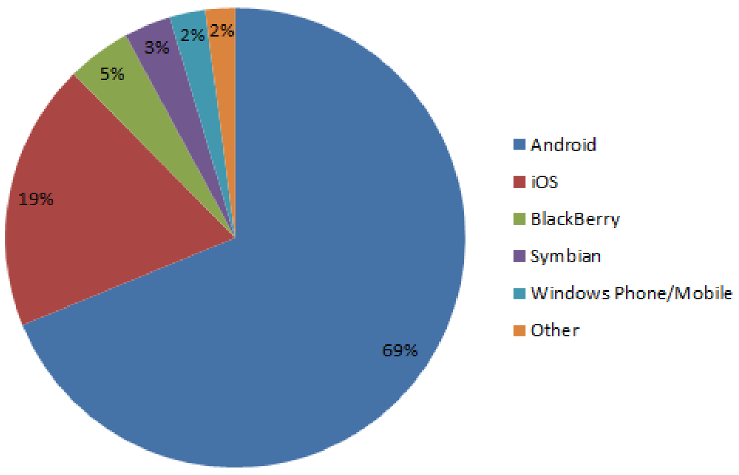


Figure 6 - IDC 2012 Smartphone OS Market Share[26]

According to *ComScore*, while Android remains the top smartphone OS, Apple stays as the top selling smartphone vendor in the UK[27] with more than 31% market share. Figure 7 shows the UK market share for different vendors.

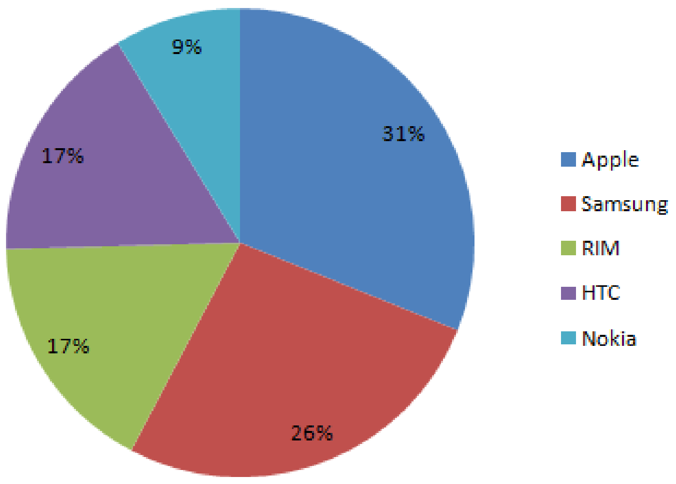


Figure 7 - ComScore 2012 Smartphone Vendor Market Share[27]

The data for tablet OS shows that Apple is at the top position with more than 54% of the market share and Android market share of 43%[28].Figure 8 shows the IDC 2012 Tablet OS market share

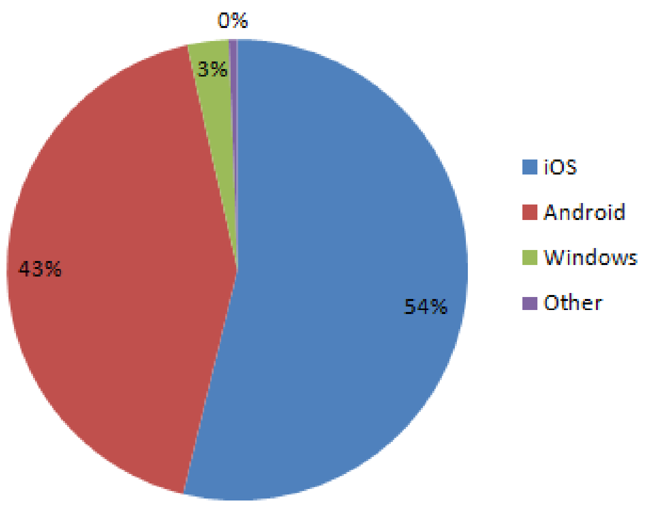


Figure 8 - IDC 2012 Tablet OS Market Share[28]

The results from the market share analysis shows that Android has the major share compared to Apple iOS. With different vendors and versions of operating system, Android has captured a major share of the smartphone market. Apple takes the second position in terms of the market share. This cannot be neglected as Apple iOS only works on iPhone and iPad. This makes Apple a major competitor in the smartphone and tablet market.

In the next part, issues are discussed related to Android platform. This platform is selected for study on the basis of market share analysis.

2.4 Issues in Android

Android smartphones are shipped by several vendors such as Samsung, HTC and Motorola. According to Google CEO, Eric Schmidt, there are 1.5 million Android devices activated each day[29]. Google has been struggling to support each and every hardware device for its Android operating system. This creates a major compatibility issue for the developers as they have to write application to support hundreds of different devices which increases the development and testing cost as compared to iOS.

Being an open source development platform, Android has major significant advantages over iOS. There are practically no restrictions on developing any feature. Apple has a restricted development environment with specific guidelines for building Apps. The Android market place also has no limitations and restrictions. Because of this openness, certain problems have been occurring for the Android users and developers.

➢ Compatibility

There are several smartphone vendors with different devices supporting Android OS. Google also release frequent updates to Android OS. Due to different vendor and the OS, most of the devices fail to update to latest OS. The latest OS is Android version 4 but majority of the devices are still on 2.3 Gingerbread[30]. Table 2and Figure 9showAndroid OS distribution.

OS Version	OS Name	Distribution
1.6	Donut	0.10%
2.1	Eclair	1.50%
2.2	Froyo	3.20%
2.3	Gingerbread	36.60%
4.0	Ice Cream Sandwich	25.60%
4.1 / 4.2	Jelly Bean	33.00%

Table 2 - Android OS Distribution

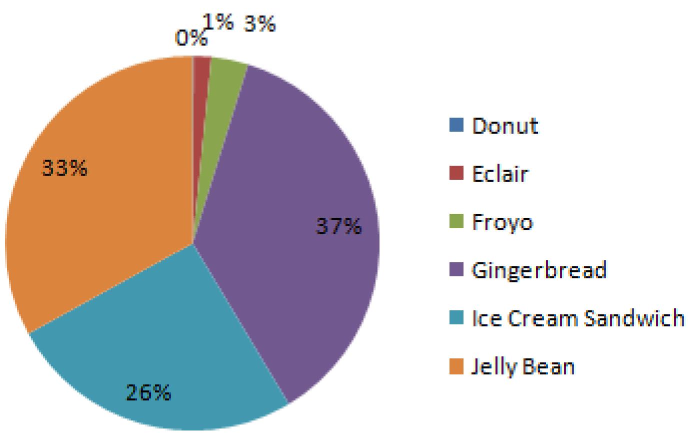


Figure 9 - Android OS Distribution

➢ **Security and Privacy**

Android applications can be downloaded from the market place as well as from any private store. This results in poor check and balance on the quality of available Apps. There have been several malware Apps submitted in the store[31]. Although anti-virus applications are available and Google removes these malware applications time to time. The number of malware attack on Android is much higher as compared to iOS[31].

➢ **Power Utilization**

The hardware capabilities of smartphones have increased exponentially in the recent years. Multimedia features and high performance gaming requires extensive power source. Unfortunately the power back-up (battery) has not made significant advancement. According to survey, users are not satisfied with battery utilization of Android smatphones [32]. There have been various studies on how to optimize battery utilization[33] including certain programming techniques for the developers[34].

2.5 Conclusion

This chapter focused on reviewing related museum applications, evaluating the development approach-es for mobile applications, performed a market analysis and pointed out the major issues in Android platform. The next chapter evaluates the existing Timespan iOS application to gather the requirements.

Chapter 3 | **Evaluation of the iOS App**

This chapter focuses about evaluating a mobile apparition. There are several methods available to determine the *Usability* of a mobile application. For this project we are using *PACMAD* model, which evaluates usability on the basis of *Effectiveness*, *Efficiency* and *Satisfaction*[35]. The process involves manually exploring the application; therefore the existing iOS application is evaluated for each feature using this model. The App has been designed following iOS design guidelines[36] except few minor problems identified.

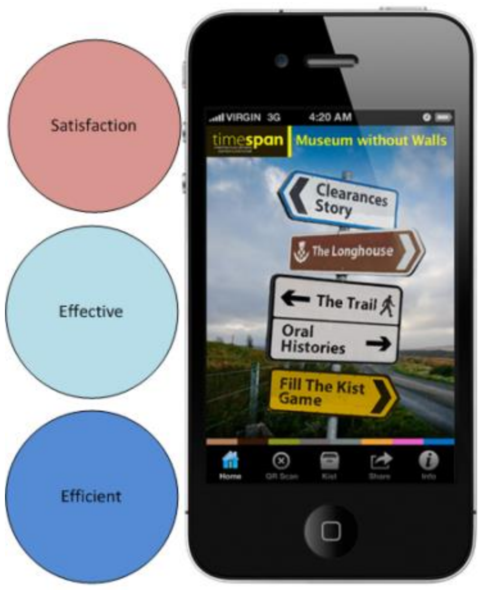


Figure 10–Mobile Usability Factors

Following are the features in the iOS application:

The '*Kildonan Trail'* is a journey which takes the user to ten locations along the Strath of Kildonan. The user learns about the geographical landscape and how it has changed over time since the clearances. Audio narrations and historical picture gallery is available for the user. The sites are visible on the trail map. The '*Clearances Story'* follows the Macpherson family and how they faced eviction from their home and emigration to North America. This story is based on the letters written two hundred years ago by the people who were forcefully evacuated. Audio and images are available for the user to browse through the story. '*The Longhouse*' tells the user about the Macpherson's house and different features of houses in the highlands. The interpretation and construction of the houses is derived from the drawings and archaeological research techniques. The '*Oral Histories'* contains voices of the descendants, emotional poems and traditional music. '*Fill the Kist*' is a game for youngsters which use QR Codes to collect items in the Kist. The App can also be used for educating people about Clearances as it has now become an integral part of Scottish Studies.

➢ **Home**

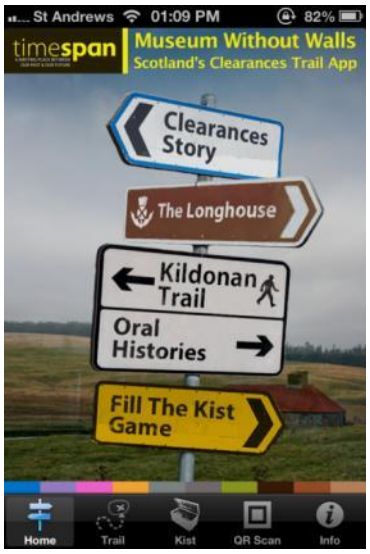
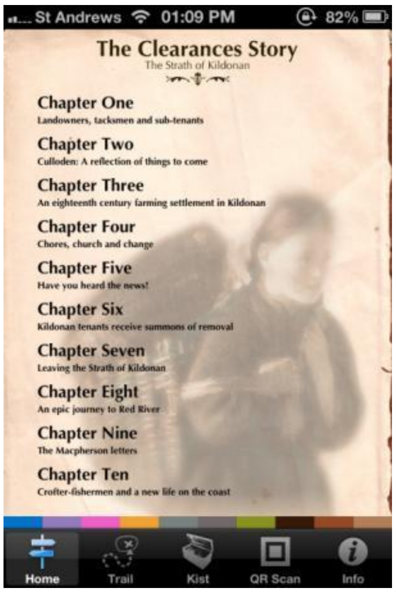


Figure 11 - Timespan iOS App Home Screen

The Home screen consists of horizontal and vertical menus as shown in Figure 11. The horizontal menu at the bottom is the native iOS Tab-Bar controller. User can navigate to other features from this screen using the menu buttons. The background image keeps changing with a specified interval.

➢ **Clearances Story**

This section contains a book which contains text, images and audio. There are ten chapters in the book, each chapter containing several pages. The user can go back and forth by swiping the pages.

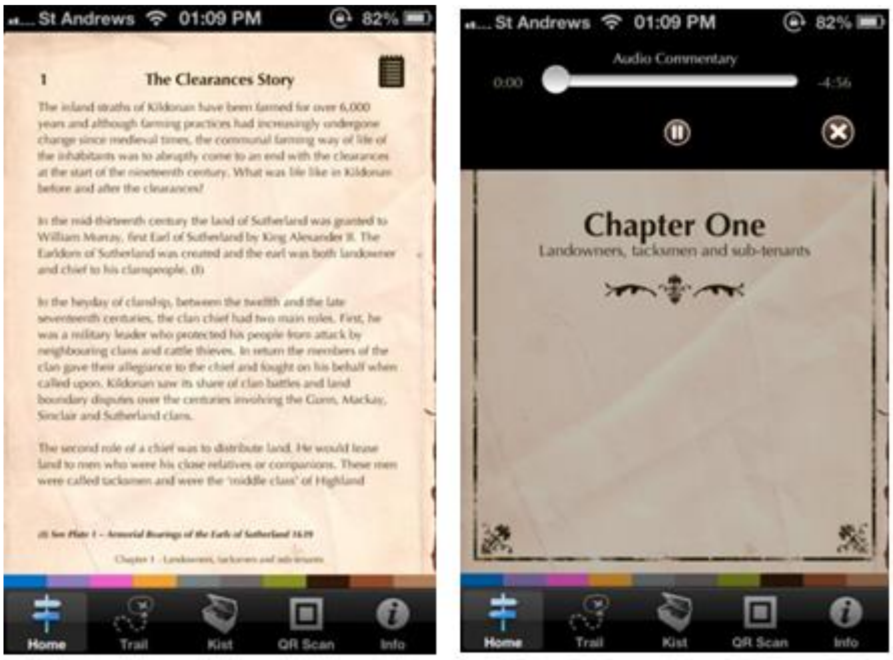


The Clearances Story
The Strath of Kildonan

Chapter One
Landowners, tacksmen and sub-tenants
Chapter Two
Culloden: A reflection of things to come
Chapter Three
An eighteenth century farming settlement in Kildonan
Chapter Four
Chores, church and change
Chapter Five
Have you heard the news!
Chapter Six
Kildonan tenants receive summons of removal
Chapter Seven
Leaving the Strath of Kildonan
Chapter Eight
An epic journey to Red River
Chapter Nine
The Macpherson letters
Chapter Ten
Crofter-fishermen and a new life on the coast

Figure 12 - Timespan iOS App Clearances Story - Chapters List Screen

Figure 12 shows the first screen which is the table of contents from which user can navigate to different chapters in the book. This provides the user to easily navigate within the book.


1 The Clearances Story

The inland straths of Kildonan have been farmed for over 6,000 years and although farming practices had increasingly undergone change since medieval times, the communal farming way of life of the inhabitants was to abruptly come to an end with the clearances at the start of the nineteenth century. What was life like in Kildonan before and after the clearances?

In the mid-thirteenth century the land of Sutherland was granted to William Murray, first Earl of Sutherland by King Alexander II. The Earldom of Sutherland was created and the earl was both landowner and chief to his clanspeople. (I)

In the heyday of clanship, between the twelfth and the late seventeenth centuries, the clan chief had two main roles. First, he was a military leader who protected his people from attack by neighbouring clans and cattle thieves. In return the members of the clan gave their allegiance to the chief and fought on his behalf when called upon. Kildonan saw its share of clan battles and land boundary disputes over the centuries involving the Gunn, Mackay, Sinclair and Sutherland clans.

The second role of a chief was to distribute land. He would lease land to men who were his close relatives or companions. These men were called tacksmen and were the 'middle class' of Highland

(I) See Plate 1 – Armorial Bearings of the Earls of Sutherland 1619

Chapter 1 - Landowners, tacksmen and sub-tenants

Figure 13 - Timespan iOS App Clearances Story

After selecting a chapter, the application displays the screen as shown in left part of Figure 13 . There are certain problems on this screen. The user cannot perform zoom actions to read the text properly. The second problem is that when the audio is being played as shown in right part of Figure 13, user cannot interact with the App. Sometimes the application crashes when user switches back and forth. The book used in the iOS application is not in a reusable format; for example *PDF*. This makes it impossible to reuse the content for further application development.

