INFORME@DUBLIN AN ANDROID APPLICATION THAT SHOWS THE HISTORY OF DUBLIN AND SURROUNDING AREAS

By

Adam O'Connor

Supervisor(s): Dr. Simon McLoughlin

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BSC (HONS) IN COMPUTING
AT
INSTITUTE OF TECHNOLOGY BLANCHARDSTOWN
DUBLIN, IRELAND
17 MAY 2017

Declaration

I herby certify that this material, which I now submit for assessment on the programme of study leading to the award of **BSc (Hons) in Computing** in the Institute of Technology Blanchardstown, is entirely my own work except where otherwise stated, and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfillment of the requirements of that stated above.

Dated: 17 May 2017

Author: Adam O'Connor

Abstract

The project in hand takes on the role of a software developer which is a computer science student at the Institute of Technology Blanchardstown. The proposer of the project puts into focus, that historical monuments which are located throughout the suburb's and city of Dublin have been neglected over some time on which the developer wishes to change. Bringing the past and present together as one is what the application is trying to achieve as well as educating local people about their area.

The proposer of the project is willing to design and implement a location-aware Android application which provides Geofence locations of specific historical monuments. On entering such a monument, the Information will then be presented to the user of such device. The language of which the app would be coded will be Java on which the developer has learned over numerous years.

The following application would provide an inexpensive way of displaying historical information to patrons of the app and other tourists of which wish to learn about Dublin. The application could change the way locals and tourists learn about the surrounding area of Dublin. The developer will contact Dublin county council and Fingal county council to receive their feedback or funding for the rollout of such an application to the general public.

Acknowledgements

Throughout the project in question, the author has of great gratitude for Dr Simon McLoughlin for his coherence in the development of this application. With giving advice and the direction of which to take an application of this proportion. The author would also like to thank Dr Stephen Sheridan for providing the 4th year students with a room to complete these projects and getting the necessary public attention for the projects on which all of the 4th years computing students completed.

Table of Contents

List of Tables

List of Figures

Abbreviations

APP	Application
OS	Operating System
CSS	Cascading Style Sheet
GNU	GNU's Not Unix
IDE	Integrated development environment
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
LBS	Location Based Services
ADB	Android Debug Bridge
GCM	Google Cloud Messaging
WAH	Windows Azure Hypervisor
GPS	Global Positioning System

Chapter 1

Introduction

1.1 Background

From the beginning of time finding information about particular areas or building was a kind of tricky job unless you knew where to go or whom to talk to about the history attached to these certain places. More recently we can access websites and see the information about a certain location or area. Usually, the Dublin county council office has the data on each town for example Lucan. With the suburbs growing steadily over the last number of years the changing of towns has been in an extraordinary state. Towns that have been around for hundreds of years are the main reason for the development of such an application (App) so that Dublin's history can be brought back to life.

What the developer is willing to achieve is to allow people to navigate, explore and learn about the history of different buildings. Throughout the use of this application, the user's will then grasp a better knowledge of the area in which they live. Throughout the development of the Android Operating System (OS) starting in 2006 steadily numbers grew after Google's release of sharing its operating system to all mobile manufacturer's. The first mobile to be released with such an OS was the HTC Dream.Wilson (2008) which is why the developer has chosen to develop an android application. With the introduction of the Google play market, which launched on the March 6th, 2012, There has been a huge demand for smart technology. Now we can use our mobile phone for doing just about anything, such as surfing the web, reading emails, even playing games and the interpretation of books.

1.2 Introduction of the InforMe@Dublin Project

With regards to the following proposed project on which its aim is to provide additional information to tourists or just the common public who wish to learn more about the history of their local area and wish to view old photos or view historical places in their area. When the user is in this specific area pop-ups will appear that can be selected and information will then be display for that designated place. Users may post their findings of images whether that be past or present to allow others to learn from what users have posted.

The information such as the history of the area will be sourced from a variety of on-line sources that will be compiled by the developer. The information will be accessed by the application using Firebase database which is a cloud storage service for Android, OS and web projects of which was created by Google.

With the research that has been uncovered in the literature review of the following thesis. The developer has then chosen what technologies should be used in order to complete the InforMe@Dublin application.

1.3 Research Question's

The Research questions of which the developer must answer are defined below. Each of these questions will be answered throughout the following thesis.

1. ***Is there a need for an application like this on the marketplace?***
2. ***Is there a way to save battery life while using GPS / Internet connection within an android application?***
3. ***Will location services, and mobile internet take much battery or data?***
4. ***Will the APP provide a wiki type interface where visitors can add to the information about a historic site?***

Chapter 2

Project Management

2.1 Project Name / Supervisor / Manager

Project — InforMe@Dublin
Supervisor — Dr Simon McLoughlin
Manager — Adam O'Connor

2.2 Success Criteria

- The complete program should be able to locate their current location and if a building of significant historical value.
- The program should allow other users to learn and about their area.
- The program also allows the users to explore their town and get fit by doing so.
- The application should educate the user and tell them the information.
- Program should have great user functionality and work very efficiently for the user.
- Should be very user-friendly.

2.3 Assumptions Risks and Obstacles

- Coding a responsive application for the Android operating system.
- Using the Google Maps API and location service's that are incorporated into an android device.
- Using Android Studio to create this application.
- Coding from scratch and learning new technologies on the way.
- Allocating time and resources to develop the application as a whole.
- Timekeeping to allow us for the project to be completed within a reasonable time-scale.

2.4 Motivation for choosing project

With the influence of new mobile technologies throughout this day and age, the developer has always seen a niche market for an application that utilises and explores the user's town. Everyone has readily available internet connectivity on our mobile phones and if not a GPS module incorporated into that device. With towns growing rapidly throughout the years all the hidden gems of these area's are being left behind with no-one knowing the history of what is there.

A developers job is to either make things easier on people or to produce new idea's or technologies for a normal user to use. The only flaw of this project is the user needs to own an Android device and have an internet connection. The project enables the user to walk around their town and to discover new areas with the use of this app. The app should provide some Wikipedia type adjustments that will be sent to the admin or requests to show lost monuments or where historical relics have been found.

Due to some of the historical finds that have been found around Dublin and some of the structures with the likes of Neolithic axes found as well as the oldest bridge in Ireland that provides passage over the Griffeen Valley River.

2.5 Methodology

V Model The model that we will be using to develop this project will be the V Model, why the developer chose this model, is because the steps in progress are in a sequential manner in and the type in question is also known as the verification and validation model. Which means after each step is complete, the software and hardware are both Verified and validated. The model chosen is an augmentation of the waterfall model and is based on the testing of each phase of the development which means each step of the development life-cycle there is then a directly associated testing phase with each part of code. The type of proposed model, as follows, is a highly disciplined model in that the next step only starts on completion of the open phase. Point (2015)

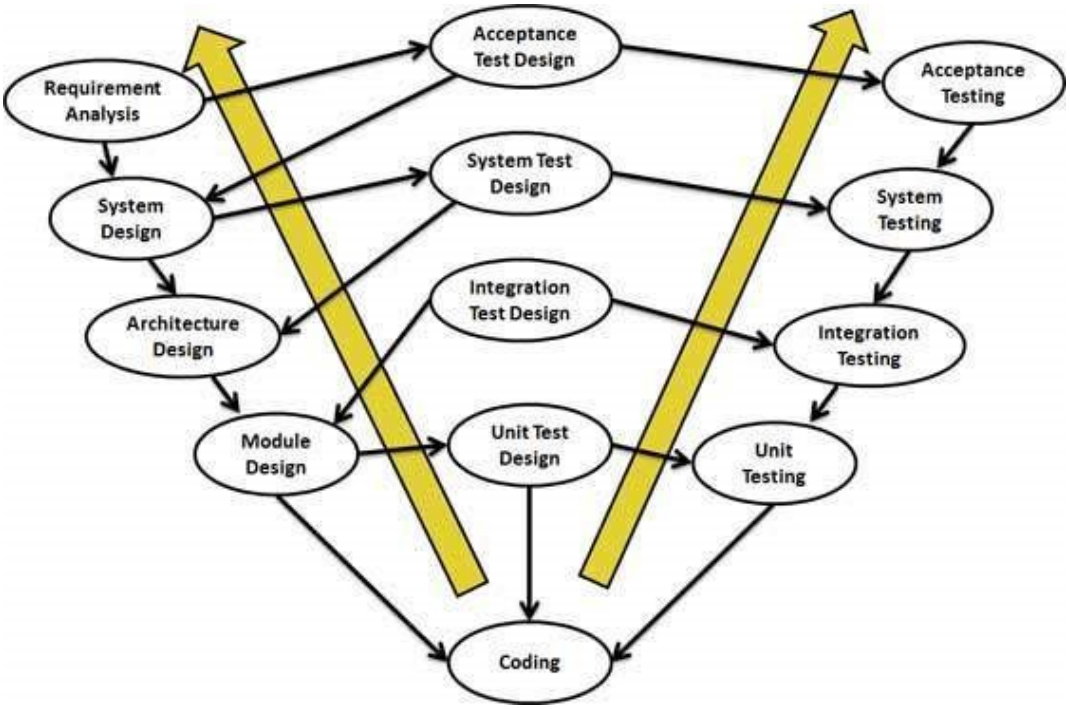

Figure 2.1: Image of the V Model (Point, 2015)

2.6 Work Breakdown Structure

Objective	Task
Complete Project Proposal.	Complete within 2 weeks.
Research and test Google maps API and geo-location.	Complete within 1 week.
Research android application development with the cloud.	Complete within 1 week.
Preform literature review on the project.	Complete within 2 weeks.
Risk analysis.	Complete within 2 weeks.
Development plan.	Complete within 1 week.
Build prototype.	Complete within 3 weeks.
Second risk analysis.	Complete within 1 week.
1st validation and verification.	Complete within 1 week.
Operational prototype.	Complete within 1 week.
2nd validation and verification.	Complete within 1 week.
Final development.	Complete within 2 weeks.
Implementation.	Complete within 1 week.

Table 2.1: Table of Work Breakdown Structure.

2.7 Technologies and Skills Needed

1. Programming in the Java development language.
2. Using and incorporating APIs into Android applications.
3. Knowledge of XML for design layouts.
4. Knowledge of JSON.
5. Knowledge of Firebase and its associated methods.
6. Experience of using Google's developer console.
7. Experience of developing application's for Android devices.
8. Using and adapting gradle properties of Android applications.
9. Underlying architecture of the Android OS.
10. Utilising modules of the Android Device.

11. Learning new permission requests of Android Marshmallow and above.

12. Utilising the battery life of Android devices.

13. Knowledge of Location Services.

14. Knowledge of Computational processes of Android devices.

2.8 Gantt Chart

Below is a following gantt chart with regards to the work of which will be completed by the developer throughout the development of such an application of this size.

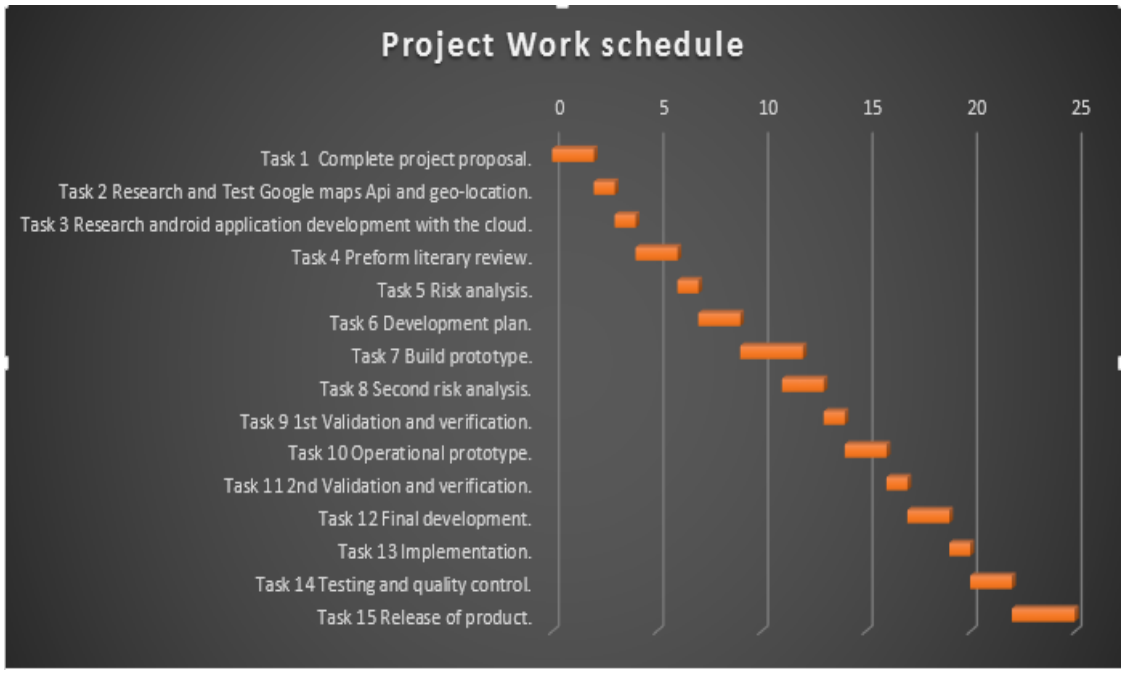

Figure 2.2: Gantt Chart

2.9 Development Device

Concerning the development section of the following thesis of which is about the device on which the Android application would be developed on. The device in question is as follows -

Mobile Name	Alcatel Pop 4+
Operating System	Android 6.0 (Marshmallow)
CPU	Quad-core 1.1 GHz Cortex-A7
Chipset	Qualcomm MSM8909 Snapdragon 210
RAM	1.5GB Ram
Internal Memory	16GB
Battery	Removable Li-Ion 2500 mAh battery

Table 2.2: Alcatel Pop 4+ Specifications.

Figure 2.3: Alcatel Pop 4+

2.10 Conclusion

This project as a whole is ready to start. Some changes may occur during the development throughout the build and production. The next stage of the project in the question is to research all available sources and technologies that can be used in the creation of such project. The Finished development of the application scheduled for the end of April 2017, with a mock-up build of the project due in December 2016.

Chapter 3

Literature Review

3.1 The New Era of Mobile Phone Technology

Regarding the mobile phone industry first starting off and then rapidly building over the past 49 years when the first cell phone that was produced by the Motorola company. That makes the very first hand-held modular phone created on the 3rd of April 1973, By Martin Cooper, he was also a Motorola researcher and executive who developed the only type of hand-held subscriber equipment. Martin placed the first call to Doctor Joel S. Engel of Bell Labs.

A funny side of this story was because Bell Labs and Motorola where in direct competition to develop this kind of technology. This mobile phone weighed 1.5 kilos on which the talk time was only 30 minutes and took 10 hours to charge its battery full. Cooper (2006)

Compare this to a mobile phone device to today's standards. The Samsung Galaxy S7 edge which weighs a biscuit crumbling 157 grammes which also holds a large talk time of 27 hours and a full charge in less than 1 hour. Put this together with a 1440 x 2560 pixel display compared to our counterpart the Motorola Dyna-Tac, which had no screen. Technology in the last 49 years has sky-rocketed with the introduction of screens and new ways to get portable and to connect with one another. Arena (2016)

3.2 The Rise of Android OS

First off what is the android operating system or who even created this OS? Well, Android was developed by Google for the use of mobile devices such as smartphones and tablets. It is an OS that's been available on devices made by a variety of manufacturers since 2006. Which has made the OS very popular among users throughout the world. This OS has given consumers a lot more choice with hence to device styling, and pricing put that together with customising our modular phone to whatever way the user would like, and it is the most used product in the world. Wireless (2016)

The Android OS has an underlying architecture with the Linux kernel as its base layer. Linux is an open-source operating system that is free under the GNU or General Public Licence. Google adopted the Linux kernel to fit their needs, and it gives the Android developers a pre-built already maintained operating system, so a lot of the work had already been completed pre-development. How-To-Geek (2014)

Android's creators were formally not a company of Google it was Android Inc, Rich Miner, Nick Sears, Chris White and Andy Rubin started this company's goal was to produce ***"smarter mobile devices that are more aware of its owner's location and preferences"***. Eligin (2005)

Due to its creation, the first commercial version of the Android OS released on the 22nd of October 2008 running on the HTC Dream. Wilson (2008) Each new version of the Android OS are set in alphabetic order which started with version 1.5 "Cupcake" followed by 1.6 "Donut". With the help of the development of Android versions, the most up to date at version this time is 7.0 Nougat released on August 22nd, 2016 which has a distribution level of less than 0.1%.

3.3 A Look at Smartphone Frameworks

Due to the Research of many proposed frameworks the developer has been looking at the following:

1. Corona SDK, Corona is designed to enable a fast development environment, to code

along with their rich APIs which makes adding complex features easy to produce. Also, the work-flow lets us see these changes instantly. The development in Lua which seems to be an easy to learn development language. Williams (2013)

2. PhoneGap, The framework sponsored by Adobe at its 6.0 version used for building applications with the use of HTML5, Cascading Style Sheet (CSS) and JavaScript Development. This framework now supports Windows phone support. Cordova also has an added WebView as this make's it easier to incorporate into larger applications.Myer (2011)

3. Xamarin, All code can be written in C# and deployed within Android for developing apps and games with this framework. This framework also works for IOS devices as well as Windows. Testing of applications is allowed through the cloud with timely monitoring of the application. Dickson (2013)

4. Google Service's framework, Google Services Framework provides support for Google APIs, which includes Google Cloud Messaging (GCM), Location, Maps If an application that needs this framework is not installed on a device on which the developer has developed the application in question will crash.Fogl et al. (2016)

3.4 The Creation of a New IDE

The official (IDE) for the Android operating system is Android Studio. Which Google announced on May 16th, 2013 at the Google I/O conference. Android Studio is now a free source of development platform which is available under the Apache Licence 2.0. Android Studio was in an early access preview stage from version 0.1 which was at the beginning of May 2013. Developers (2015)

Android Studio is based on JetBrains IntelliJ IDEA software, which is designed specifically just for Android application development.Studio (2016)

With this development studio its a great advantage to any programmer or developer that's out there. Within the integration of such an IDE is the new preview layout manager being able to see our screen design for the application. Preloaded and in all perfection the IDE

Figure 3.1: Android Studio Logo (Studio, 2016)

runs developed applications straight to an Android phone with the help of the Gradle compiling the code as well with the Android Debug Bridge (ADB) to run applications fast on an Android Device.