- **The Longhouse**

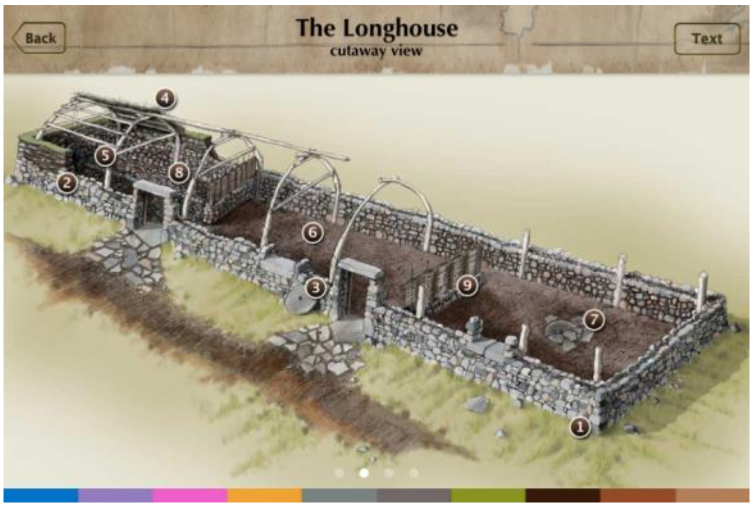


Figure 14 -Timespan iOS App The Longhouse

This feature contains an image of a Longhouse which shows additional information by tapping on the labelled numbers as shown in Figure 14. This feature will not be developed as required by the client.

➢ **Kildonan Trail**

This section contains an historical map of the Strath of Kildonan which works offline without internet connectivity. The user can tap on map pins which leads to further detail views as shown in Figure 15. The user can also scroll and zoom on this map.

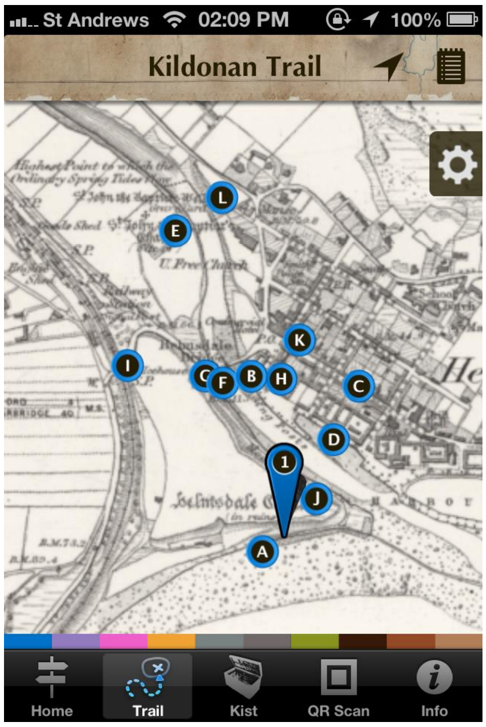


Figure 15 - Timespan iOS App Kildonan Trail

3.1 Conclusion

In this chapter we evaluated the Timespan iOS application. The following list shows the issues found during evaluation:

- Clearances Story book does not have the zoom feature.
- The book is not in a reusable format.
- The audio playing makes the application unusable.
- The application does not support landscape orientation.
- The application crashes on certain scenarios.

In the next chapter, this report discusses the software requirements for this project. The requirements are gathered after meetings with the client and the findings from evaluation of iOS application.

Chapter 4 | **Software Requirement Specifications**

After discussing with client about the market share analysis and different cross platform development methodologies, this project focused on developing the application for Android smartphone. This chapter defines the set of requirements to build the mobile application for Android platform.

4.1 Functional Requirements

"*Functional requirements are statements of the services that the system should provide*" [37]

The functional requirements are divided into three parts on the basis on priority as primary, secondary and tertiary requirements.

➢ **Primary**

a) All of the unavailable content (Map, Book and Images) should be generated for this application.

b) The application should support Android smart phones.

c) The application should support landscape and portrait orientation.

d) The application should display historical Geo-referenced maps of the Strath of Kildonan.

e) The application should use the user geo location and display it on the Geo-referenced Map.

f) The application should allow zoom and panning feature on the map.

g) The application should allow pins on the map.

h) The application should allow app usage (multi-threading) during audio play.

➢ **Secondary**

a) The application should provide PDF book reading with text zoom support.

b) The application should allow text zoom.

c) The application should display photo galleries.

d) The application should play audio podcasts.

➢ **Tertiary**

a) The application should support full screen on photo galleries.

b) The application should be published on the Google Play Store.

4.2 Non Functional Requirements

According to the definition: *"Non-functional requirements are constraints on the services and functions offered by the system"*[37]

The non-functional requirements are list down below:

a) The application should follow Android User Interface standards to keep the App usability simple and easy for the users

b) The application should ask the user to use geo location data and allow the user to change privacy settings at any time. The geo location data should not be stored within the app and destroyed after the usage.

c) The application should perform smoothly and keep the user interface separate from database calls or processing thread.

d) The application should display correct and consistent information with respect to the option selected by the user. The records in database should be unique and identified by primary keys.

The application should handle errors in a user friendly way so the user is aware of any memory or hardware errors. The application should not crash unexpectedly.

4.3 Conclusion

This chapter presented a set of requirements necessary to develop this mobile application. The requirements are divided into three categories based on the priority as primary, secondary and tertiary. Design and development phases of this project follow the requirements as described in each category.

The next chapter discusses the software development methodology used in this project.

Chapter 5 | **Software Development Methodology**

Mobile applications are developed in a rapid application development environment. The changes are frequent and requirements keep changing on the basis of feedback from users/client. The application needs to be tested on multiple devices which are running different versions of the operating system. This creates a need for a software development methodology which can handle change and provide a way to meet the rapid development requirements.

This chapter discusses the agile methodology in terms of mobile application development. At the end of this chapter, it discusses the source code management system used for this project.

5.1 Agile Development

Agile development is a software development methodology that suits the requirements of this project. According to the Agile Manifesto, responding to change is more important than following a plan[5].

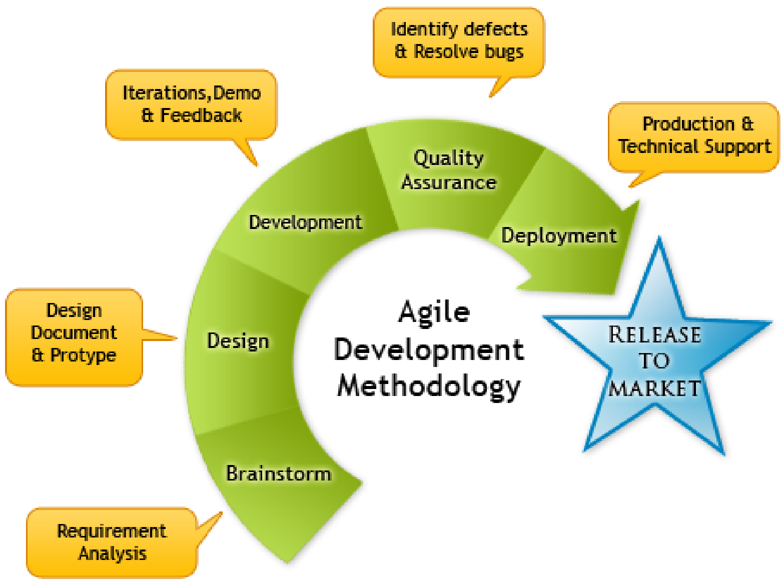

Identify defects & Resolve bugs
Iterations,Demo & Feedback
Production & Technical Support
Quality Assurance
Development
Deployment
Design Document & Protype
Design
Agile Development Methodology
RELEASE TO MARKET
Brainstorm
Requirement Analysis

Figure 16 - Agile Development Phases[6]

Agile development has several phases as shown in Figure 16 which makes it suitable for rapid application development. Each process follows several iterations. The following phases which are used in this project are discussed below.

[5]http://agilemanifesto.org/
[6]Image Credit: http://agileenterprises.com/

➢ **Design**

In this phase, the application architecture is designed according to the requirements. Several iterations are performed before moving to the development phase. The final design is verified and approved by all parties involved in the project i.e client and development team.

➢ **Development**

The development phase kick starts the implementation of the application. From setting up environment to deciding of development tools, all of the decisions are made in this phase. The application is developed in iterative cycles depending on the feedback from further phases.

➢ **Deployment and Release to Market**

The application is deployed on mobile devices and tested by the developer and client. After several iterations and successful feedback from client, the application is released to the market.

5.2 Source Code Management

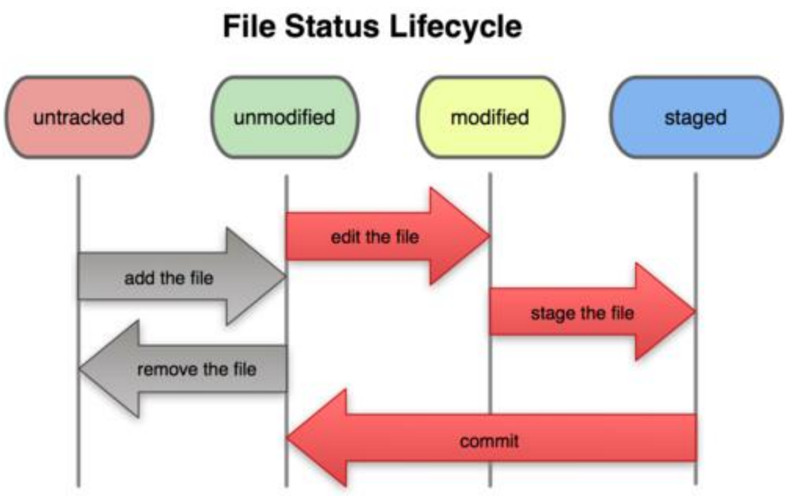


Figure 17 - lifecycle of a file in Git SCM[7]

A source code repository provides management and tracking of changes in the application code. Figure 17 shows the lifecycle of a file inside code repository. Using an online repository provides a safe place to store your code. There are several source code management tools and online providers. This project is using *UnFuddle*[8] online repository which provides free space for Git source management system.

5.3 Conclusion

This chapter discussed about the different phases involved in this project and how mobile applications are developed following a sequence of iterations.

The next chapter defines the application design which is the software and user interface design.

[7] Image Credit: http://git-scm.com/book/en/Git-Basics-Recording-Changes-to-the-Repository
[8] https://unfuddle.com/

Chapter 6 | **Android Application Design**

This chapter discusses the software and user interface design of the application. By designing the software architecture for the project, it produces a re-useable and commonly understandable application. The application can be verified at the end of implementation. Software design also provides the initial documentation of application[38].

Mobile applications are highly dependent on a friendly user interface. The application is only useful for the user if it's easy to use. For creating a user friendly interface this project follows the Android User Interface guidelines[9].

6.1 Model View Controller Design Pattern (MVC)

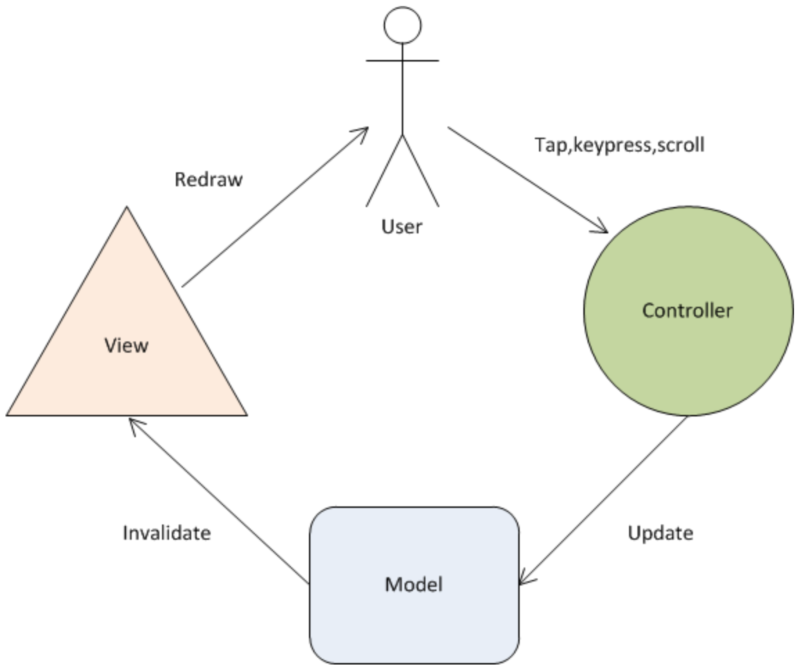


Figure 18 - Android MVC[10]

By default Android projects follow *Model-View-Controller* (MVC) design pattern as shown in Figure **18**. The *Model* is responsible for implementing the business logic of the application. It contains the code which is required by different methods of the application such as database queries, web calls, scientific calculations and other data interaction methods. Every application has data interaction tasks and Model connects the other two components within the application providing a channel between Controller (handles user input) and View (rendering user interface).

[9]http://developer.android.com/guide/topics/ui/index.html
[10]http://www.cs.ccsu.edu/~stan/classes/CS355/notes/03-AndroidUI.html

The *View* is responsible for presenting a visualisation of the Model such as rendering the user interface and sending audio to be played on speakers. Android provides this View component as a tree of sub-class of *View* class. Anything which is displayed on the Android user interface is inherited by this View class.

The *Controller* is responsible for handling the external actions to the application, such as keystroke, touchscreen tap or incoming call etc. It is implemented by the Android OS and maintains a queue to execute each action. The OS removes each action from the queue in the order it was received and executed. On execution of the action, the Model may perform some changes in the data and View updates the user interface accordingly.

The benefit of using design patterns such as MVC in the application allows reusability and easily understanding of the source code. In Android applications, using Views across different applications is a common practice and there are many open source View components available. This project also uses open source Views. The Model is least reused in software applications due to the fact that program logic differs from application to application. Controller is implemented by the Android OS, so it is reused by default in every Android application.

6.2 Software Architecture

This project follows the standalone application layered architecture to keep the separation between different components of the system. Layered architecture keeps the design simple and clean. Figure 19 shows the layered architecture of this project.

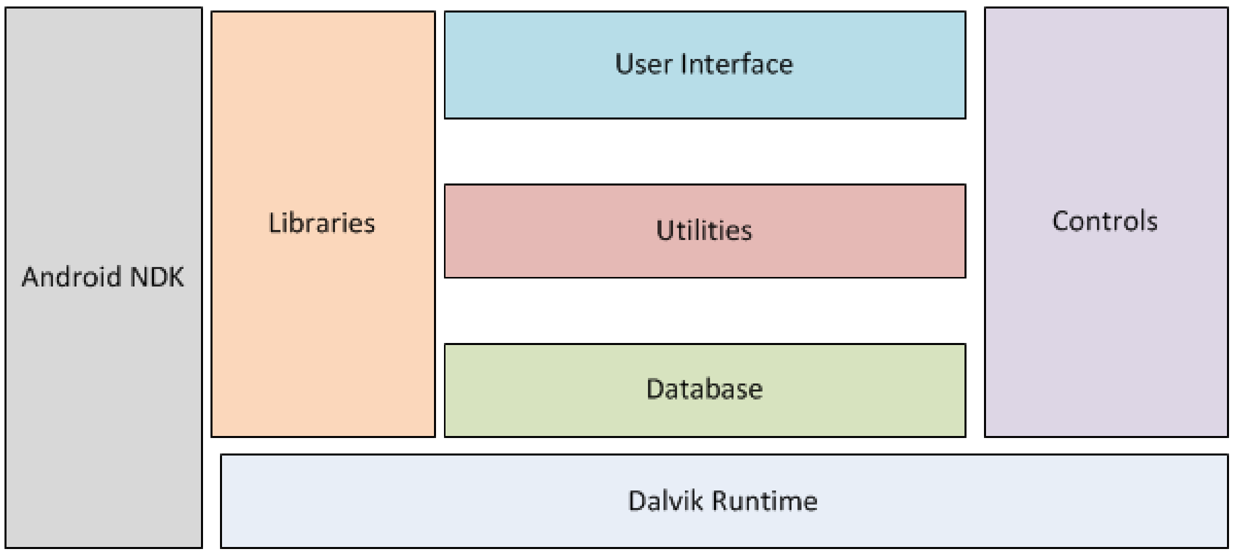


Figure 19 - Layered Architecture of Timespan App

User Interface layer handles the input events in the form of external interactions with the App. These inputs are provided by the user in form of type, tap or scroll etc. *Utilities* layer provides helper functions to the application such as data parsing, data verification and file read/write. These utilities are used by the *Libraries* and *Controls* layer. *Libraries* layer contains the third party libraries used by the application. This project uses only open source third party libraries. *Android NDK* layer contains the C++ runtime for Android which is used by one of the library in this project. It is used to compile the libraries which are written using C++ language. *Controls* layer provides widgets and data controls used by User Interface layer. These widgets are custom controls developed for this project. *Database* layer handles all the database related tasks such as fetching and storing data in the database. *Dalvik Runtime* is the Android SDK providing the standard Android libraries.