With all of this action packed coding magnificence Android Studio now has an integrated Dex process with enables programs being built to run faster than ever before but needs a little modification to the gradle properties file to increase the standard ram memory available to two gigabytes. Also with this in mind instant run is now a thing which is now available on the latest release of the Android Studio 2.1 - 2.2.

3.5 Moving to the Cloud

3.5.1 Firebase

With Google's style of cloud, access is of somewhat technique-specific with a sandbox style. Its services are quite good, Due to the usage of Android Studio incorporating this service into the development of Android applications is somewhat quite straight forward. Well that is if all the little parts of the coding are correct then its connection to the cloud is phenomenal. Qian et al. (2009)

Firebase is a cloud service provider which is used as a back-end service. What's incorporated within these service's allows an application to work hand in hand with the database across a network in real-time which allow's for the likes of a developer's to store and sync data across multiple clients. Metz (2012)

3.5.2 Microsoft Azure

With Azure being announced in October 2008 and then finally released on the 1st of February 2010 as Microsoft Azure. The service which uses Windows Azure Hypervisor or (WAH) is used as the underlying cloud infrastructure and .NET as the application container. Qian et al. (2009)

Its licence for use is mostly a closed source platform with open-source modules for client SDK's which is not entirely bad but still most developers are looking for open-source content Microsoft has many services is that customers have to pay for its services. With that in mind at least it provide's both Paas and Laas. Martain (2014).

3.6 Utilising Google's Geographical Mapping Systems

The Google Maps API, in reality, is just a web-based mapping service which is provided by Google, free to developers that doesn't cost money unless the developer is selling that application to the public. The service provides a slick and highly responsive graphical interface which was built using AJAX technologies which is the name for Asynchronous JavaScript + XML. This service produce's detailed street and aerial imagery information. The following API allows customization of the map which can also use application specific data that has been created by the developer.

Google Maps API is formally based on simple classes. With Google Map's and Location Based Service's (LBS) use's information services based on the current or a known location, which is supported by the electronic map platform. Ren and Dunham (2000) Such projects that will use these services are the like's of A Tourist Guide project, Cheverst et al. (2000), A Pinpoint Tourist Guide, Wan and Alagar (2006), and a PinPoint Tourist Guide, Codd (1972).

Together with the Google Maps API, is used on the Android OS, developers can create better apps with the usage of map fragments which allows for better computational adjustments to load the map and on displaying the user's location in real time. Thus throughout the usage of this, we can change the different styling needs of the application by using custom markers, indoor maps, polylines and numerous other styling options. The only additional

adjustments that are necessary for the application to run with the loading of the map and additional components is the developer's console key for the API, and each API used during the development of applications needs a specific key for that API to run. Google's API's are usually free to use under educational needs with the like's of free applications where the developer will not gain anything from it, or the number of user's that downloaded the application is a small number. Sayed et al. (2005)

3.7 Acquiring Location Based Services

Throughout usage of location service classes that Google provides for its developers. It is of an exceptional nature, knowing where the user is, allows for a greater understanding of the user's area. Which allows the app to become a smarter and more diverse project which will deliver better on the go information for the user. Hence developing location-aware applications in this day and age is so sought after. Together with the Android OS and utilising the Global Positioning System (GPS) location module of the device. The network location provider allows developers to acquire the site of the user's mobile device. With this in mind though although GPS is, in theory, more accurate, it only works outdoors and uses a lot more battery life, also it does not return the user's location back as fast as we would all like. For a more rapid way of utilising these classes, of which Google provides to the developers of the world. Developers can use cellular towers and wifi signal to acquire the location of one's device. Thus providing a way of obtaining the specific area in two ways instead of just one. Ferraro and Aktihanoglu (2011)

Table 3.1: Battery Usage of Positioning Techniques Bareth and Kupper (2011)

Technologies	*Accuracy*	*Precision*	*Energy*
GPS	10m	95%	6.616Ws
WiFi	50m	90%	2.852Ws
Cellular-ID	5km	65%	1.013Ws

With the information stated there could be complications in getting the location of the device, some reasons include, The multitude of location sources which could be of somewhat

coherent between GPS, Cellular identification, mobile internet connection, or WI-FI can each provide an indecisive clue to where the user's whereabouts is. Determining which to use and to trust is a matter of trade-offs between accuracy, battery efficiency and don't forget speed. Sayed et al. (2005)

Another section to look at during the LBS is the user's movement as the location changes the developer must account for movement by re-estimating the user's location after a certain amount of time. Retrieving each device's location services differs on how each method is called, and there is never a correct time for the calling the position or retrieving the location of the device.

Due to the variance of location estimates produced from each location source are never consistent in their accuracy because locations can change so much in ten seconds. From source to source of location strategies one might be more accurate than the newest location from another or the same source. With these problems, this can make it quite difficult to obtain a reliable user location reading.

3.8 Details in Inquiring the Location of a Device

The methods for providing the geographical location of a user by just utilising the device that they are using. This is done without the client's approval or request. The mobile device gathers data from the locality of the apparatus which gains the location of the user. Based on virtual mapping using location based resources the device is then fundamentally capable of gaining its location using its integrated GPS. The device then uses its location information and compares it to available location based resources within the mobile apparatus. Rankin and Griffiths (2005)

the figure below assembles of how a location develops from a device. The handheld device communicates with the device's network from the port. The connection may only be possible when the mobile connected to the device's network from a non-permanent link. A link could be a WiFi connection, mobile internet connection. The mobile device is now the same as a personal computer or other mobile or portable device's which can periodically connect to the web. The mobile device includes a processor which then executes program

instructions such of that the app that the developer is willing to produce which is held in memory. The memory includes random access memory (RAM) for program execution and non-volatile memory for storage of the Android programs or apps and systems data. The mobile device also includes integrated components such as a touchscreen, speaker, proximity sensor, GPS, microphone. New Android devices or as such mostly all incorporate all of these extra's and some other's even more which a Includes location positioning. Alizadeh-Shabdiz et al. (2007)

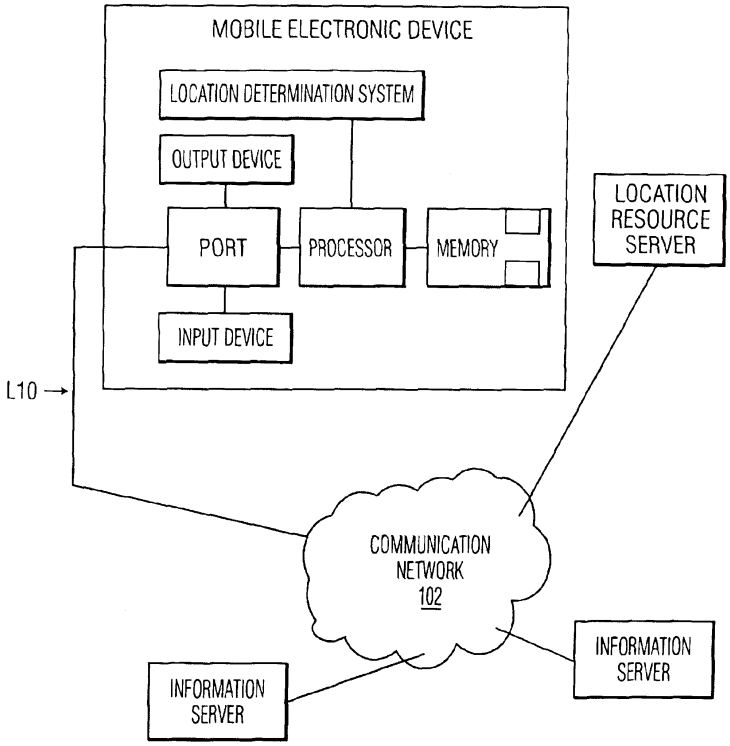

Figure 3.2: Diagram inquiring location (Rankin and Griffiths, 2005)

With this measure, the device can determine its current geographic location. A well-known method to provide the location determination function is to use a GPS receiver which is readily available in these phones, which can receive satellite signals to determine a location to within 10 meters of the user's location which has gotten better over the last number of

years. GPS is the first kind of a self-contained location system. Another self-contained location system is a system which receives signals from nearby radio towers which are accurate to within 5km. The last is the internet connection to that device which can either be WiFi or mobile internet provided by the provider which is correct to within 50 meters.