The benefit of using a layered architecture is separating the responsibility of different component within the system. The layers are able to reuse the functionality among them and application layers are distributed over physical layers. This provides performance improvement, scalability and error management. The application is more testable by standard methods and future application extensions are easier to develop.

6.3 User Interface Design

Android graphical user interface is usually composed of XML files called *layout*. Before designing the user interface files, the application screens are designed using static images generally known as *Mockups*. These mockups are then transformed into interactive layouts. The mockups for this application are used from the iOS application so this part discusses about the Android layout design in XML files. The user interface logic code is written in the Android *Activity* file which is the basic unit of an Android application. An Activity is used to display the Android application user interface which contains widget such as buttons, labels and text boxes etc. Usually the user interface is defined using the XML file rather than writing in the activity file. At the time of compilation, the XML is loaded by calling *setContentView*() method in the activity class which implements the *onCreate*() event handler as shown in Figure 20:

```
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.main);
}
```

Figure 20 - Initialization of an Android Layout User Interface

➢ Views and ViewGroups

View class is the Android's most basic component from which user interfaces can be created. It is similar to the *Swing JComponent* class for Java application. A View derives from the base class *android.view.View*. Any visible rectangular area on an Android application is defined by a View. Usually an Android *Activity* contains Views and ViewGroups. A View is implemented as a *widget* that has an appearance on screen. Widgets are used to create interactive UI components. Examples of widgets are buttons, labels, text boxes, etc. Views can be grouped together into a *ViewGroup*. Several Views and ViewGroups can be grouped together in to create a *layout*. Layouts are invisible containers used for holding other Views and nested layouts in a sequence order. Examples of ViewGroup Layout are LinearLayout, FrameLayout, AbsoluteLayout, TableLayout and RelativeLayout. LinearLayout and RelativeLayout are the most common Layouts.

Figure 21 shows the View hierarchy of a ViewGroup in Android Layout.

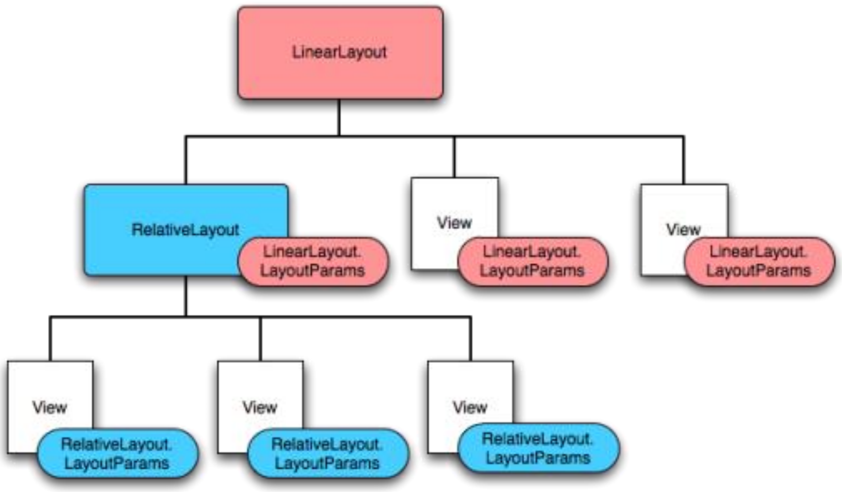
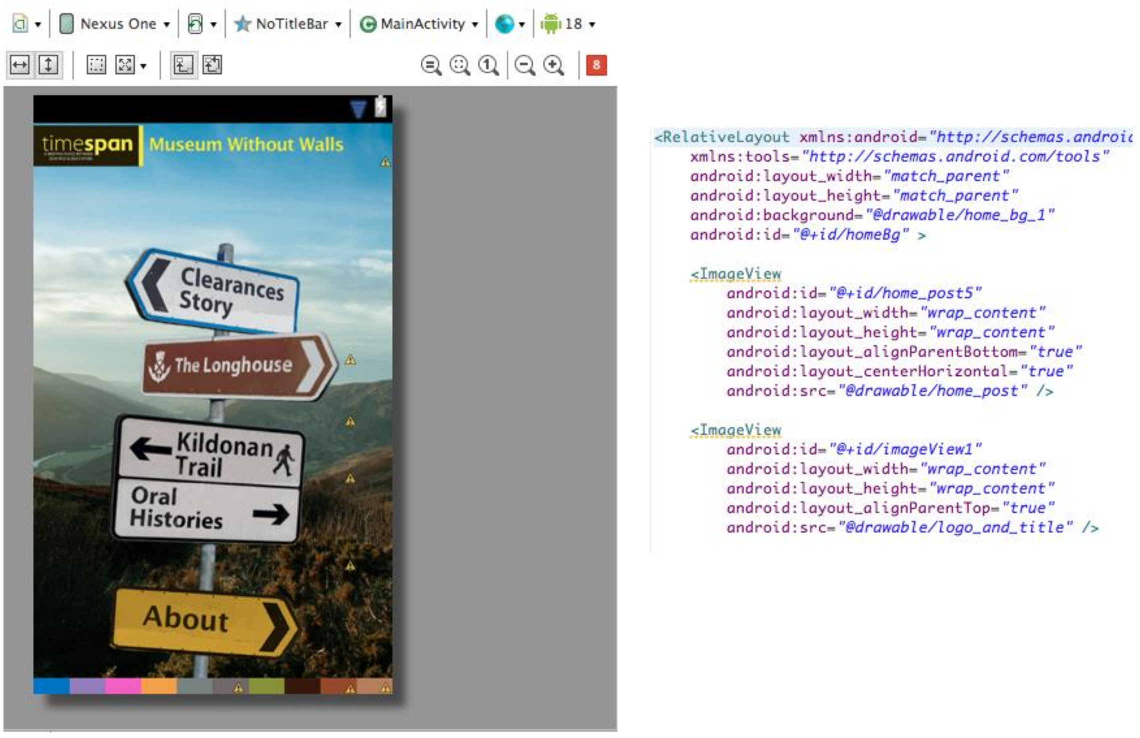

LinearLayout
RelativeLayout
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
LinearLayout.LayoutParams
View
RelativeLayout.LayoutParams
View
RelativeLayout.LayoutParams
View
RelativeLayout.LayoutParams

Figure 21 - Android Layout Hierarchy

Android layouts can be edited using Eclipse which provides both code editor and the graphical editor. Figure 22 shows the Eclipse Android layout editor.



```
<RelativeLayout xmlns:android="http://schemas.androic
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/home_bg_1"
    android:id="@+id/homeBg" >

    <ImageView
        android:id="@+id/home_post5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:src="@drawable/home_post" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:src="@drawable/logo_and_title" />
```

Figure 22 - Android Graphical UI (Left) and XML Layout (Right)

6.4 User Interface Widgets

Android application user interface is composed of different elements known as *widgets*. These widgets provide functionality to display different types of data. Most of the basic data widgets such as text and image are included in the SDK. Developers can also create custom widgets following the guidelines. There are also third party open source widgets available. This project uses both basic and custom widgets. Every screen in this project is composed of multiple widgets. Figure 23 shows some of the widgets used on the main screen of the application

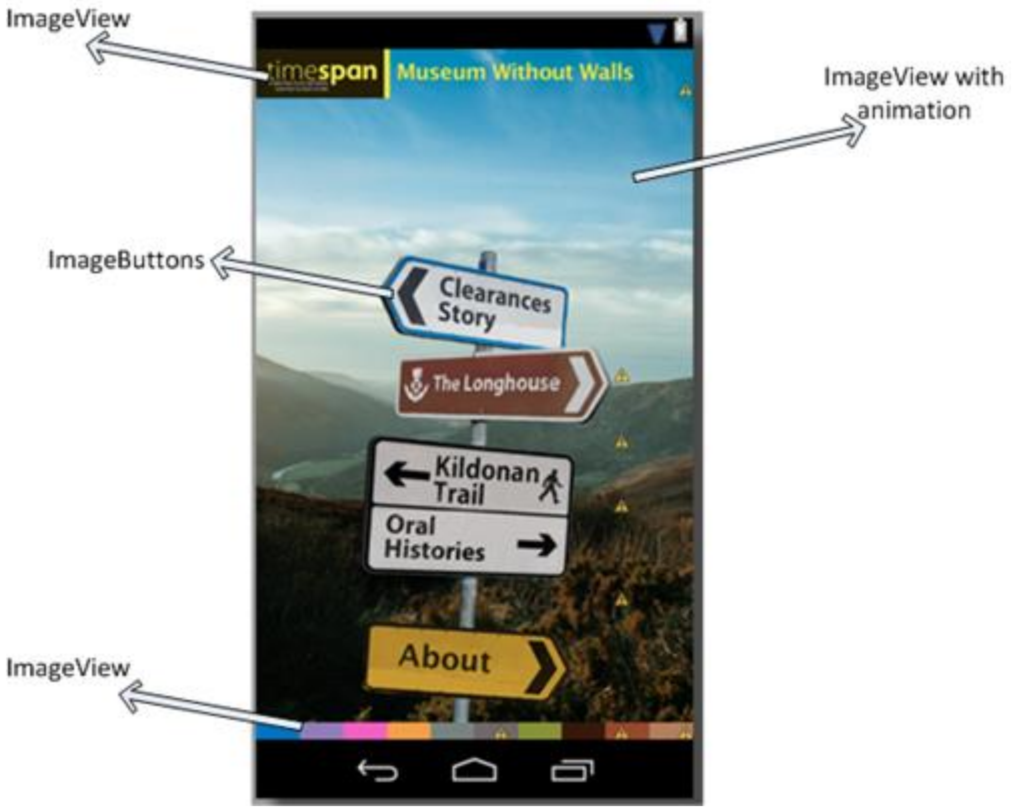


Figure 23 - Main Screen Widgets

The main screen is composed of several widgets such as *ImageViews* and *ImageButtons*. These widgets are described later in this part. The background *ImageView* rotates after a specified interval. Another interesting feature is the irregular shape menu buttons created using *ImageButtons* for the main menu.

This part briefly describes the different type of widgets used in this project. Detailed information can be found in the Android documentation[11].

➤ TextView

A TextView widget is used for displaying text on the user interface. It offers different customization such as changing the font type, font color and font size. Other advanced features include such as allowing the user to edit text.

Figure 24 shows how to define a TextView in the Android XML layout.

```
<TextView
    android:id="@+id/top_bar_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Kildonan Trail"
    android:textColor="#251e04"
    android:textSize="22sp"
/>
```

Figure 24 - Defining a TextView Widget

Figure 25 shows how TextView widget is used in this project as the title text on every screen.

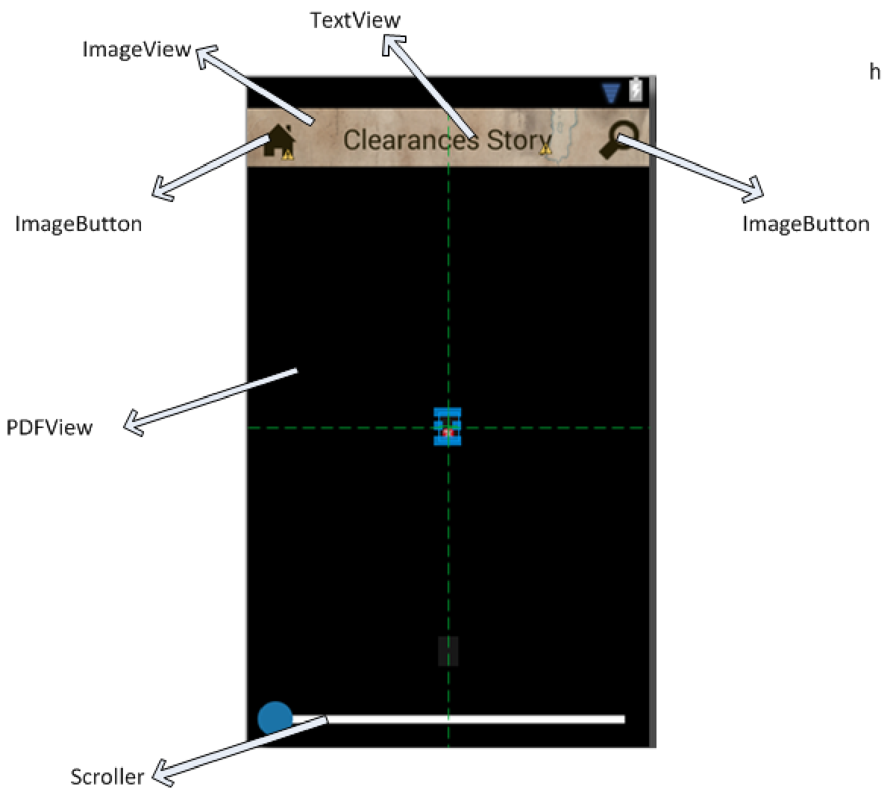


Figure 25 - Clearances Story Screen Widgets

[11]http://developer.android.com/reference/android/widget/package-summary.html

➢ ImageView

An ImageView widget is used for displaying image on the user interface. The source of image can be inside the project (local path) or it can also be loaded from URL. The ImageView widget supports different features such as resizing, scaling and opacity control.

The local images are placed inside *res/drawable* folder. To support multiple screens, the image should be placed in each drawable folder following hdpi, ldpi, xhdpi etc. Figure 26 shows how to define an ImageView in the Android XML layout and Figure 23 shows the ImageView widget being used on different screens of this project for displaying images.

```
<ImageView
    android:id="@+id/top_bar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:background="@drawable/top_bar" />
```

Figure 26 - Defining an ImageView Widget

➢ ImageButton

An ImageButton widget is used for displaying buttons in the form of images on the user interface. User can tap on the button which triggers an event in the application. ImageButton allows the developers to set different images for different states – *normal* and *pressed*. The XML for ImageButton is created separately and placed in the *drawable* folder. Figure 27 shows how to define ImageButton in the Android XML layout and Figure 23 shows how ImageButton widget is used for creating the menu buttons on main screen.

```
<ImageButton
    android:id="@+id/audioButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:background="@drawable/btn_audio_state"
    android:layout_centerVertical="true"
    android:layout_marginRight="10dp"
/>
```

Figure 27 - Defining an ImageButton Widget

➢ ListView

A ListView widget is used for displaying a scrollable list of items on the user interface. The user can interact by scrolling and tapping on the list items. The ListView is composed of an *Adapter*, which is used for holding the list items. The list items can be stored in an array, database or even by making a HTTP web call. Each list item is converted into a View before displaying on the interface. Figure 28 shows how to define ListView XML in the Android XML Layout.

```
<ListView
    android:id="@+id/content_list_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_below="@id/gallery_relative_layout"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"
    android:paddingTop="8dp"
    android:layout_marginBottom="5dp"
    android:background="@drawable/map_table_bg"
/>
```

Figure 28 - Defining a ListView Widget

There are two classes proved by Android - *ListView* and *ExpandableListView*, both capable to display list of scrollable items. The ExpandableListView provides grouping of items. Both classes support list item of any type so developers can create custom list item types. The Adapter of the ListView manages the data source in the form of list items and loads them into individual rows. It extends the BaseAdapter class. Each row can represent a simple text data to complex graphical interface. Developers can also create custom row interfaces.