The strategy provides the best kind of accuracy to battery life the ratio seems to be just correct compared to just using GPS which uses an astonishing 3.764Ws more than GPS by just using a WIFI network. During this project, we must use all three location strategies to enable the battery of the device to last the longest. By reviewing our previous research, this path seems. By examining our previous research, this route appears to be the correct path to take in comparison to using one strategy as this might inhibit our use of gaining the location of the device.Bareth and Kupper (2011)

3.9 Literature Conclusion

Throughout this literature review process of gathering information from residual resources from different and also related fields. Such as the invention of the handheld modular phone, location strategies, cloud based services, and other Android applications that are on the same scale on which the developer is willing to produce incoherence with his Bsc(Hons) degree in computing. Due to the limited amount of time available for this project as follows, also with a small spectrum of fields that were discussed. It was somewhat difficult to find related projects on the same topic as the application itself. Which somewhat inhibited a wide-scale research on previous works associated with the project in question.

The concept of such project as a location-based historical, educational tool is rare as not many people would think of making applications like these, especially for a specific town. In recent years these applications are what clients need and want, also with new technology flowing through the markets smartphones are not a thing of the future they are readily available and don't cost that much to purchase.

Together with the methods of location strategies with respects to battery life and internet postage. The application is being developed at the correct time, As batteries would not last as long as they do now because of the cost of the processing power together with the integrated

components of that mobile device. The code is now smarter in ways of allowing less idle time and microprocessors in these devices use a lot less power with more speed.

Of course, we can look at some of the many difficulties and concerns about mobile applications with the advances of malware with respects to privacy and reliability, but this field is growing and getting better with each day with the development of connecting everyone and their data together. Which is why when coding app's users have to accept what the app is trying to access specifically such as storage, location, wifi.

Together with the help of the IDE which contributes to helping the developer on coding the different classes of the app, and which also contributes to helping the developer in coding with the likes of the Google map API and firebase. Each needs a specific key to the API that is used which was expressed throughout the research of the project.

The firebase cloud service helps the developer in managing user's and sending data throughout the system on updating the area's that are needed to display the information. With this in mind, databases can be easily updated and sent throughout the client base when that specific user logs in so that a larger update of the system is not needed on the google play store unless other graphical displays are in need of updating.

Chapter 4

Methodology Chapter

4.1 Appropriateness of the Research Design

The following project was based on both a quantitative and qualitative approach to research techniques. Together with the amount of investigating completed collectively with this project before it had begun and throughout the developer received many high words of praise for an application of this type throughout the small and large suburbs of the Dublin area.

4.2 Pilot Study of Proposed Project

A pilot study of the application was first introduced to friends and family to get a sense of feel for if the application would be a good idea and to see if this is something that the people of Dublin would want to see coming to their mobile phones. Soon after getting a great response from family the developer chooses to produce a survey based on the people of Lucan. Based on the application and what if any specifications they would like to see. The developer received a response of over 150 people completing the survey on survey monkey. The application was indeed a certain choice for the developer to go ahead. Through wire-framing each screen that would be used throughout the application the developer had gotten an overall feel for what the end product should be.

4.3 Ethical Considerations

Each part of the project that has been carried through is deemed to be the developer's work unless stated otherwise. Other factors may contribute to the writing of this formal document and the project as a whole. Information that has to be gathered for the project as follows will be referenced appropriately. Such as that the monuments within the application itself will not be information that the developer has written but information that has been gathered from a third party. Such as an internet source which will be verified.

4.4 Validity

The data sourced for the project is to the developer's knowledge correct. Multiple sources have been read through and verified such data that is being used. Articles such as journals and other particles of information were referenced throughout to the original authors that the information was taken from or sourced.

4.5 InforMe@Dublin Survey Results

Regarding the following chapter which discussed the methodologies of why the developer chose to develop an Android application for the intended use on patrons learning about historical monuments in their area's. The results are as follows. The following survey was completed using survey monkey.

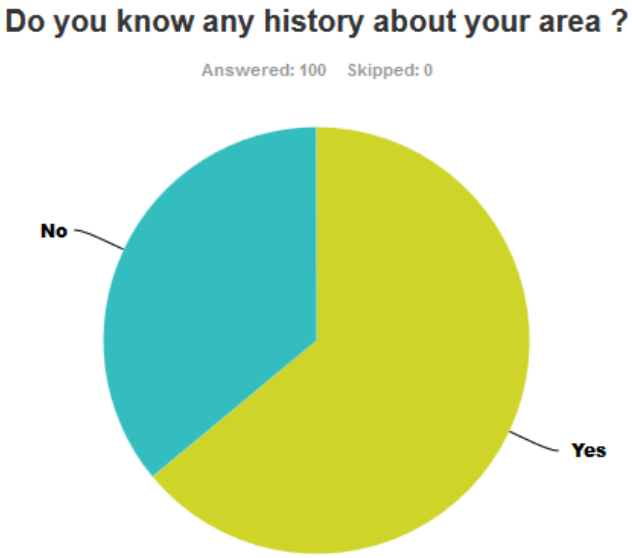

Figure 4.1: History Survey

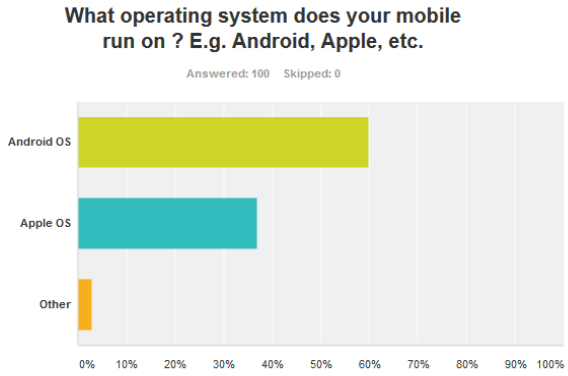

Figure 4.2: History Survey

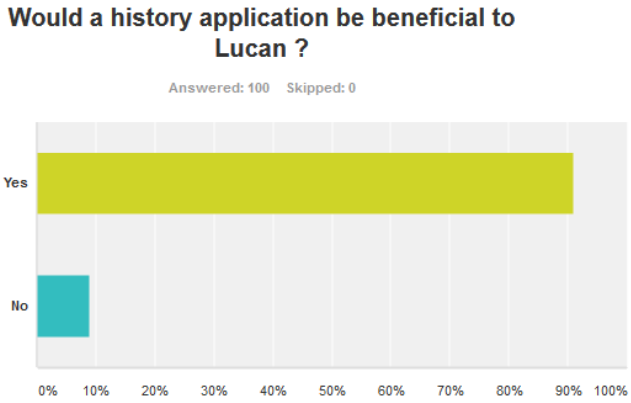

Figure 4.3: History Survey

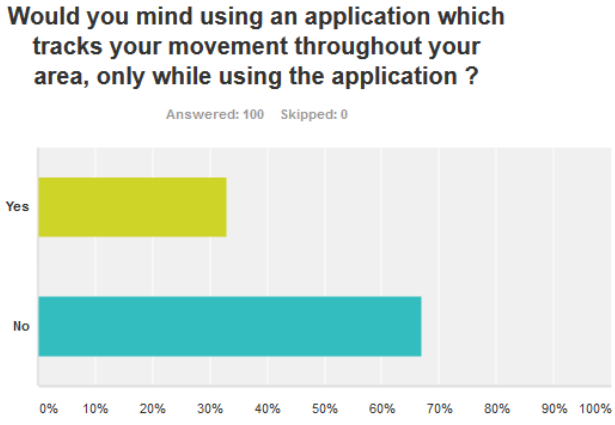

Figure 4.4: History Survey

4.6 Summary

The following chapter of which, concluded the research that has gone into choosing such a project in coherence with the BSc(Honours) Degree programme in the Institute of Technology Blanchardstown. The following was to see if the application that the developer had invented was deemed appropriate and needed in the area of Dublin. With the facts stated and nothing of the sort available online from other countries or in fact Ireland. The projects seemed plausible if the developer could, in fact, produce something of the application in mind.

Chapter 5

System Requirements and Specification

5.1 Description

Throughout the following chapter, the system designs of the project will be represented as diagrams. Using use-case diagrams, Activity diagrams and sequence diagrams with descriptions of these such diagrams displayed throughout the chapter as an aid in understanding the communications between each specific component of the app. These are needed for the reader to be able to comprehend the app as a whole.

5.2 Underlying Architecture of the Android OS

Regarding the underlying architecture on which an application will be in fact lying on top of the Android operating system. The following operating system is, in fact, a stack of software components which are mapped in coordinance with the corresponding hardware modules at the Linux kernel layer. The software stack is roughly divided into five specific sections and four different layers of which the frameworks, libraries, kernel and applications sit. The following diagram in figure 5.1 shows this.