Figure 29 shows how ListView widget is used to create list of items in the Kildonan Trail screen.

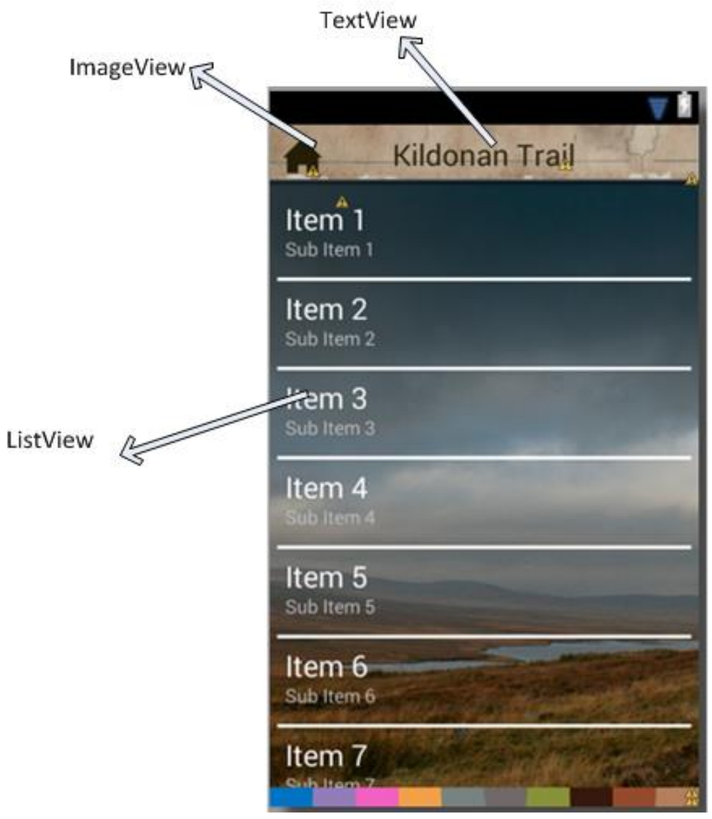


Figure 29- Kildonan Trail NearBy Screen Widgets

➢ MapView

A MapView is used to display *Open Street* map widget. It is an open source widget provided by the OSMDroid library. The data source for map can be online or offline map tiles. It can display maps from various sources including Open Street API, Google Maps API and for this project reading the map tiles from MBTiles. Figure 30 shows the code to define MapView in the Android XML layout.

```
<com.timespan.map.BoundedMapView
        android:id="@+id/mapview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/mapViewBottomColorBar"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/top_relative_layout"
        android:layout_marginTop="-5dp"
        android:enabled="true"
        android:visibility="visible" />
```

Figure 30 - Defining a MapView Widget

Figure 31 shows how the MapView widget is displayed to the user.

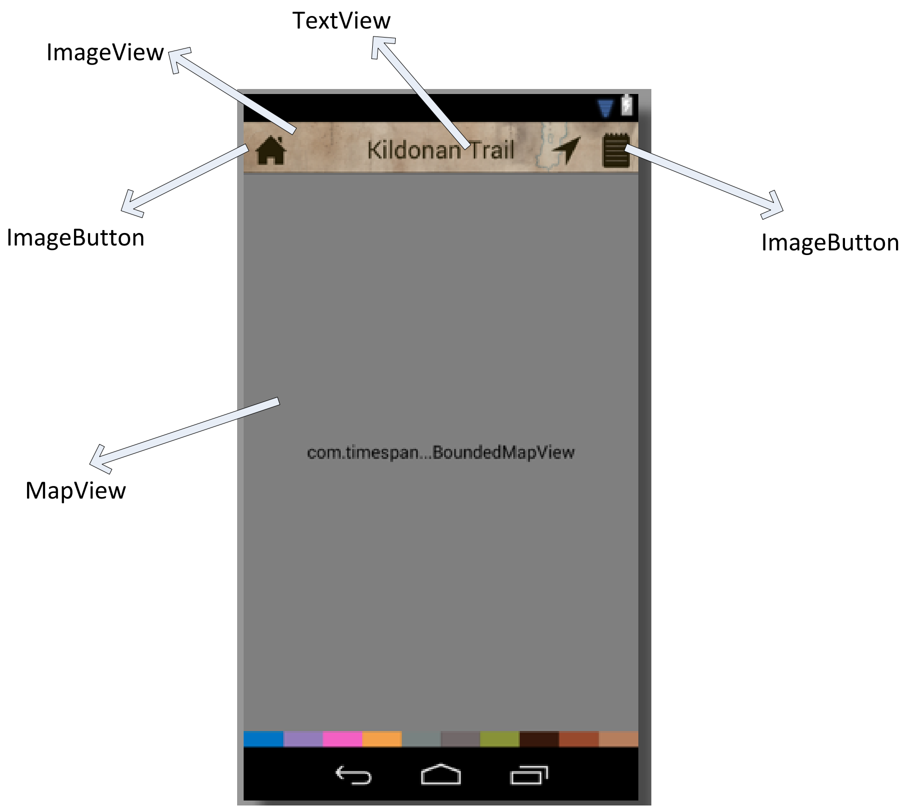


Figure 31 - Kildonan Trail Screen Widgets

6.5 Supporting Multiple Screen Sizes

Designing an Android application which works perfect on a single device is not sufficient. Android is an open source operating system which is used by different hardware vendors for their smartphones. These devices vary from small to medium to extra-large screens as shown in Figure 32. When developing an application, the developers have to make sure their application performs consistent on all screens.

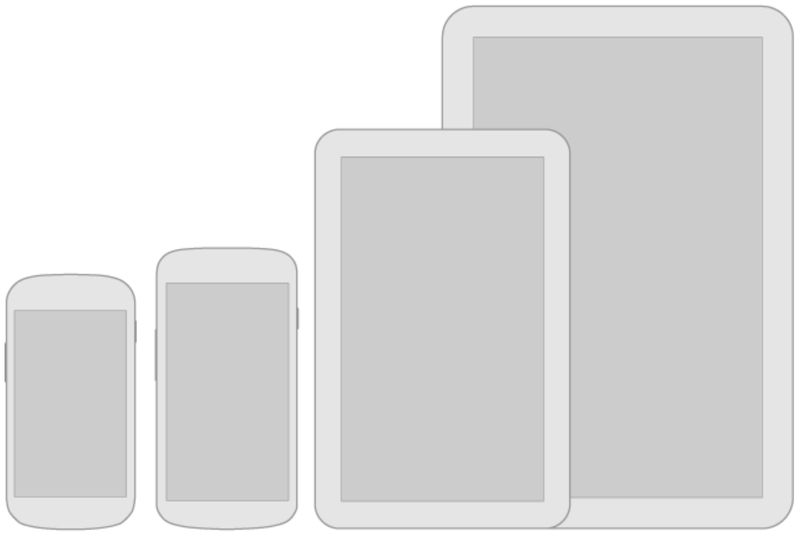


Figure 32 - Android Screen Sizes[12]

Android provides a user interface platform to solve this problem. This platform handles the user interface behaviour on screens of different sizes. There are basically four types of screens defined in the documentation[13]:

- *xhdpi* – extra high density screens are at least 960 x 720 pixels
- *hdpi* – high density screens are at least 640 x 480 pixels
- *mdpi* - medium density screens are at least 470 x 320 pixels
- *ldpi* – low density screens are at least 426 x 320pixels

The ratio between these screens is shown in Figure 33 :

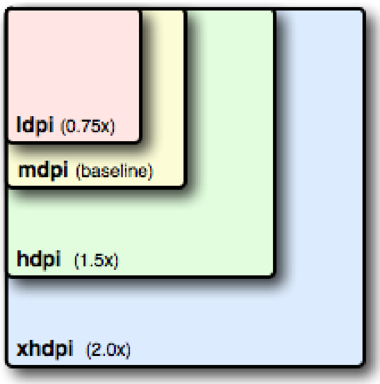


Figure 33 - Ratio between different screen sizes[14]

[12] http://developer.android.com/design/style/devices-displays.html
[13] http://developer.android.com/guide/practices/screens_support.html

The system handles scaling and resizing to support the application on different screens. To optimize the performance, images and layout XMLs should be places in their respective folders. The folders are named according to the screen sizes as shown in Figure 34.

```
res/layout/my_layout.xml             // layout for normal screen size ("default")
res/layout-small/my_layout.xml       // layout for small screen size
res/layout-large/my_layout.xml       // layout for large screen size
res/layout-xlarge/my_layout.xml      // layout for extra large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra large in landscape orientation

res/drawable-mdpi/my_icon.png        // bitmap for medium density
res/drawable-hdpi/my_icon.png        // bitmap for high density
res/drawable-xhdpi/my_icon.png       // bitmap for extra high density
```

Figure 34 - Android Files Hierarchy for Different Screen Sizes

6.6 Conclusion

This chapter discussed the application design for this Android project. It explained the software design and pattern used in this project and how the Android user interface is composed of different screen widgets. It also discussed about how the developer can support multiple screens for the application using the strategy provided by Android platform.

The next chapter covers implementing this design and user interface into a running application. It covers the major tasks achieved during Android development of this project.

[14] Image Credit: http://developer.android.com/guide/practices/screens_support.html

Chapter 7 | **Android Software Development**

This chapter discusses the implementation of this Android application. The first part of this chapter discusses about the environment setup and required tools/libraries for this project. The second part covers the basics of Android applications and third part covers the implementation details which are specific to this project.

7.1 Development Tools and Libraries

Every software is developed using a specific set of tools and libraries. Following are the tools and libraries required for developing this Android application.

➢ **JDK**

Java Development Kit[15] (JDK) is the core of any java application. Android uses the Java SE platform and JDK provides the implementation for Java SE. The JDK is used for developing and debugging java applications. It includes the tools such as *javac* compiler, *javadoc* and many other features.

➢ **Android SDK**

Android Software Development Kit[16] (SDK) provides the core libraries and tools for Android development such the emulator. Developers can create applications using this SDK. The Android SDK also contains sample projects with source code for guidance and it provides the *Dalvik* runtime which runs on top of Linux Kernel, designed for Android platform.

➢ **Eclipse ADT**

Android Development Tools17 (ADT) is a plugin for the Eclipse IDE which integrates JDK and Android SDK to provide developers with the Android application development environment. ADT adds features in Eclipse such as build new Android projects, import, export and Android Layout editing capabilities. Finally the Android application can be easily debug using ADT and generates the executable Android file called the APK package.

➢ **Android NDK**

Android Native Development Kit[18] (NDK) is a set of tools which allows writing native-code such as C and C++ in Java Android application. The application can also reuse existing native-code libraries. Applications which use the NDK are self-contained and CPU-intensive such as image and signal processing and physics simulation.

[15] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[16] http://developer.android.com/sdk/index.html
[17] http://developer.android.com/sdk/index.html
[18] http://developer.android.com/tools/sdk/ndk/index.html

7.2 The Android Software Stack

Before explaining the details specific to this project, this part discusses the basics and foundation of Android platform which is made up of different components as shown in Figure 35.

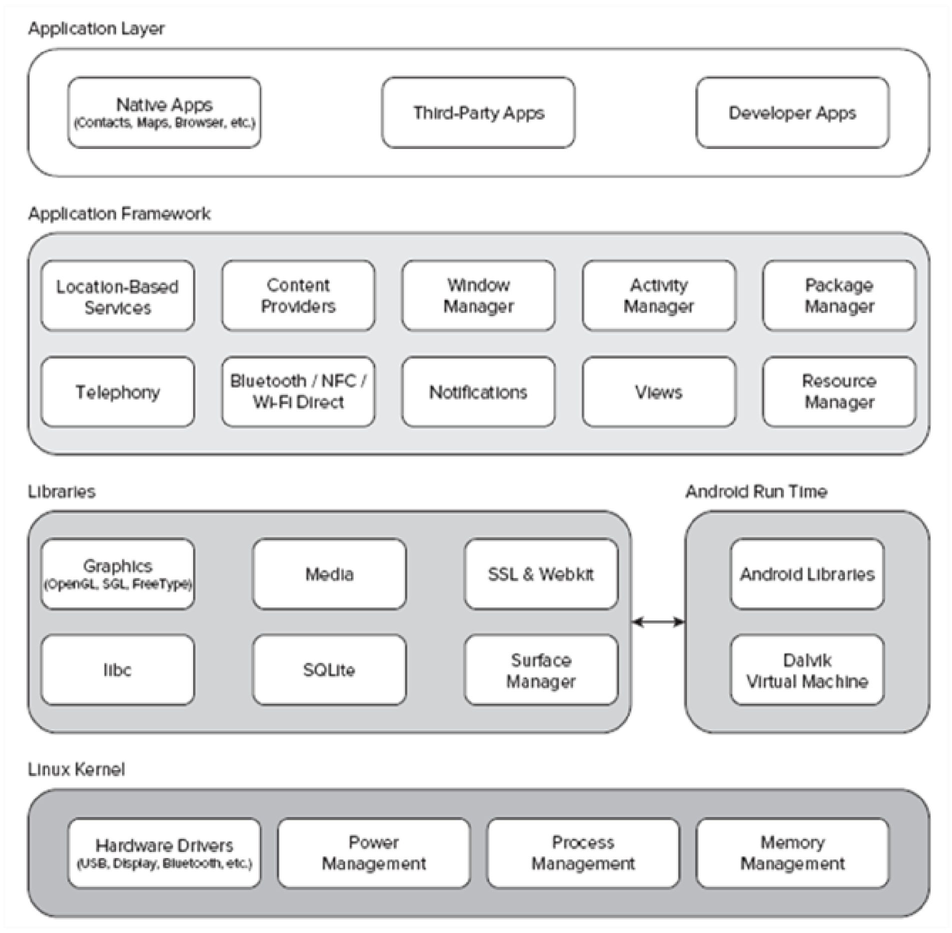


Figure 35 - Android Software Stack[39]

The Android Software Stack is composed of Linux Kernel and C/C++ libraries which work under the layer of application framework to provide services to run applications on the Android devices using the Android Run Time[39]. The components are briefly explained below.

Linux Kernel: It provides an abstraction between the hardware and other layers in the software stack. This layer is responsible for core services such as hardware drivers, process and memory management.

Libraries: It provides C/C++ libraries for media handling such as audio/video, SQLite native database, OpenGL 2D and 3D graphic libraries and SSL /WebKit for internet browsing.

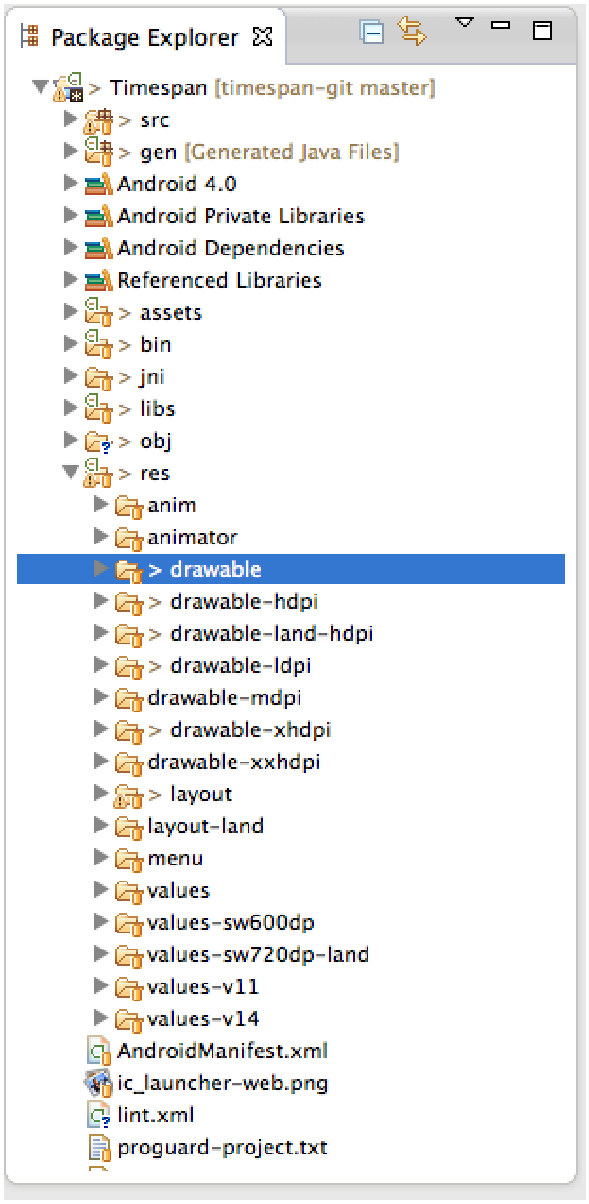
Android Run Time: It provides the Android VM known as Dalvik to run Linux on mobile phones. This is the main difference between Linux and Android operating system. Dalvik VM and Android Libraries inside the Run Time layers powers the applications and forms the application framework.