Concerning the Linux layer of the following OS. The version of which is Linux 3.6 on which has approximately 115 patches to its name. This layer acts as a level of abstraction between the devices software modules and its hardware components on which contains the following, camera module, keypad, display, GPS and a wifi module, etc. Utilising the fact

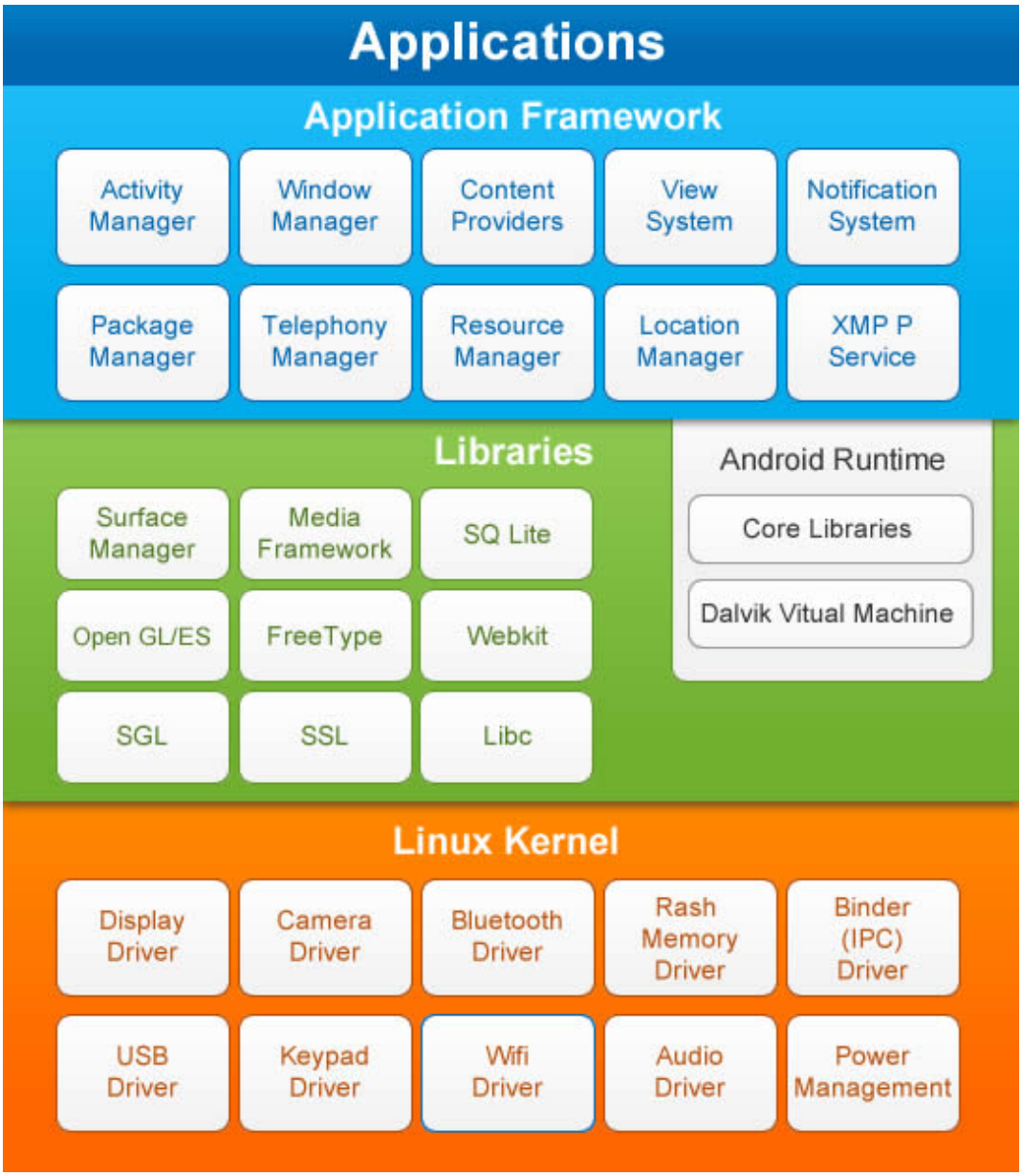

Figure 5.1: Android Architecture (JavatPoint.com, 2013)

that Linux is a great source for networking and on which supports a vast array of drives which provided a solid base on providing a somewhat reliable OS.

5.3 Android's Incorporated libraries

Concerning the libraries of which are incorporated within the application these apps are used to retrieve information on which may be related to specific applications on the device. On top of the Linux kernel, there are a set of libraries which includes an open source web engine WebKit, libc which is a standard C library which was created by Google for its Android OS and also an SQLite database of which is a useful repository for storage and sharing of application data throughout the program. Other libraries also include SSL which is responsible for the security of the device with regards to internet connectivity. Other libraries are also included to record and play audio and video.

5.4 A Look at Android's Location Service's

Utilising a client's mobile device on gaining their exact location can be done in two ways one of which is favoured over the other. The first way of allocating the user's location can be done by utilising the GPS module of the device by using Android's built-in Location API's which have been available since Android's first launch. These location gathering API's still work even to this day. However, uses a lot of the device's power to gain specific coordinates. With the following in mind, the user must more likely be outdoor's for the following to work to get a signal to a GPS satellite.

Google Play Services - The developer of the application InforMe@Dublin has utilised a new way of gaining clients geographical coordinates by way of using Google Play Services of which provides a very powerful, high-level framework to work from. This provides a choice of location gathering tools, such as WIFI and internet connection, GPS modules, cellular fixation or to bundle these together to provide a more accurate location fixture. The following API allocates power management of the device to keep each mobile phone at the top of its game.

Android gives applications of which access location services supported by the device through classes in the android.location package. The most important component of the following framework is the LocationManager which is a system service related to figure 5.1. The following provides APIs for determining the location of which the device is in at that current time. The digram located in figure 5.2 shows how the location service and manager work together with the system in order to retrieve the devices location.

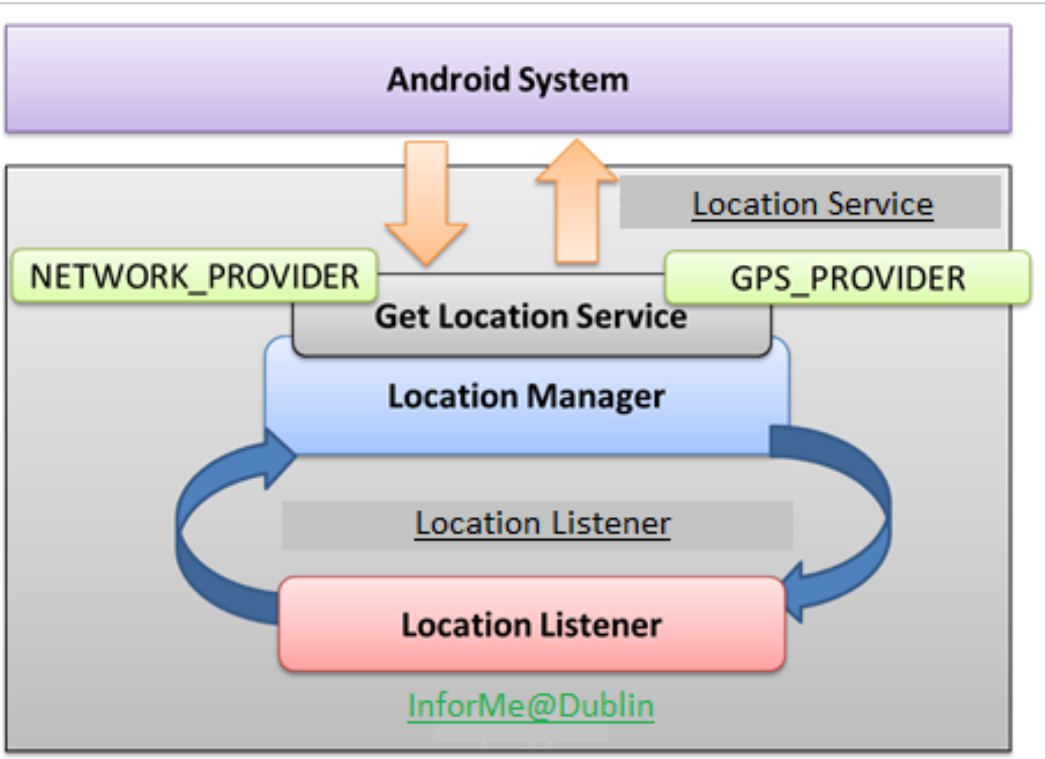

Figure 5.2: Retrieving Location

5.5 Installation of InforMe@Dublin

Before the user accessing the following application, the app may first be downloaded from the Google play store on which the developer has chosen to release to the general public. The application would be available or work for people living in the Dublin area. When the application has been downloaded and of the installation being completed the following

diagrams will show how the system works.

5.6 Login and Authentication Diagrams

The following diagrams represent the application's login page as well as some various activities on which the user may enter before moving forward while using the application.

5.6.1 Login Use Case Diagram

Regarding the Use Case diagram in figure - 5.3, The reader can grasp the associated activities that are compiled within the login page. When first starting the InforMe@Dublin application the user may only use the following activities to gain access to the map area of that app. Which are Google sign in, Creation of an account, forgotten password of an existing account or to log in with the user's credentials if account already created.