Application Framework: It provides the core classes used by developers to write their applications. The application framework is an abstraction layer between the Kernel Libraries and user application. Such classes are used for geo-location, creating activity views, making phone calls, SMS services and notification management etc.

Application layer: It provides the environment to run Android applications. This includes native and developers' applications. These applications use the application framework layer classes to perform required tasks.

7.3 Project Hierarchy

Android follows a specific folder hierarchy to maintain its file system. Figure 36 shows the folder hierarchy for this project.



```
Package Explorer
▼ > Timespan [timespan-git master]
  ▶ > src
  ▶ > gen [Generated Java Files]
  ▶ Android 4.0
  ▶ Android Private Libraries
  ▶ Android Dependencies
  ▶ Referenced Libraries
  ▶ > assets
  ▶ > bin
  ▶ > jni
  ▶ > libs
  ▶ > obj
  ▼ > res
    ▶ anim
    ▶ animator
    ▶ > drawable
    ▶ > drawable-hdpi
    ▶ > drawable-land-hdpi
    ▶ > drawable-ldpi
    ▶ drawable-mdpi
    ▶ > drawable-xhdpi
    ▶ drawable-xxhdpi
    ▶ > layout
    ▶ layout-land
    ▶ menu
    ▶ values
    ▶ values-sw600dp
    ▶ values-sw720dp-land
    ▶ values-v11
    ▶ values-v14
  AndroidManifest.xml
  ic_launcher-web.png
  lint.xml
  proguard-project.txt
```

Figure 36 - Android Project Hierarchy

7.4 Class Diagram

A class diagram gives the technical implementation details of an application written in any object oriented language. Android uses Java, which is an object-oriented language and the classes in Android are called *Activities*. These activities represent views. For this application, we are going to create activities for each screen. Figure 37 shows the activities and their relations in this project.

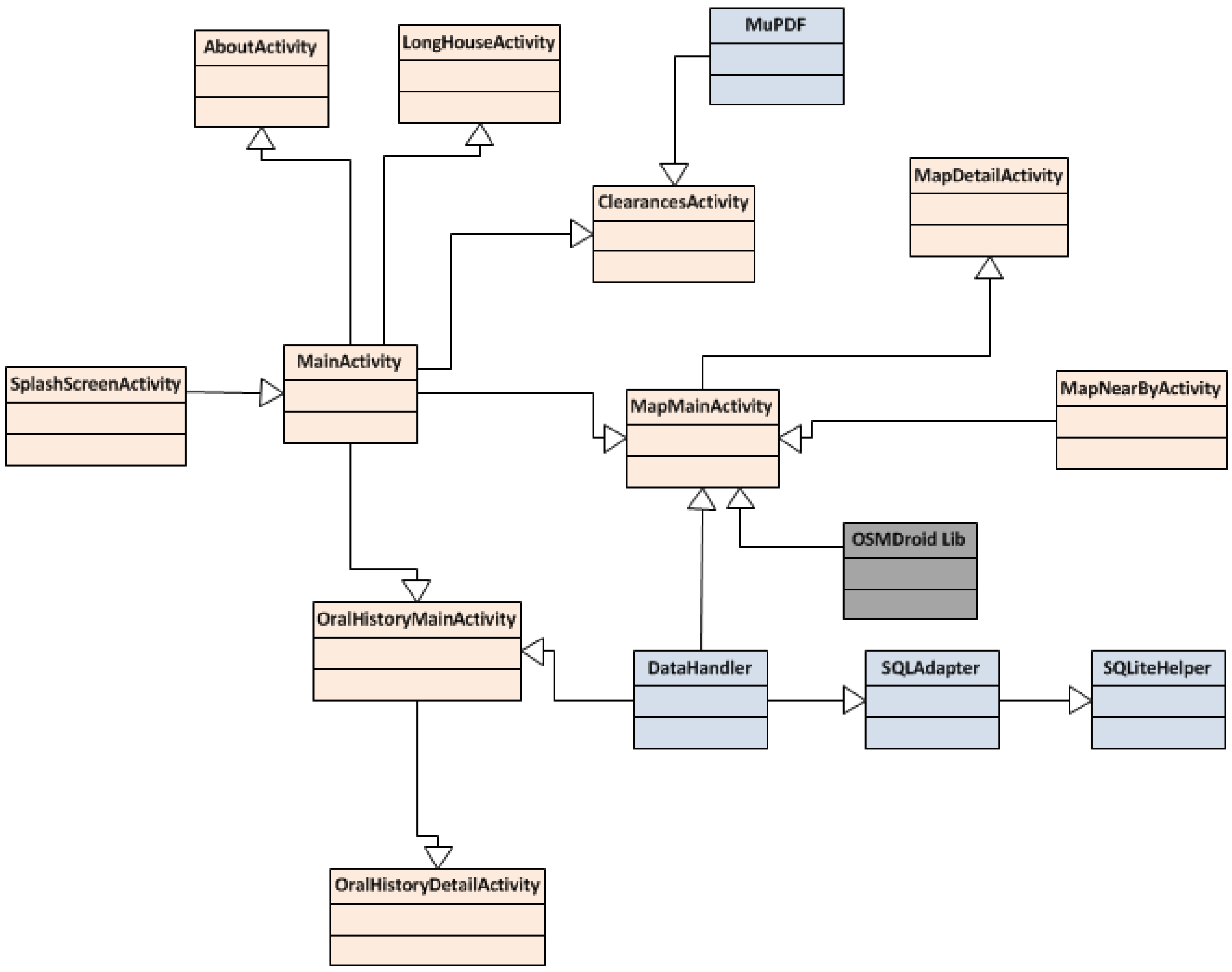


Figure 37 - Class Diagram

The activities are briefly explained below:

SplashScreenActivity: This activity displays the app launch animation/intro screen. After a certain time (3 seconds), the next activity is loaded automatically for the user.

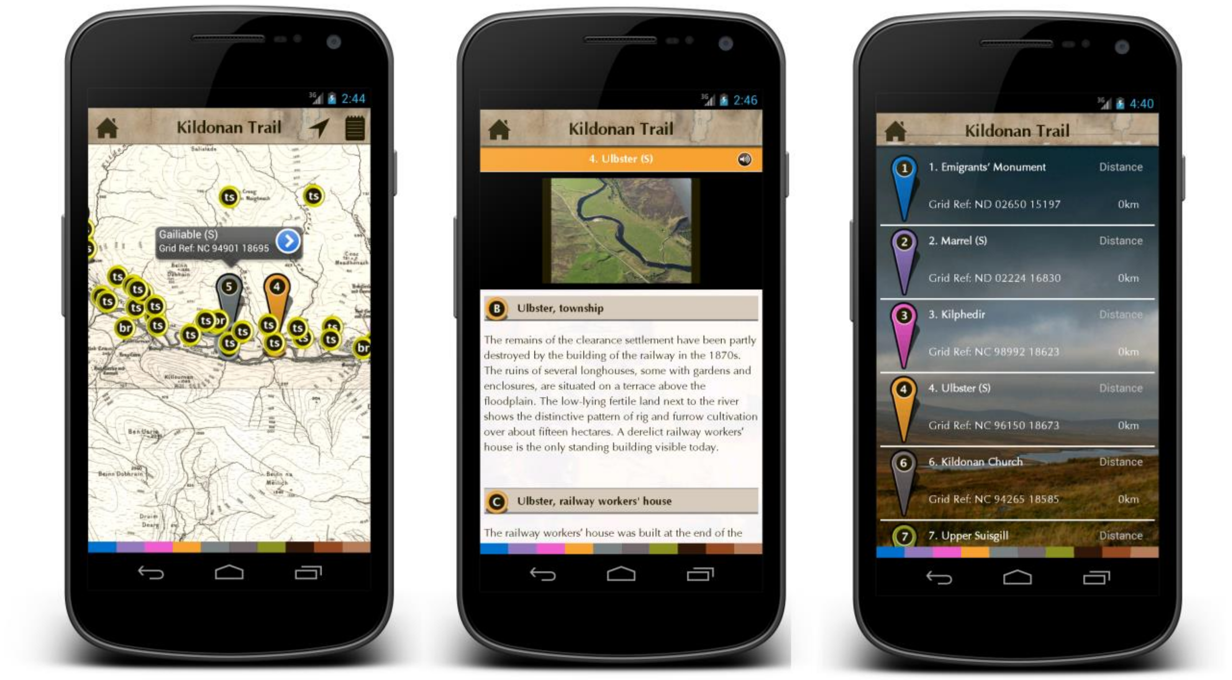
MainActivity: This activity is the main home screen of Timespan Android App. From here, user can select Clearances Story, Kildonan Trail, Oral Histories, the Longhouse or the About screen.

MapMainActivity: This activity shows the Geo-referenced historical Kildonan Trail offline map. User can scroll and zoom using the touchscreen. This screen supports both landscape and portrait orientation. There are map pins present on the map as show in Figure 38. User can select any pin and open a detail view details about each pin.

This activity uses the *OSMDroid* library to show map and *DataHandler* for fetching map pins location information from application database. The left image in Figure 38 shows how MapMainActivity is displayed to the user.

MapDetailActivity: This activity shows a photo-gallery and text information about the selected point from map. The data is shown as a list and user can scroll vertically. The photo gallery allows horizontal scroll to swipe through images. The middle image in Figure 38 shows how MapDetailActivity is displayed to the user.

MapNearByActivity: This activity uses user GPS location and display a list sorted by distance to different pin locations on the map. User can tap on any pin to go to *MapDetailActivity*. The right image in Figure 38 shows how MapNearByActivity is displayed to the user.



B Ulbster, township

The remains of the clearance settlement have been partly destroyed by the building of the railway in the 1870s. The ruins of several longhouses, some with gardens and enclosures, are situated on a terrace above the floodplain. The low-lying fertile land next to the river shows the distinctive pattern of rig and furrow cultivation over about fifteen hectares. A derelict railway workers' house is the only standing building visible today.

C Ulbster, railway workers' house

The railway workers' house was built at the end of the

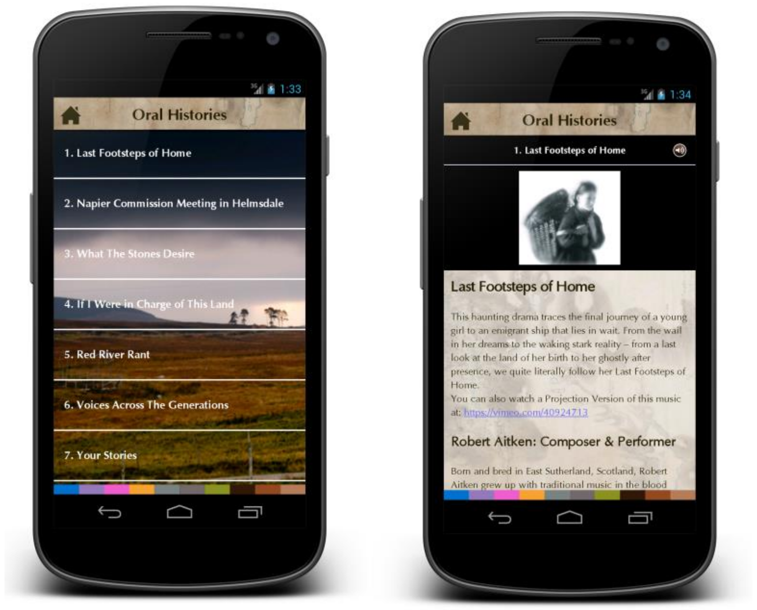
1. Emigrants' Monument	Distance
Grid Ref: ND 02650 15197	0km
2. Marrel (S)	Distance
Grid Ref: ND 02224 16830	0km
3. Kilphedir	Distance
Grid Ref: NC 98992 18623	0km
4. Ulbster (S)	Distance
Grid Ref: NC 96150 18673	0km
6. Kildonan Church	Distance
Grid Ref: NC 94265 18585	0km
7. Upper Suisgill	Distance

Figure 38 – MapMainActivity, MapDetailActivity and MapNearbyActivity

OralHistoryMainActivity: This activity shows the audio podcasts which are displayed in a list view. User can select any podcast which opens the *OralHistoryDetailActivity*.

OralHistoryDetailActivity: This activity shows a photogallery and audio button to the user.

Figure 39 shows *OralHistoryMainActivity* and *OralHistoryDetailActivity* are displayed to the user.


1:33
Oral Histories
1. Last Footsteps of Home
2. Napier Commission Meeting in Helmsdale
3. What The Stones Desire
4. If I Were in Charge of This Land
5. Red River Rant
6. Voices Across The Generations
7. Your Stories
1:34
Oral Histories
1. Last Footsteps of Home

Last Footsteps of Home

This haunting drama traces the final journey of a young girl to an emigrant ship that lies in wait. From the wail in her dreams to the waking stark reality – from a last look at the land of her birth to her ghostly after presence, we quite literally follow her Last Footsteps of Home.
You can also watch a Projection Version of this music at: https://vimeo.com/40924713

Robert Aitken: Composer & Performer

Born and bred in East Sutherland, Scotland, Robert Aitken grew up with traditional music in the blood

Figure 39 - OralHistoryMainActivity and OralHistoryDetailActivity

ClearancesActivity: This activity shows the user interactive PDF book containing text and images. It uses *MuPDF* library which uses native Android NDK for fast performance. User can swipe through pages and perform search. Figure 40 shows how *ClearancesActivity* is displayed to user.

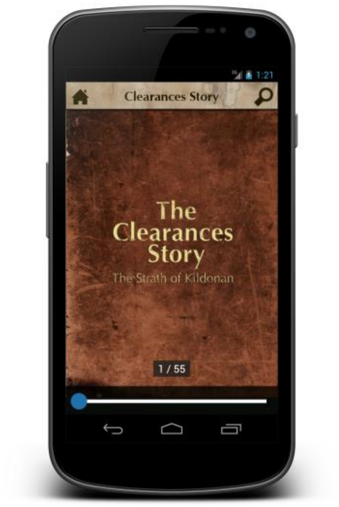


Figure 40 - ClearancesActivity

7.5 Offline Maps

The major part of this project focused on creating the *Kildonan Trail* map. This is based on an historical offline map of the Starth of Kildonan. Developing this feature involved major challenges in this project. Due to the unavailability of source code of iOS App and map file assets, this objective have been achieved by extracting the iOS .ipa (binary) file and obtaining missing assets.

➢ **Extracting the iOS IPA File:**

The first step is to decompile the iOS IPA file and extract the source Map tiles. Obtaining the IPA is pretty straight forward. After installing Timespan iOS App on iPhone, connect it with *iTunes* and copy paste the App in hard drive. The IPA files are encrypted by XCode, (Apple's development IDE).

Figure 41shows the file structure of an IPA file.

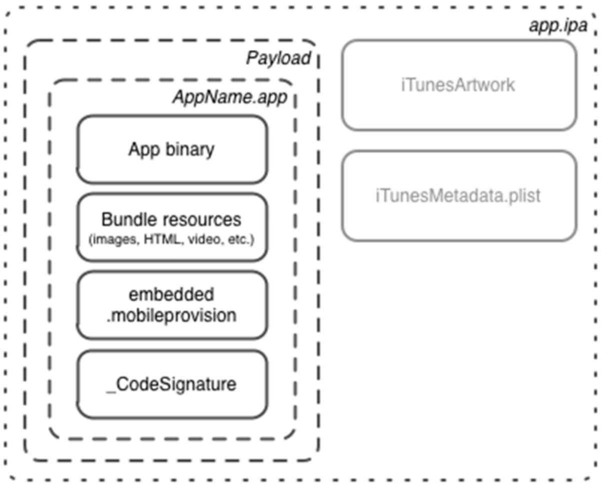


Figure 41 - iOS IPA Structure[42]

After obtaining the IPA file, it needs to be decrypted to obtain the data inside IPA file. A Ruby script called '*appcrush'*[40] is used to perform the decryption. This script uses *PNGCrush* application inside XCode to decrypt the IPA contents and provides the readable data.

After the extraction and decryption of IPA, the map tiles file is obtained in *MBTiles* format.

➢ **Loading MBTiles in MapView**

This part describes the background information about MBTiles file and format. MBTiles is a compact SQLite implementation to store map tiles in a database format. It supports interactivity gird in the form of coordinates as latitude and longitude. It contains the metadata information such as map bounds and centre. MBTiles support compression to reduce the size and optimize application performance.

The tiles are created as small squares of the total map as shown in Figure 42. For this project, each tile is a 256x256 resolution image.

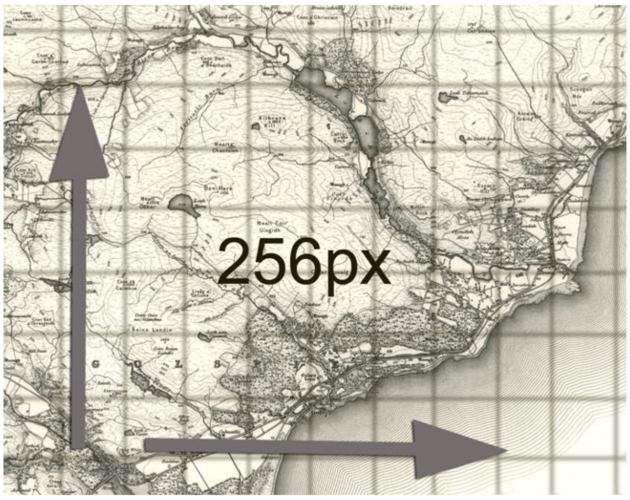


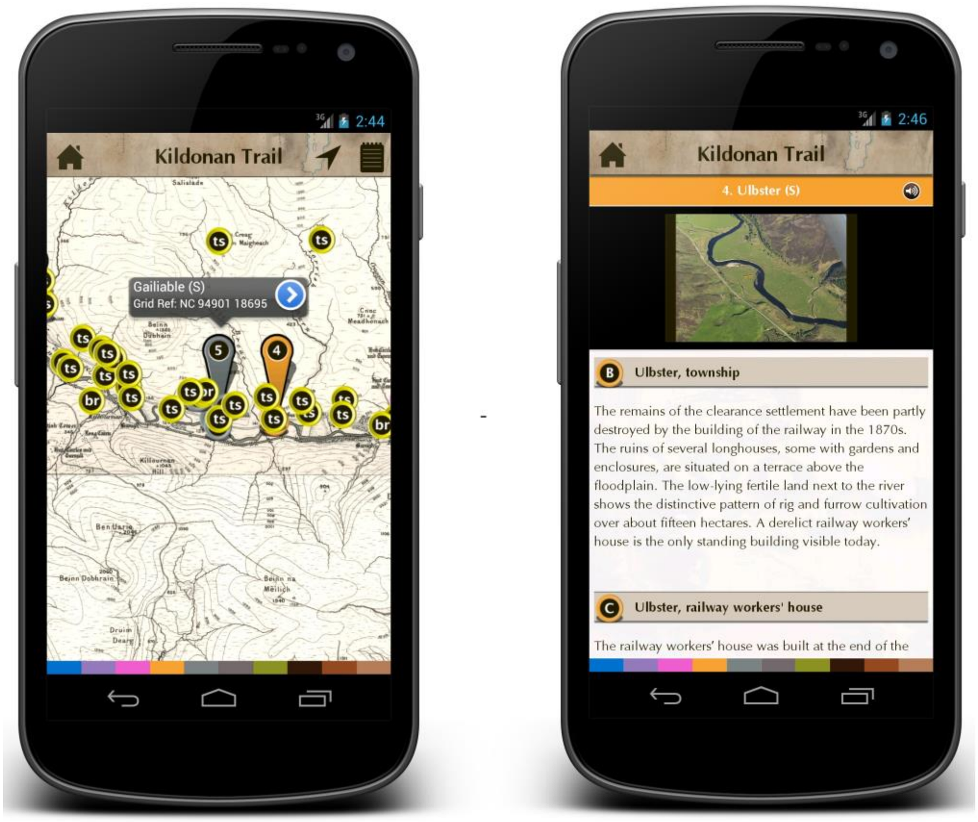
Figure 42 - MBTiles format visualisation

MBTiles is a portable map tiles storage format. Timespan iOS application is using MBTiles for the Kildonan Trail map feature. These tiles can be reuse on Android by using the open-source library called *OSMDroid*. Figure 43 shows the initialization of MBTiles on a MapView using the OSMDroid library.

```
private void initMBTiles(){

        //initialize the tile source
        XYTileSource MBTILESRENDER = new XYTileSource("mbtiles",
        ResourceProxy.string.offline_mode, MAP_MIN_ZOOM, MAP_MAX_ZOOM, 256,
        ".png", "http://timespan.org/");

        SimpleRegisterReceiver simpleReceiver = new
        SimpleRegisterReceiver(this);

        File f = new File(Environment.getExternalStorageDirectory(),
        "clearances-story-app-map-05.mbtiles");

        IArchiveFile[] files = { MBTilesFileArchive.getDatabaseFileArchive(f) };

        MapTileModuleProviderBase moduleProvider = new
        MapTileFileArchiveProvider(simpleReceiver, MBTILESRENDER, files);
        MapTileProviderArray tileProviderArray = new
        MapTileProviderArray(MBTILESRENDER, this, new
        MapTileModuleProviderBase[] { moduleProvider });


        //load the tilesource into MapView widget

        TilesOverlay tilesOverlay = new TilesOverlay(tileProviderArray, this);

        this.mapView = (MapView) findViewById(R.id.mapview);

        this.mapView.getOverlayManager().setTilesOverlay(tilesOverlay);
        tilesOverlay.setLoadingBackgroundColor(Color.WHITE);
        this.mapView.postInvalidate();

}
```

Figure 43 - Initializing and Loading MBTiles in MapView widget

➢ **Displaying the Map Pins and Map Detail View**



B Ulbster, township

The remains of the clearance settlement have been partly destroyed by the building of the railway in the 1870s. The ruins of several longhouses, some with gardens and enclosures, are situated on a terrace above the floodplain. The low-lying fertile land next to the river shows the distinctive pattern of rig and furrow cultivation over about fifteen hectares. A derelict railway workers' house is the only standing building visible today.

C Ulbster, railway workers' house

The railway workers' house was built at the end of the

Figure 44 - Map Pins(Left) and Map Detail View (Right)

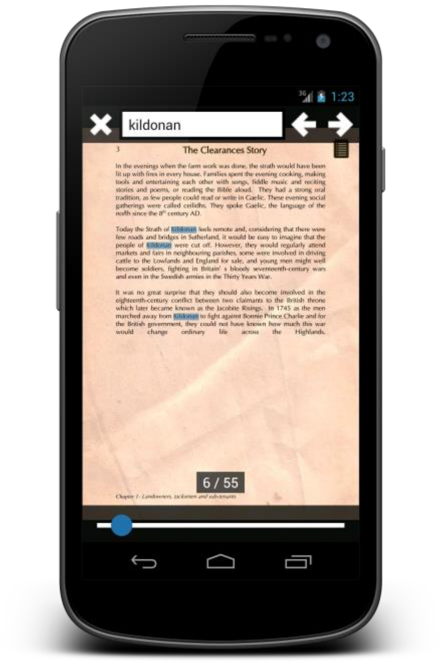
The map pins are displayed by placing them on the map using Latitude and Longitude grid. A digital map is one which is built with the latitude and longitude database. The MBTiles map file for this project contains the digital information about the Strath of Kildonan. OSMDroid supports displaying of map annotation pins. Following two classes are used to draw the pins:

- *ExtendedOverlayItem*
- *ItemizedOverlayWithBubble*

The pins are distributed as ten main pins and each main pin has several sub pins. The data for the pins is stored in the application SQLite database. The pins with numbers are the main pins and pins with al-phabets are the sub pins. On tapping on any pin, a callout is displayed as shown in Figure 44 . The callout contains a button to open a further detail view called *MapDetai ViewActivity*.

The detail view contains a photo gallery, list view and audio options. User can scroll through the list view and slide left/right on the photo gallery. The audio can be played in the background so user can still use the list view and photo gallery while the audio is being played.

7.6 PDF Reader



3 The Clearances Story

In the evenings when the farm work was done, the strath would have been lit up with fires in every house. Families spent the evening cooking, making tools and entertaining each other with songs, fiddle music and reciting stories and poems, or reading the Bible aloud. They had a strong oral tradition, as few people could read or write in Gaelic. These evening social gatherings were called ceilidhs. They spoke Gaelic, the language of the north since the 8th century AD.

Today the Strath of Kildonan feels remote and, considering that there were few roads and bridges in Sutherland, it would be easy to imagine that the people of Kildonan were cut off. However, they would regularly attend markets and fairs in neighbouring parishes, some were involved in driving cattle to the Lowlands and England for sale, and young men might well become soldiers, fighting in Britain' s bloody seventeenth-century wars and even in the Swedish armies in the Thirty Years War.

It was no great surprise that they should also become involved in the eighteenth-century conflict between two claimants to the British throne which later became known as the Jacobite Risings. In 1745 as the men marched away from Kildonan to fight against Bonnie Prince Charlie and for the British government, they could not have known how much this war would change ordinary life across the Highlands.

Chapter 1 - Landowners, tacksmen and subtenants

Figure 45- MuPDF Reader

The *Clearances Story* feature is an interactive book with a background theme, hyperlinks and images. The iOS application has not implemented this feature in a reusable format, such as PDF. So the first task is to develop the book in PDF format as shown in Figure 45. Book content is provided in MS Word documents and PNG files. The book contains a table of contents and ten chapters. The user can tap on any chapter from the table of contents to jump to that chapter.

Loading the PDF onto Android device involves handling of memory issues. Rendering and user interaction also raises many performance issues. For this project an open source PDF library MuPDF is used. It is developed by *Artifex Software Inc*[19] and works for Android, Windows and Linux. The integration of MuPDF has been done using online help.[20]

7.7 Conclusion

This chapter covered the implementation details of this project and highlighted the major challenges involved building the application which are the Kildonan Trail and Clearances Story.

The next chapter covers how to publish an Android application on the Android market called the *Google Play Store*[21].

[19] http://www.artifex.com/
[20] http://stackoverflow.com/a/17735927
[21] https://play.google.com/

Chapter 8 | **Publishing**

After completing the development and testing your application on different devices, the final step is to publish the application on the Google Play Store. This chapter discusses in detail about setting up account on the Google Play Store and hosting the external files on Google Play server.

Due to a limited available help for expansion files, this chapter includes demonstration of technical implementation for hosting the external files in the form of code snippets. This demonstration can be helpful for anyone working on a similar problem.

8.1 Setting up Google Play Account

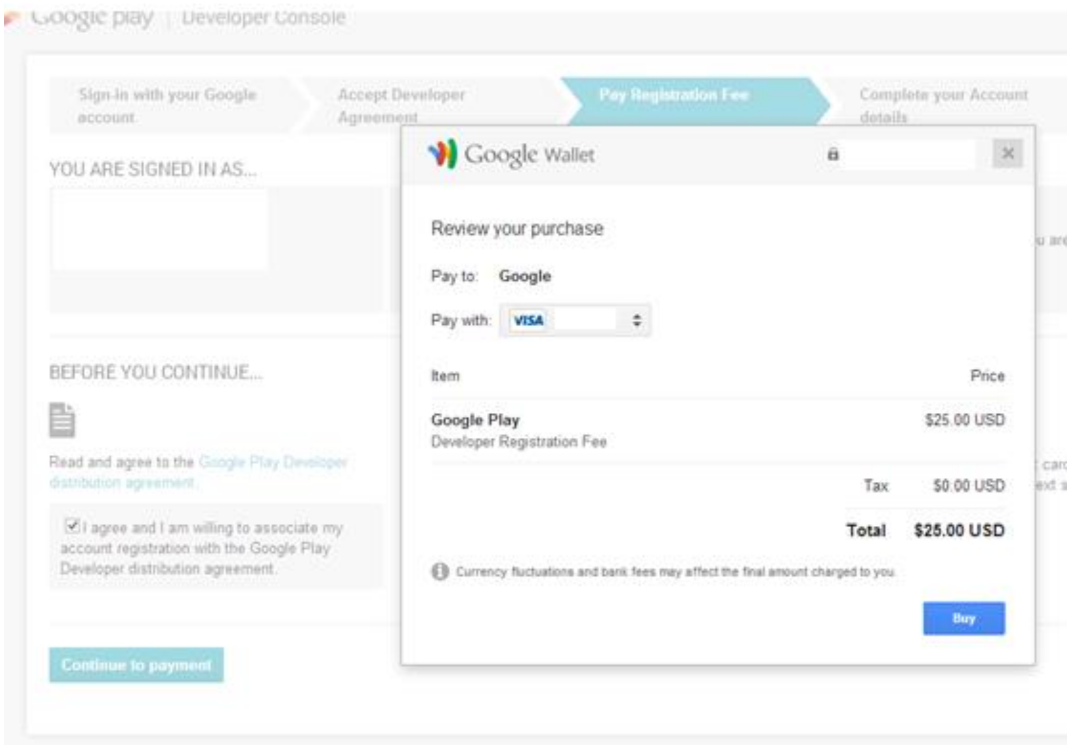
A Google Play developer account provides the facility to publish the application on the market store. Following steps show how to setup the Google Play account.

a) Open URL in browser https://play.google.com/apps/publish/signup/

b) Login with using your Google account

c) Read and accept the Developer Agreement

d) Pay the registration fee as shown in Figure 46.


Google play | Developer Console
Sign-in with your Google account
Accept Developer Agreement
Pay Registration Fee
Complete your Account details
YOU ARE SIGNED IN AS...
BEFORE YOU CONTINUE...
Continue to payment
Google Wallet
Buy

Read and agree to the Google Play Developer distribution agreement.

☑ I agree and I am willing to associate my account registration with the Google Play Developer distribution agreement.

Review your purchase

Pay to: Google

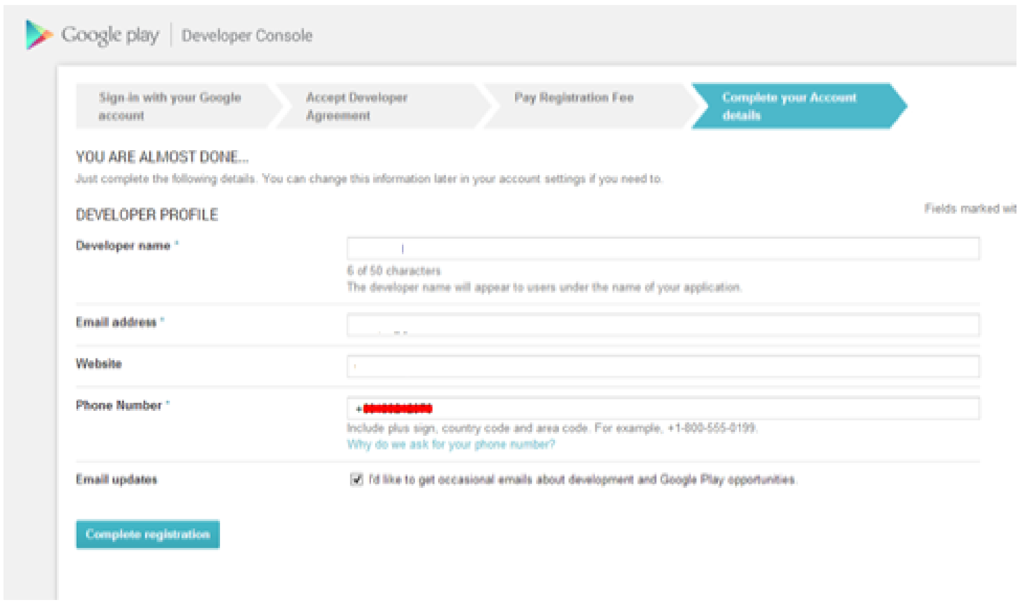
Pay with: VISA

Item	Price
Google Play Developer Registration Fee	$25.00 USD
Tax	$0.00 USD
Total	**$25.00 USD**

Currency fluctuations and bank fees may affect the final amount charged to you.