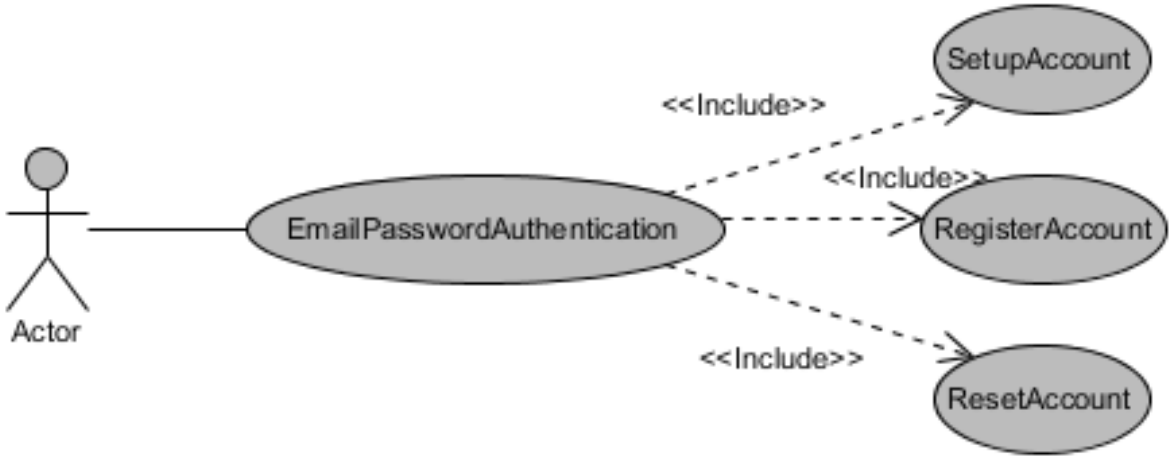

Figure 5.3: Login Use case Diagram

5.6.2 Login Activity Diagram

Regarding the following Activity Diagram in figure - 5.4, The developer has produced an activity diagram displaying how each component works with one another. As the following classes and activities are seen is how the source code is laid out. Each specific part of

the Android app has its specific package for the purpose of modularity. While visualising the diagram the reader can distinguish how each of the specific button clicks are mapped. Each specific activity has its very own methods on which the user must specifically progress through to gain access to the signed in activities these specific buttons and menus are hidden until the client has been authenticated by firebase's authentication method which is located in the EmailPasswordAuthentication activity.

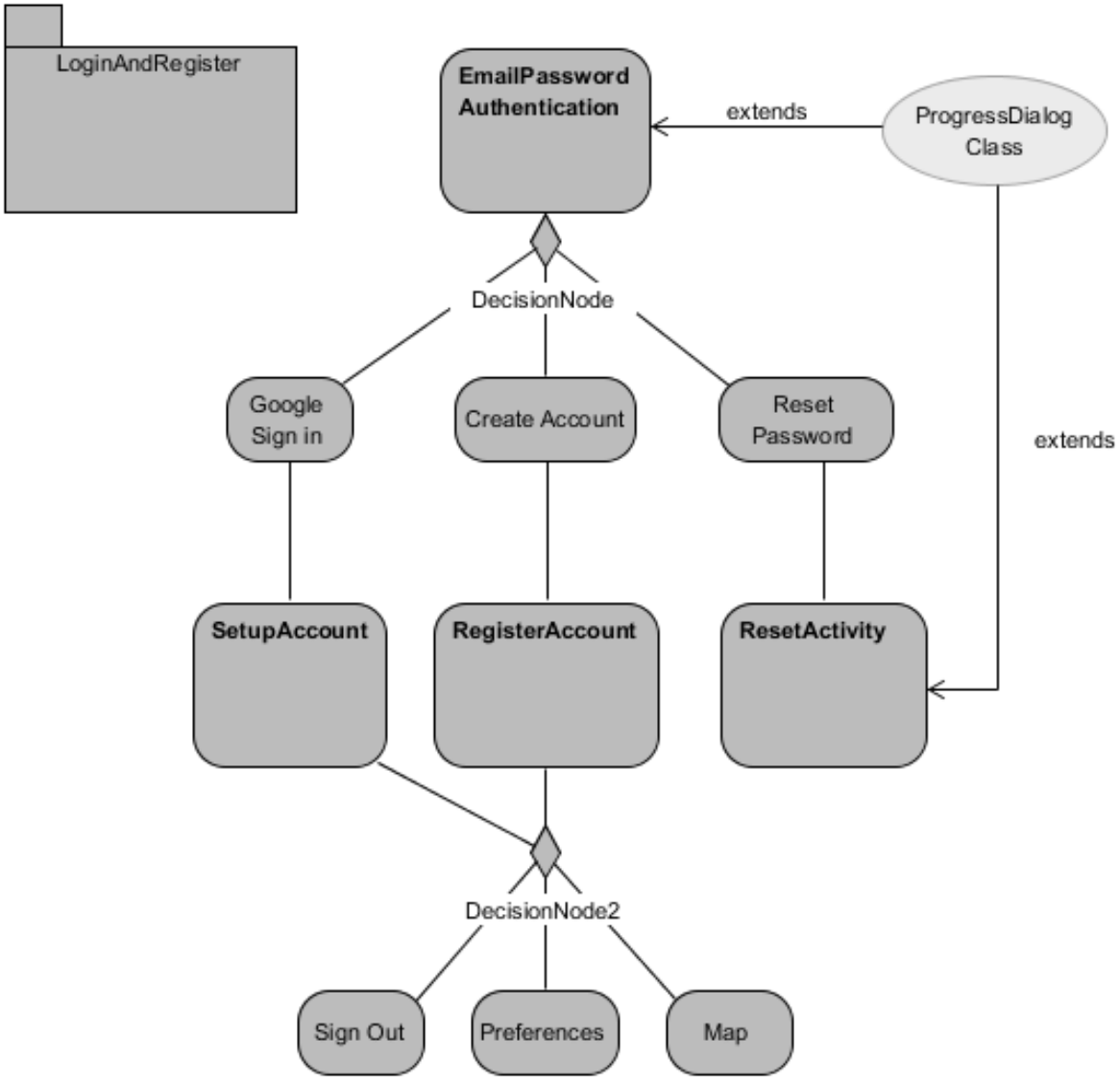

Figure 5.4: Login Activity Diagram

5.6.3 Login Sequence Diagram

Regarding the sequence diagram located in figure 5.5, The Reader can comprehend what tasks are being processed in the background by their actions on which they are pursuing. While starting the application and looking at the following diagram the reader can see the generic association between the actor and the EmailAndPasswordAuthentication activity which is created on starting the app. Which checks immediately if a user was already signed in using the mobile device and have not specifically signed out which allows the user to access a new set of activities which enable them to discover Dublin in a new light and to find hidden monuments that have stood the test of time.

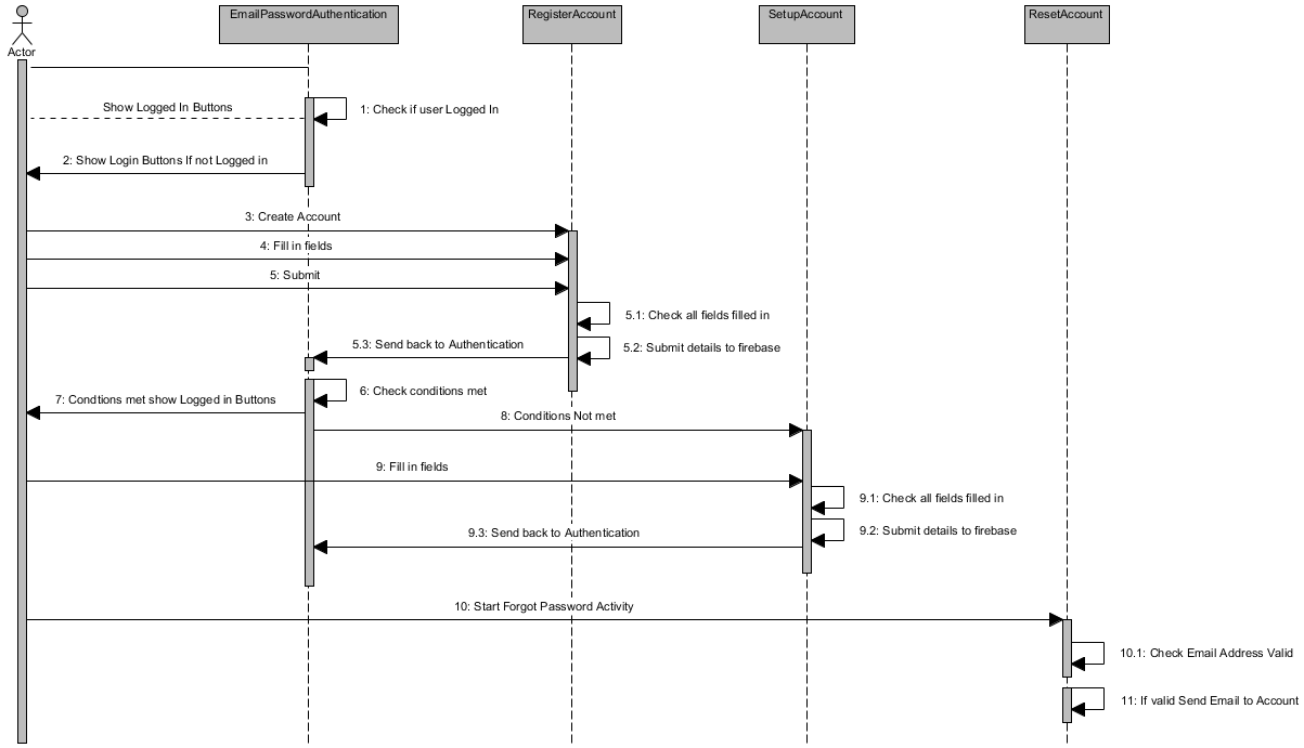

Figure 5.5: Login Sequence Diagram

While looking at the different activities on which the user may take there are also associated checks which need to be performed to proceed to the next area of the application. Even when a new user creates an account the following account needs to be verified with the Firebase database to check if such a user exists this is not just for authentication but to enable

users to post their accurate information with an allocated name, profile picture and unique ID which distinguish's them from other users. When a user has either created an account by registering with an indifferent email address or signing in with the aid of Google sign in, on the authorization of the account The user is brought back to the EmailAndPasswordAuthentication Activity. On which the user is validated, and new options are then shown to the user for them to proceed to the MapActivity or to change preferences as well as signing out of the application.