Figure 46 - Google Play Store Registration Payment

e) After accepting the payment, the signup process asks for account details as shown in Figure 47.


Google play
Developer Console
Sign in with your Google account
Accept Developer Agreement
Pay Registration Fee
Complete your Account details
Complete registration

YOU ARE ALMOST DONE...

Just complete the following details. You can change this information later in your account settings if you need to.

DEVELOPER PROFILE

Fields marked wit

Developer name *

6 of 50 characters
The developer name will appear to users under the name of your application.

Email address *

Website

Phone Number *

+[redacted]

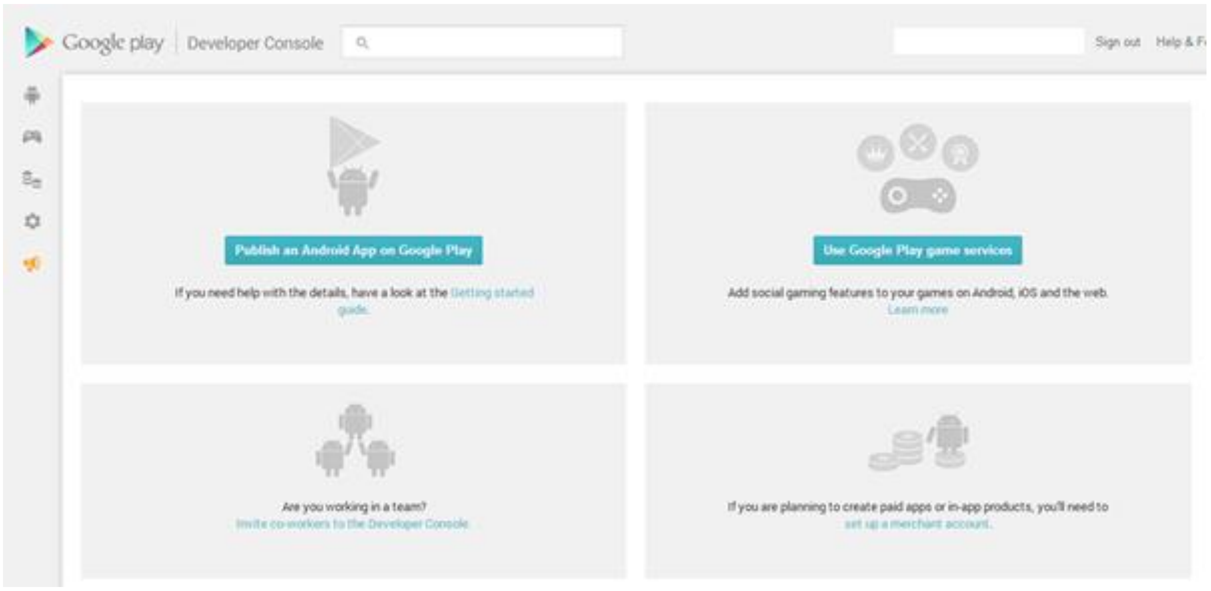
Include plus sign, country code and area code. For example, +1-800-555-0199.
Why do we ask for your phone number?

Email updates

[✓] I'd like to get occasional emails about development and Google Play opportunities.

Figure 47 - Google Play Store Registration Account Details

f) After completing the registration process, the developer *console* can be accessed as shown in Figure 48.


Google play
Developer Console
Sign out
Help & F
Publish an Android App on Google Play
Use Google Play game services

If you need help with the details, have a look at the Getting started guide.

Add social gaming features to your games on Android, iOS and the web.
Learn more

Are you working in a team?
Invite co-workers to the Developer Console.

If you are planning to create paid apps or in-app products, you'll need to set up a merchant account.

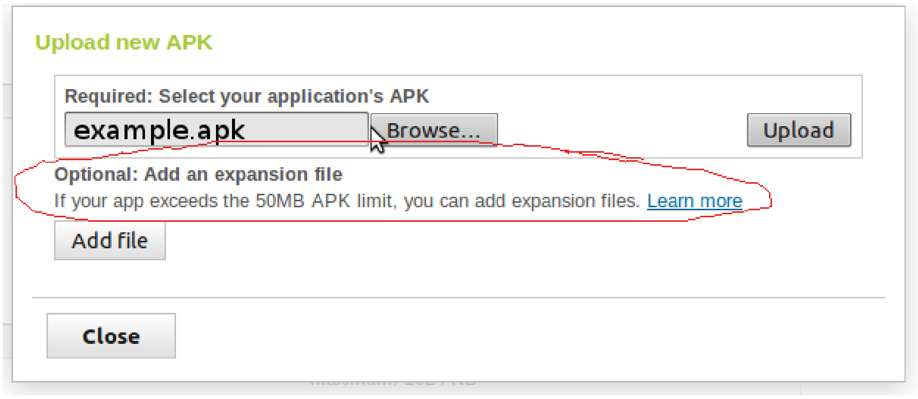
Figure 48 - Google Play Store Developer Console

8.2 Expansion Files

This part covers the technical implementation of hosting external files following the guidelines from Android documentation[22].

An *APK* file is generated after compiling the final implementation of the Android application. This APK is hosted on the Google Play store so that the users can easily download and install on their devices. Google Play store allows a maximum size of 50MB for the APK file. The total size of this application including map tiles, audios and PDF easily exceeds the maximum size allowed by Google. Until March 2013, applications had to host the media files such as audio, video or map files on a private server and download them after the installation on device. Hosting on a private server involves cost and server issues to be handled by the developer. The mechanism is still same but Google now provides the file hosting called Expansion Files.

The expansion files have a maximum limit of 4GB with 2GB per file size. So we can upload two files of 2GB each. The files are in "*obb*" format, which is basically a zip file. The file is downloaded on SD Card and then either extracted or read from the zipped file instead of unzipping the content. The expansion files are uploaded at the time of uploading the *APK* file as shown in Figure 49:


Upload new APK

Required: Select your application's APK

example.apk

Optional: Add an expansion file

If your app exceeds the 50MB APK limit, you can add expansion files. Learn more

Figure 49 - Adding Expansion Files

➢ Expansion Files Basics

There are basically two types of expansion files

- Main – the primary expansion file
- Patch –the optional expansion file

Although there is no restriction of how to use these files but it is recommended that *main* file is used for the main content and the patch is used for the content that needs to be updated with every release. Even

[22] http://developer.android.com/google/play/expansion-files.html

if the application does not offer content updates with every new release, the expansion files are required to be uploaded every time. The expansion files should be recreated because the previous files cannot work with the new "*version code*" of the application.

Before uploading the expansion files, the developer needs to create them locally and test the application. The files follow a naming convention specified in the documentation.

- *[main|patch].<expansion-version>.<package-name>.obb*

The first part *[main or patch]* defines the type of this expansion file. Only one main file and one patch can be uploaded for each application. The second part <*expansion-version*>is an integer value which is same as the application's *android:versionCode* value.

The third part <*package-name*> is the application's package name specified in AndroidManifest.xml

The expansion file should be placed in the SD card after creating using the naming convention. The application should check whether the files are present every time the user uses the app. Following is the file path in SD card for most of the Android devices.

- *Sdcard0/Android/obb/org.timespan.uk/expansionfile.obb*

➢ Expansion Files Implementation

This part discusses the implementation details of Expansion files. Figure 50 gives an overview of the complete process.

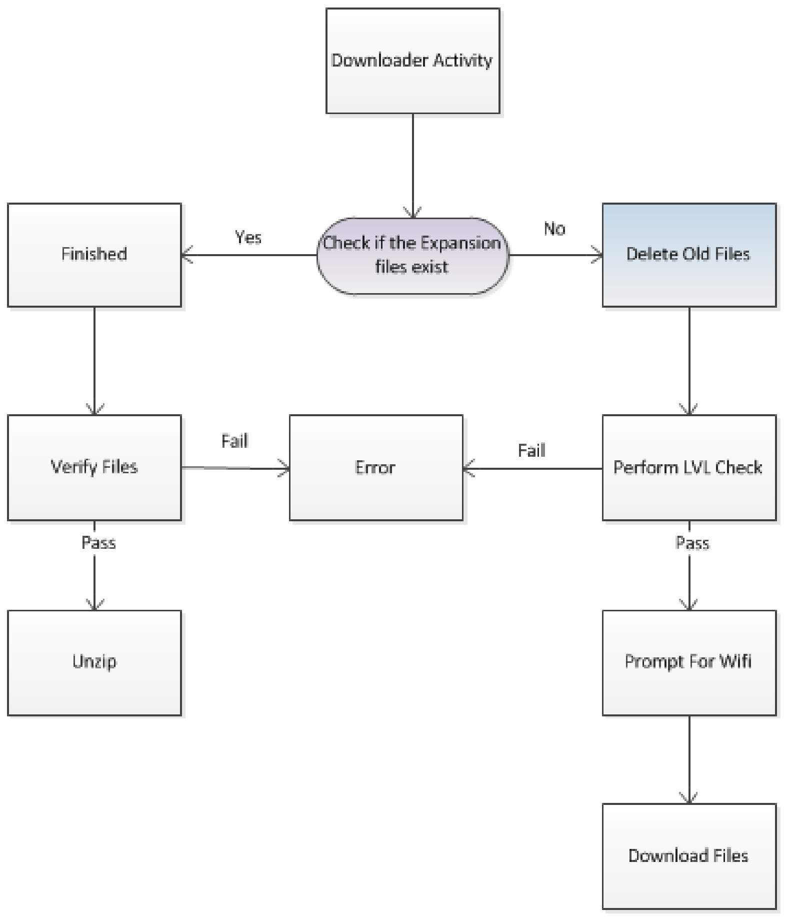


Figure 50 - Expansion files implementation

Most of the time, Google Play allows the application to download and save the expansion files at the time of installation. However, in some cases Google Play cannot allow to download the expansion files or the user might have deleted previously downloaded expansion files. To handle these situations, the app must be able to download the files itself when the main activity starts, using a URL provided by Google Play. Following steps highlights the application installation and expansion file download scenarios.

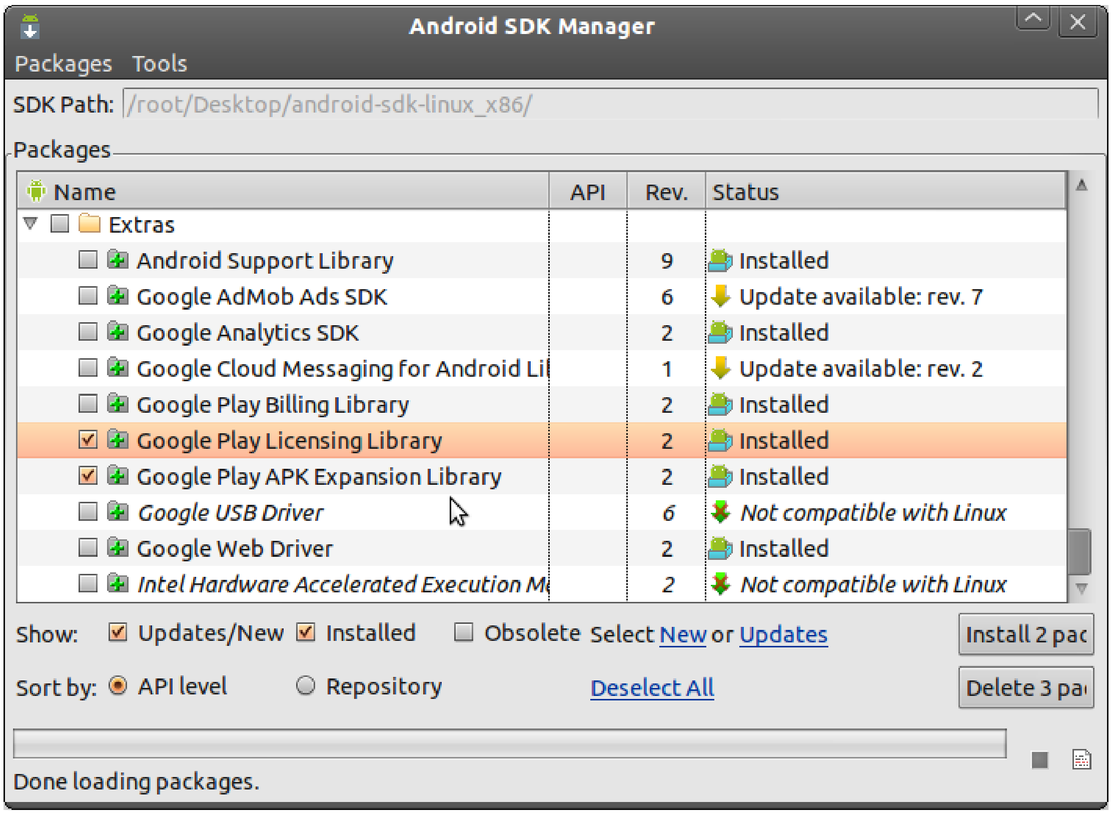
- User installs the app from Google Play store.
- The expansion files are downloaded if Google Play allows the application.
- Else the application is installed and the expansion files are not downloaded if Google Play does not allow the application.
- User launches the application; it should automatically check whether the expansion files are in the SD card and then user can use the application.

- Else the application should download the expansion files over HTTP from Google Play. This is achieved by sending a request to the Google Play client using the *Google Play's Application licensing service*. The application receives the file name, file size, and URL for each expansion file. Then the application starts downloading the files and save them to the SD Card.

Application Licensing Service is used for enforcing licensing policies and ensures that the user has paid for the app. Licensing service has been enhanced to return the download URL of the expansion pack therefore, they can be used in free apps as well. For this reason the application need to need to include *License Verification Library(LVL)* to use expansion files.
A *Downloader Library* is provided by Google which implements the functionality of requesting expansion file URLs using the licensing service and then downloading them. It also supports pause, resume during download.

Using Downloader Library

- First open the Android SDK Manager in Eclipse and expand Extras as shown in Figure 51:
- Check the following two packages
    - *Google Play Licensing Library package*
    - *Google Play APK Expansion Library package*



Name	API	Rev.	Status
Extras			
Android Support Library		9	Installed
Google AdMob Ads SDK		6	Update available: rev. 7
Google Analytics SDK		2	Installed
Google Cloud Messaging for Android Li		1	Update available: rev. 2
Google Play Billing Library		2	Installed
✓ Google Play Licensing Library		2	Installed
✓ Google Play APK Expansion Library		2	Installed
Google USB Driver		*6*	*Not compatible with Linux*
Google Web Driver		2	Installed
Intel Hardware Accelerated Execution M		*2*	*Not compatible with Linux*

Show: ✓ Updates/New ✓ Installed ☐ Obsolete Select New or Updates
Sort by: ● API level ○ Repository Deselect All
Install 2 pac
Delete 3 pa
Done loading packages.

Figure 51 - Installing Google Play Licensing and APK Expansion Library

- After installation, the packages are downloaded in the <SDK/extras/google> directory.

- Import both of the projects *market_licensing* and *downloader_library* in Eclipse.
- Add the following permissions in *Android Manifest* as shown in Figure 52

```
<?xml version="1.0" encoding="UTF-8"?><manifest ...>
    <!-- Required to access Google Play Licensing -->
    <uses-permissionandroid:name="com.android.vending.CHECK_LICENSE"/>

    <!-- Required to download files from Google Play -->
    <uses-permissionandroid:name="android.permission.INTERNET"/>

    <!-- Required to keep CPU alive while downloading files (NOT to keep screen
awake) -->
    <uses-permissionandroid:name="android.permission.WAKE_LOCK"/>

    <!-- Required to poll the state of the network connection and respond to
changes -->
    <uses-permissionandroid:name="android.permission.ACCESS_NETWORK_STATE"/>

    <!-- Required to check whether Wi-Fi is enabled -->
    <uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>

    <!-- Required to read and write the expansion files on shared storage -->
    <uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    ...
</manifest>
```

Figure 52 – Android Manifest Permissions for Downloader Library

The downloader library provides the following features:

- Handles network connectivity by registering *BroadcastReceiver* to listen to *CONNECTIVITY*_ACTION
- In case of download failure, downloader library schedules an RTC_WAKEUP alarm to retry the download
- Verifies SD Card operations such as mounting, space availability, write access and handling duplicate files errors etc. In-case of any error, it notifies the user by a notification

Declare the service in the Android Manifest file as shown in Figure 53.

```
<application ...>
  <receiver android:name=".SampleDownloaderService" />

</ application >
```

Figure 53 - Android Manifest Permissions for Downloader Service

Implementing Alarm Receiver

The downloader service uses alarm service by scheduling an *RTC_WAKEUP* call. This should be defined in the following way as shown in Figure 54 to make sure the application downloads and restarts if necessary.

```
public class SampleAlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        try {

                  DownloaderClientMarshaller.startDownloadServiceIfRequired
            (context, intent,
                                SampleDownloaderService.class);
                    } catch (NameNotFoundException e) {
                        e.printStackTrace();
                    }
                }

      }
```

Figure 54 - Alarm Receiver Initialisation

Declare the receiver in the Android Manifest file as shown in Figure 55:

```
<application ...>
  <receiver android:name=".SampleAlarmReceiver" />

</ application >
```

Figure 55 - Android Manifest Permissions for Alarm Receiver

Starting the download using the Downloader Library requires the following procedures:

- Check for the files if they are downloaded.
- The library determines if the download is required or not when the download is started.

  Possible values are *NO_DOWNLOAD_REQUIRED*, *LVL_CHECK_REQUIRED*, *DOWNLOAD_REQUIRED*.
- Initialize *IStub* if the library returns anything other than NO_DOWNLOAD_REQUIRED.
- *IStub* is used for binding application activity with the downloader service.

Receiving Download Progress

During the download process of expansion files, it is necessary to show progress to the user. The application needs to interact with the download server by implementing *IDownloaderClient* interface. Commonly the activity which starts the download process in the application should implement this interface and display the download progress to the user. Figure 56 shows the Initialization of IDownloaderService interface.

```
private IDownloaderService mRemoteService;
...

@Override
public void onServiceConnected(Messenger m) {
    mRemoteService = DownloaderServiceMarshaller.CreateProxy(m);
    mRemoteService.onClientUpdated(mDownloaderClientStub.getMessenger());

}
```

Figure 56 – Initialization of IDownloaderService interface.

After initialization of IDownloaderService, the following methods are used in the application for updating the progress and performing other download operations.

- *requestPauseDownload() – to pause the download*
- requestContinueDownload() – to resume download
- *onDownloadProgress(DownloadProgressInfo progress)* – returns download progress

Once the files are downloaded successfully, the application has to read the files.

Reading the expansion Files

Reading the files can be achieved in several ways. It does not matter how the application performs expansion files reading, it must perform an external storage check before reading. This check makes sure that the SD Card is present in the device and the user has not removed it

Figure 57 shows how to initialize the path variables and read the expansion files:

```
// The shared path to all app expansion files
private final static String EXP_PATH = "/Android/obb/";

static String[] getAPKExpansionFiles(Context ctx, int mainVersion, int
patchVersion) {
    String packageName = ctx.getPackageName();
    Vector<String> ret = new Vector<String>();
    if
(Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
        // Build the full path to the app's expansion files
        File root = Environment.getExternalStorageDirectory();
        File expPath = new File(root.toString() + EXP_PATH + packageName);

        // Check that expansion file path exists
        if (expPath.exists()) {
            if ( mainVersion > 0 ) {
                String strMainPath = expPath + File.separator + "main." +
                        mainVersion + "." + packageName + ".obb";
                File main = new File(strMainPath);
                if ( main.isFile() ) {
                        ret.add(strMainPath);
                }
            }
            if ( patchVersion > 0 ) {
                String strPatchPath = expPath + File.separator + "patch." +
                        mainVersion + "." + packageName + ".obb";
                File main = new File(strPatchPath);
                if ( main.isFile() ) {
                        ret.add(strPatchPath);
                }
            }
        }
    }
    String[] retArray = new String[ret.size()];
    ret.toArray(retArray);
    return retArray;

}
```

Figure 57 - Reading the Expansion Files

In-case of reading the file without extracting it, this can be achieved by using the APK *Expansion Zip Library* as shown in Figure 58.

```
// Get a ZipResourceFile representing a merger of both the main and patch
files
ZipResourceFile expansionFile =
APKExpansionSupport.getAPKExpansionZipFile(appContext,
        mainVersion, patchVersion);

// Get an input stream for a known file inside the expansion file ZIPs
InputStream fileStream = expansionFile.getInputStream(pathToFileInsideZip);
```

Figure 58 - Reading the Expansion Files from Zip

8.3 Conclusion

This chapter explained the procedure of publishing Android application on the Google Play Store. One of the major challenges involved in this process was to handle the implementation of expansion files.

Benefits of using expansion files give ability to the application to update without re-downloading all of the original assets. Because Google Play allows the developer to provide two expansion files with each APK, the patch file can be used for providing updates and new assets. Doing so avoids the need to re-download the main expansion file which could be large and expensive for users.

The patch expansion file is technically the same as the main expansion file and neither the Android system nor Google Play perform actual patching between the main and patch expansion files. The application code must perform any necessary patches itself.

As long as the main expansion file associated with the APK is not changed in the Developer Console, users who previously installed your application will not download the main expansion file. Existing users receive only the updated APK and the new patch expansion file (retaining the previous main expansion file).

The next chapter concludes this report and presents the overall work completed in this project. Several new features which can be added to this application are discussed in the future section of next chapter.

Chapter 9 | **Conclusion**

This project successfully delivers a user interactive Android mobile application for Timespan museum. The purpose of the application is to provide the user with a virtual tour about *Highland Clearances* [1] using an interactive book, audio narratives and a historical map. The project evaluates Timespan's existing iOS App and extends it with added features for Android. Table 3 displays a summary of major improvements achieved in this project. It is worth mentioning that the source code for iOS was unavailable and this project had to achieve the similar application behaviour without it.

Feature	iOS	Android
Reusable Book (*PDF*)	X	✓
Text Zoom	X	✓
Text Search	X	✓
Reusable Map (*MBTiles*)	✓	✓
Map Zoom	✓	✓
Map Location Pins	✓	✓
Full Screen photo gallery	✓	✓
Audio Playback	✓	✓
Audio Playback with App Usage	X	✓
Landscape Orientation Support	X	✓

Table 3- Features Comparison between Timespan iOS and Android application

The application is ready to be published on the Google Play Store. The feedback of the client is very positive and according to the Timespan's Director: (see Appendix C):

> *"The product fully meets our requirements, and we a more than happy with the outcome of the project. As discussed, the full application could not be realised due to the given timeframe; nevertheless, the student found elegant ways of reducing content such as adapting the menu or replacing content. The new product contains improvements such as added functionality in the story and the trail."*

iOS and Android smartphones now cover more than 88% of the smartphone market (see Figure 6). Therefore, the potential user base for Timespan is very broad and provides room for significant improvement in marketing and extending audience. Timespan has the potential to promote further awareness about Scottish History.

9.1 Future Work

- **Support for Tablets**

This application can be extended to support iPad and Android Tablets. Although the iOS application can work on iPad and Android application can work on Android tablets, the user interface is not designed for large screen tablets. This can be achieved by creating user interface designs for large screens and integrating it into the existing code.

- **Application Updates**

A server side implementation can provide content updates to the application without downloading the application again. This can be achieved by keeping the existing offline features and then connecting the application with a server. The users can also provide feedback/suggestions using the application which can be stored on the server.

- **Merchandise Shop**

People visiting museums often buy souvenirs such as drawings and postcards. The application can implement a merchandise shop feature from where users can buy online. The server implementation is required for this feature to perform the transaction and inventory management tasks. The mobile application will provide the items catalogue on the device.

- **Integrate Unity 3D**

A more advanced feature of 3D Virtual World can be integrated using Unity3D. It is a cross-platform game engine used for developing video games for web, desktop, console and mobile platforms. It can also be used to deploy cross-platform 3D model objects. According to Unity3D's press release, it has one million registered developers[41]. By integrating Unity 3D, the museum application can provide 3D Virtual World experience to smartphone and tablets users.

References

[1] J. Prebble, *The Highland Clearances*. Penguin Books, London, 1963, pp. 60-61.

[2] Bluemungus, “Timespan: Museum Without Walls App (Scotland’s Clearances Story),” 2012. [Online]. Available: http://www.bluemungus.com/projects/timespan-museum-without-walls/.

[3] C. . Rahul Raj, “A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach,” *2012 Annual IEEE India Conference (INDICON)*, pp. 625-629, Dec. 2012.

[4] J. Kim, “Architectural patterns for service-based mobile applications,” *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pp. 1-4, Dec. 2010.

[5] K. Andersson and D. Johansson, “Mobile e-services using HTML5,” *37th Annual IEEE Conference on Local Computer Networks -- Workshops*, pp. 814-819, Oct. 2012.

[6] Z. Benenson, F. Gassmann, and L. Reinfelder, “Android and iOS Users ’ Differences concerning Security and Privacy,” pp. 817-822, 2013.

[7] L. Corral, A. Sillitti, A. Garibbo, and P. Ramella, “Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm,” pp. 181-183.

[8] P. Smutný, “Mobile development tools and cross-platform solutions,” pp. 653-656, 2012.

[9] M. Palmieri, I. Singh, and A. Cicchetti, “Comparison of cross-platform mobile development tools,” *2012 16th International Conference on Intelligence in Next Generation Networks*, pp. 179-186, Oct. 2012.

[10] G. han, E.; Baciu, “Appendix A: Starting the iOS SDK,” in *Introduction to Wireless Localization:With iPhone SDK Examples*, Wiley-IEEE Press, 2012, pp. 245 -259.

[11] S. C. Lauren Darcey, *Introducing Android Development with Ice Cream Sandwich*. Pearson Education, 2011.

[12] A. Rizk, *Beginning BlackBerry Development*. APRESS, 250AD, p. 2009.

[13] D. J. Tom Miller (Author), *Developing for Windows Phone 7 and Xbox 360*, 1st ed. Addison Wesley, 2010, p. 528.

[14] C.-marius Grigorescu and S.-aurel Moraru, “Industrial Software Monitoring System Extension for Mobile Devices Based on GlassFish and PhoneGap,” no. Epe, pp. 25-27, 2012.

[15] L.-L. Chen and Z.-L. Liu, “Design of Rich Client Web Architecture Based on HTML5,” *2012 Fourth International Conference on Computational and Information Sciences*, pp. 1009-1012, Aug. 2012.

[16] P. E. Dickson, “Cabana : A Cross-platform Mobile Development System,” pp. 529-534, 2012.

[17] S. Kaltofen, M. Milrad, and A. Kurti, “A Cross-Platform Software System to Create and Deploy Mobile Mashups,” pp. 518-521, 2010.

[18] R. Zioma, “Unity : iOS and Android - Cross Platform Challenges and Solutions,” p. 4503, 2012.

[19] A. Puder and O. Antebi, “Cross-Compiling Android Applications to iOS and Windows Phone 7,” *Mobile Networks and Applications*, vol. 18, no. 1, pp. 3-21, May 2012.

[20] I. Papapanagiotou, “Smartphones vs . Laptops : Comparing Web Browsing Behavior and the Implications for Caching,” pp. 423-424.

[21] T. Basic and S. Loop, “Graphics Performance in Rich Internet Applications,” 2012.

[22] D. Sambasivan, N. John, S. Udayakumar, and R. Gupta, “Generic framework for mobile application development,” *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, pp. 1-5, Nov. 2011.

[23] D. Sin, E. Lawson, and K. Kannoorpatti, “Mobile Web Apps - The Non-programmer’s Alternative to Native Applications,” *2012 5th International Conference on Human System Interactions*, pp. 8-15, Jun. 2012.

[24] H. Reza and N. Mazumder, “A Secure Software Architecture for Mobile Computing,” *2012 Ninth International Conference on Information Technology - New Generations*, pp. 566-571, Apr. 2012.

[25] D. Pavlić, M. Pavlić, and V. Jovanović, “Future of Internet Technologies,” pp. 1366-1371, 2012.

[26] I. D. Corporation, “IDC - Press Release Feb 2013 Smartphones,” 2013.

[27] ComScore, “ComScore Press Release Dec 2012,” 2012.

[28] I. D. Corporation, “IDC - Press Release Dec 2012 Tablets,” 2012.

[29] D. Melanson, “Google now at 1.5 million Android activations per day,” *Engadget*, 2013. [Online]. Available: http://www.engadget.com/2013/04/16/eric-schmidt-google-now-at-1-5-million-android-activations-per/.

[30] Google, “Android OS Statistics,” 2013. [Online]. Available: http://developer.android.com/about/dashboards/index.html.

[31] E. Chin, A. P. Felt, V. Sekar, and D. Wagner, “Measuring user confidence in smartphone security and privacy,” *Proceedings of the Eighth Symposium on Usable Privacy and Security - SOUPS ’12*, no. 1, p. 1, 2012.

[32] H.-ching Chang, V. Tech, and K. W. Cameron, “Energy-Aware Computing for Android Platforms.”

[33] F. Ding, F. Xia, W. Zhang, X. Zhao, and C. Ma, “Monitoring Energy Consumption of Smartphones,” *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pp. 610-613, Oct. 2011.

[34] J. K. Lee, “Android Programming Techniques for Improving Performance - I L,” *Memory*.

[35] R. Harrison, D. Flood, and D. Duce, “Usability of mobile applications: literature review and rationale for a new usability model,” *Journal of Interaction Science*, vol. 1, no. 1, p. 1, 2013.

[36] Apple, “iOS Human Interface Guidelines,” 2013. [Online]. Available: http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html.

[37] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2010, p. 792.

[38] D. Balasubramaniam, “Architectural Styles & Patterns - Lecture Notes - CS5033 Software Architecture.” 2013.

[39] R. Meier, *Professional Android 4 Application Development*, 3rd ed. Wrox, 2012, p. 864.

[40] P. Boctor, “AppCrush,” 2011. [Online]. Available: http://github.com/boctor/idev-recipes/Utilities/appcrush.

[41] Unity3D, “Fast Facts,” 2013. [Online]. Available: http://unity3d.com/company/public-relations/.

Appendix A – External Libraries

- MuPDF

http://www.mupdf.com/doc/how-to-build-mupdf-for-android

- OSMDroid

https://code.google.com/p/osmdroid/

- OSMDroidBonusPack

https://code.google.com/p/osmbonuspack/

- SL4J

http://www.slf4j.org/android/

- JSoup

http://jsoup.org/download

Appendix B - Compiling the Source Code

1. First of all make sure you have all the libraries mentioned in Appendix A.
2. Import the source code in eclipse as an Android project.
3. Now from command line go to project's */jni* folder.
4. Run the command *ndk-build.*
5. Go back to eclipse and run the Android Application.
6. Select emulator or device

Appendix C – Client Feedback

Dear Alan,

I would like to thank you for initiating the Android app development of timespan's clearance trail app, as undertaken by your master student Mujtaba.

The product fully meets our requirements, and we a more than happy with the outcome of the project. As discussed, the full application could not be realised due to the given timeframe; nevertheless, the student found elegant ways of reducing content such as adapting the menu or replacing content. The new product contains improvements such as added functionality in the story and the trail.

There has been a high demand for an android version of the app and we are excited to be able to offer our visitors this new product.

With best wishes,

Anna Vermehren,

Development Director,
Timespan
Email: director@timespan.org.uk