5.7 Maps and Geofencing Diagrams

The following section is to show using diagrams how the maps and geofencing activity works and its associated classes and activities that are combined to retrieve notifications on which a user has entered a specific geofence. With hence to its specific class, the maps activity must have an allocated map fragment which is defined on the designed layout that is accessed by a reference by the class to produce a suitable map. On gaining the Location of the device we first need to check the permissions of the application especially if the device is using Android Marshmallow or above. As there are new permission intents on which the user must explicitly accept to use specific components of such device. When these permissions have been accepted the application then has access to the GPS module located in the device which most mobiles have in this day and age. With these specific settings turned on and with the permissions set in hand, the app is then available to accept location updates which are set on that activities class and are updated by specific intervals which are set by the developer. These are just a few of the methods that the developer will run through on the next chapter but for now the diagrams of communication between the user and device.

5.7.1 Maps and Geofence Use Case Diagram

While viewing the following use case diagram located in figure 5.6, the reader can see that it is quite an easy diagram to comprehend as there are only three use cases that can be seen on which the user can use with specific button clicks. The main area of this map activity is to utilise the GPS locator of the device to receive updates on whether the user has entered a

geofence location which a notification and dialogue pop-up will appear to the user entering such an area.

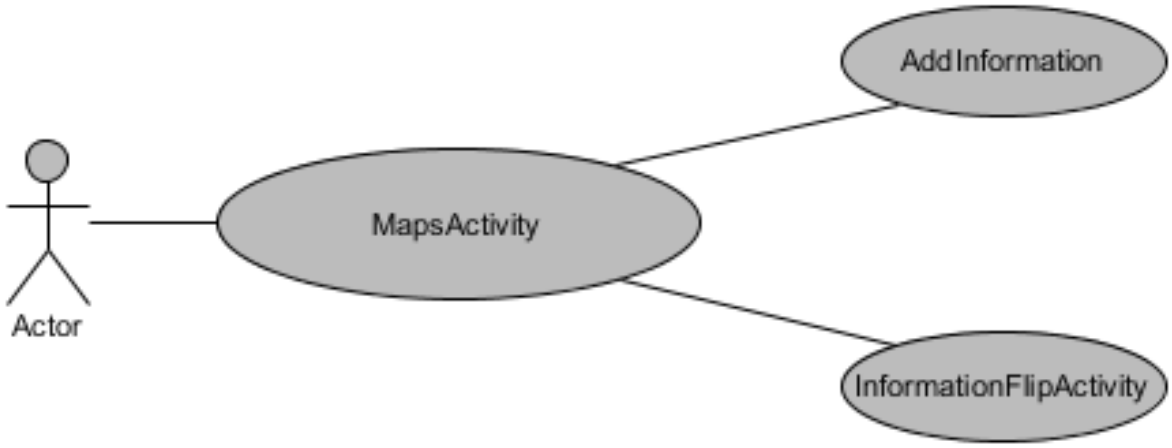

Figure 5.6: Maps & Geofence Use Case Diagram

5.7.2 Maps and Geofence Activity Diagram

We can see by looking at the following diagram figure 5.7, on which how each specific component is mapped to the main MapsActivity which does all of the computation with regards to retrieving a location from the device which allows for this app to be a location aware application. While looking at the two reference classes and distinguishing what if anything they do is simple. The Place class utilises the notification thrown when a user is in a geofence. The maps activity acts as a broadcast receiver to retrieve the information which a pending notification intent was thrown populates the Place methods with the name of the area which a notification was thrown and passed to the InformationFrontActivity when the users choose to enter the specific monuments activity.

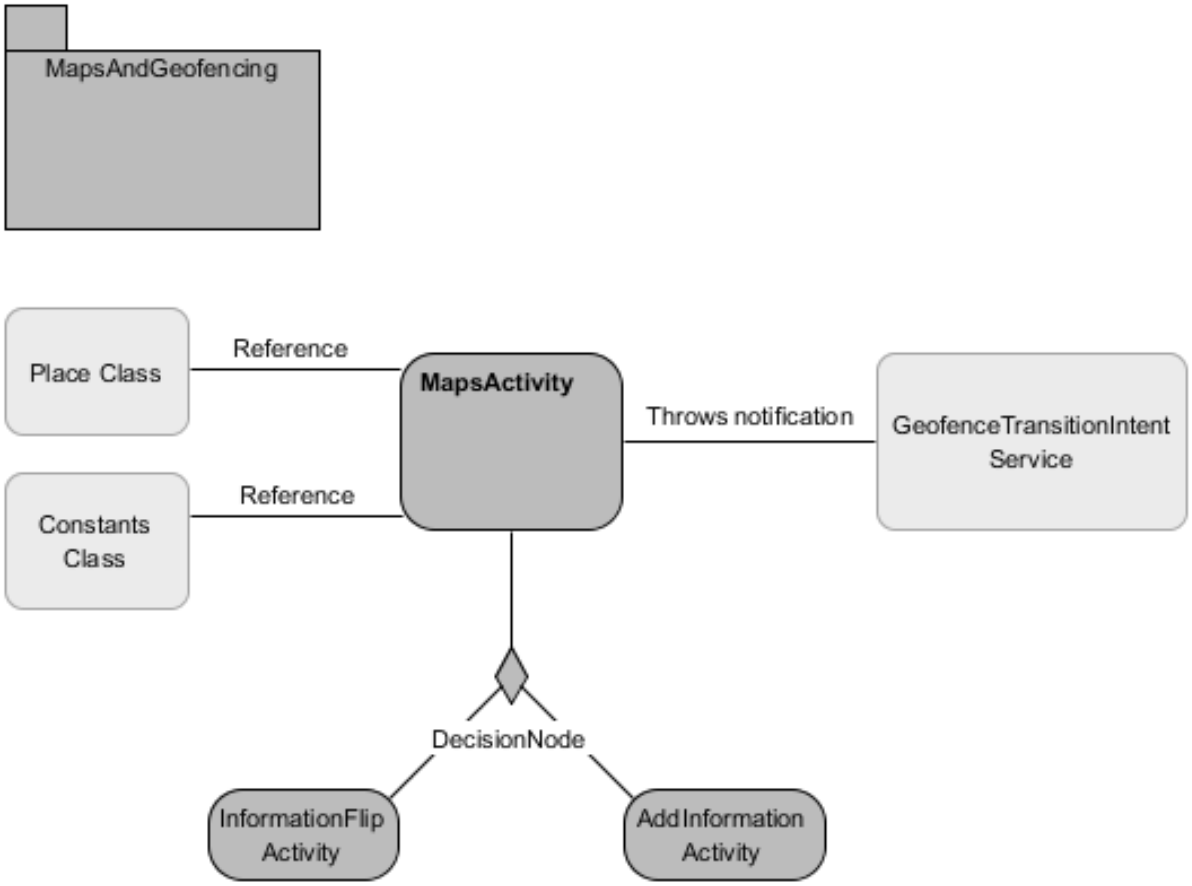

Figure 5.7: Maps & Geofence Activity Diagram

5.7.3 Maps and Geofence Sequence Diagram

The following sequence diagram located in figure 5.8 represents the states of which the user will be initiated through with hence to the background workings of the application. While viewing the diagram, the reader can see the initial connection between the user and MapsActivity. Which the onCreate of that specific activity triggers methods which set up the map fragment, adding geofences which are retrieved from firebase, GoogleApi which is needed to request the location from the user's devices as well as the connectors to the camera view to adjust the camera view of the map on location changes. With this in mind when the location changes the application renders whether the application has entered a geofence location. On going into a geofence, a pending intent will be initiated which will then trigger a notification which will display on the user's device. The user may want to view the information or not it is not compulsory to view the information on the application. Another opportunity on which a user might want to undertake is to send information to the InforMe@Dublin Gmail account to add more geofences with images and information to the InforMe@Dublin Android Application.

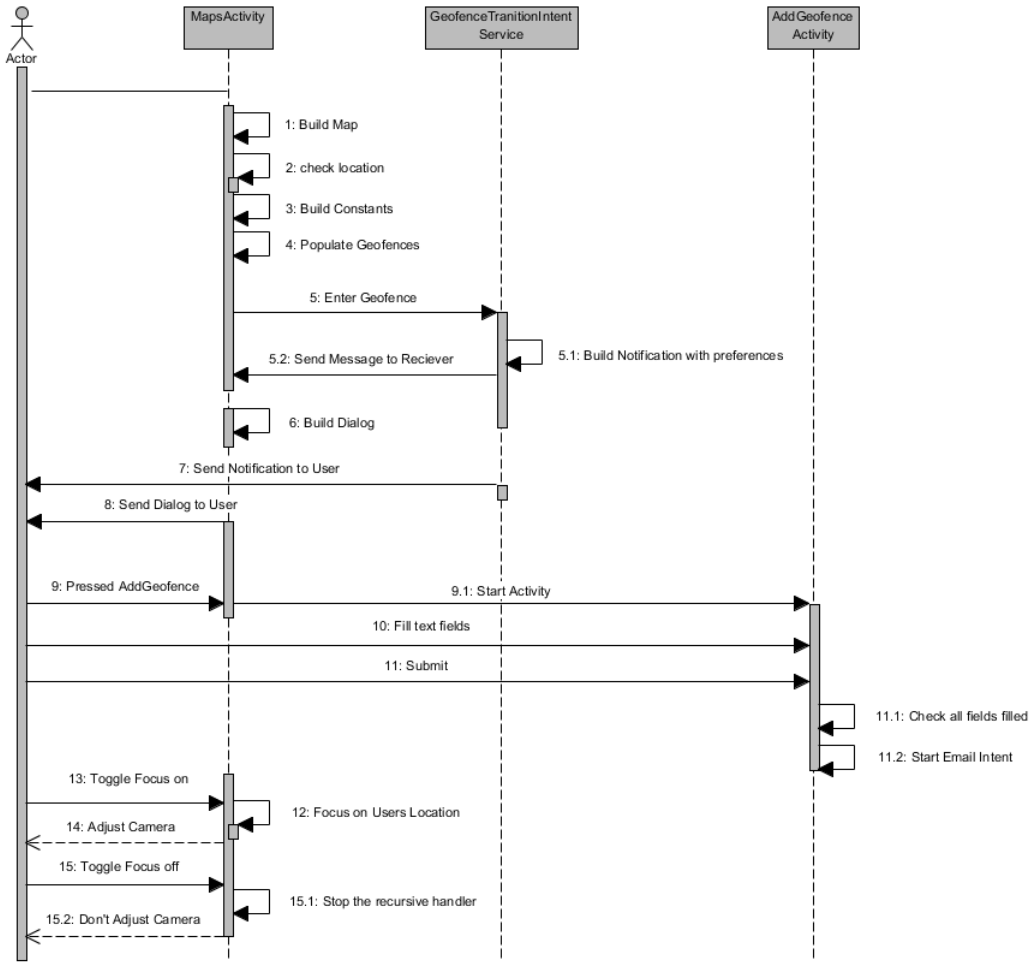

Figure 5.8: Maps & Geofence Sequence Diagram

5.8 Information and Posting Diagrams

During the following phase of the project is where the user can view the information of the monument on which they have entered and post some information about that specific area if they wish while they are in the same vicinity of the monument. These fragments which are created when the user selects to view the information on the specific area. On using the icons which change the view of each fragment activity when on the informationBackComments-Fragment is on the main view users can then choose to Post comments, update comments or just to view the post as a whole without other posts in the way. The following Diagrams

will display how each component works together to retrieve information for where exactly the user is located.

5.8.1 Information and Posting Use Case Diagram

Regarding the following diagram figure 5.9 showing the use case diagram as shown distinguishes how each of the user's actions is mapped on clicking certain buttons which were applied to the layout file and linked to the activity. On entering the InformationFlipActivity, the user is sent to the InformationFrontFragment which displays the information of the monument of which the user has entered the specific geofence. The user can only access these specific activities on entering from an allocated geofence.

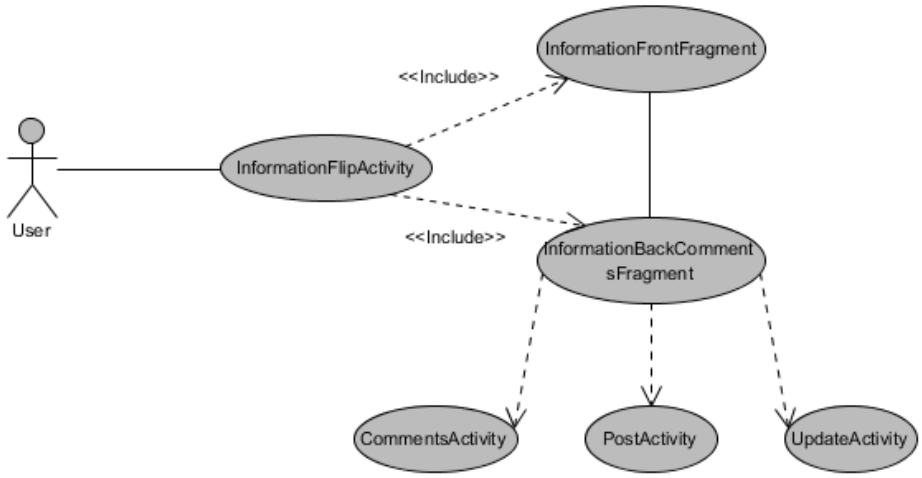

Figure 5.9: Information & Posting Use Case Diagram

5.8.2 Information and Posting Activity Diagram

Regarding the Activity diagram in figure 5.10, the reader can see how each part of the package PostingInformationAndComments the classes as follows in the diagram. The InformationFlipActivity controls both InformationFrontFragment and InformationBackCommentsFragment both of these fragments are controlled using the menu bar of these fragments the InformationFlipActivity uses a stack to flip each of the fragments back to front. Only when the InformationBackCommentsFragment is displayed may the users create and view posts made by other users and also themselves When viewing the posts, the user can then like specific posts that have been created by the user or other users.

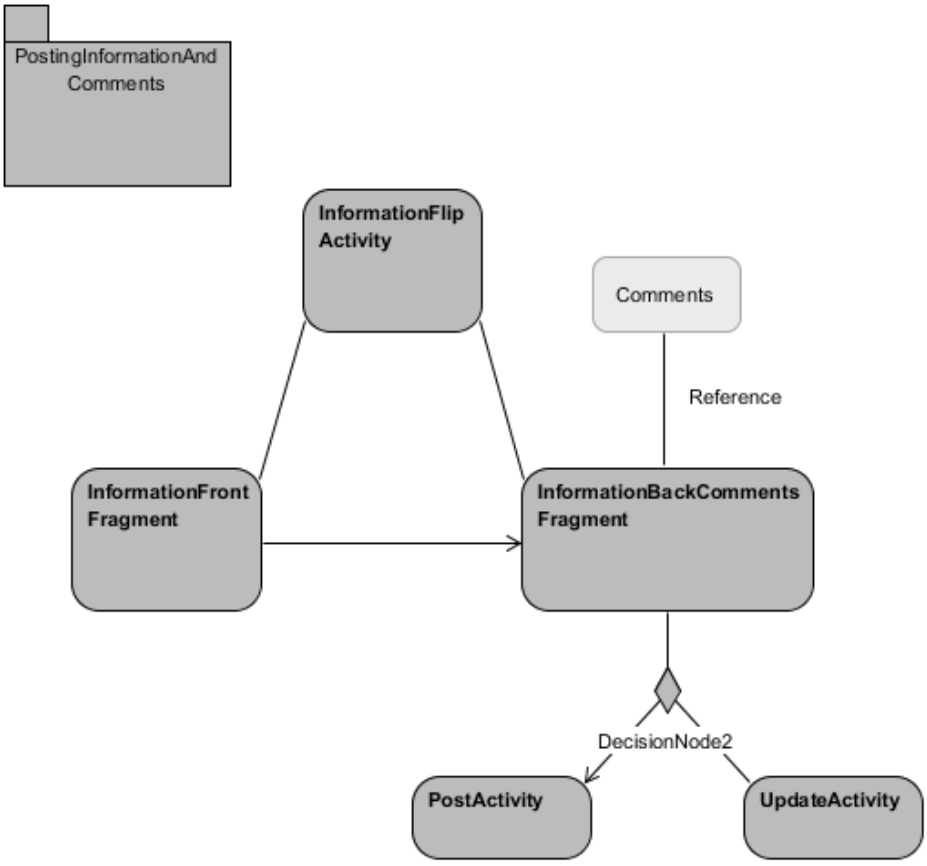

Figure 5.10: Information & Posting Activity Diagram

5.8.3 Information and Posting Sequence Diagram

Looking at the diagram in figure 5.11 the reader will grasp a better understanding of the following Activities on viewing the information of the specific monument as well as viewing the posts which are retrieved using firebase methods on which the developer will discuss in the following chapters. On entering the front fragment, the following methods are then initiated on starting the activity such as retrieving the information from the Firebase database which has been referenced to the specific JSON file. The fields of the activity will then be populated with the information which has been retrieved as well as images being loaded with specific URL's which have been saved in a hashmap and loaded within a page scroller. On viewing the posts from the user, these are populated by using the InformationBackCommentsFragment as well as the Comments class to populate the data and to display in a page view of specific user's posts which were posted on that monuments page.

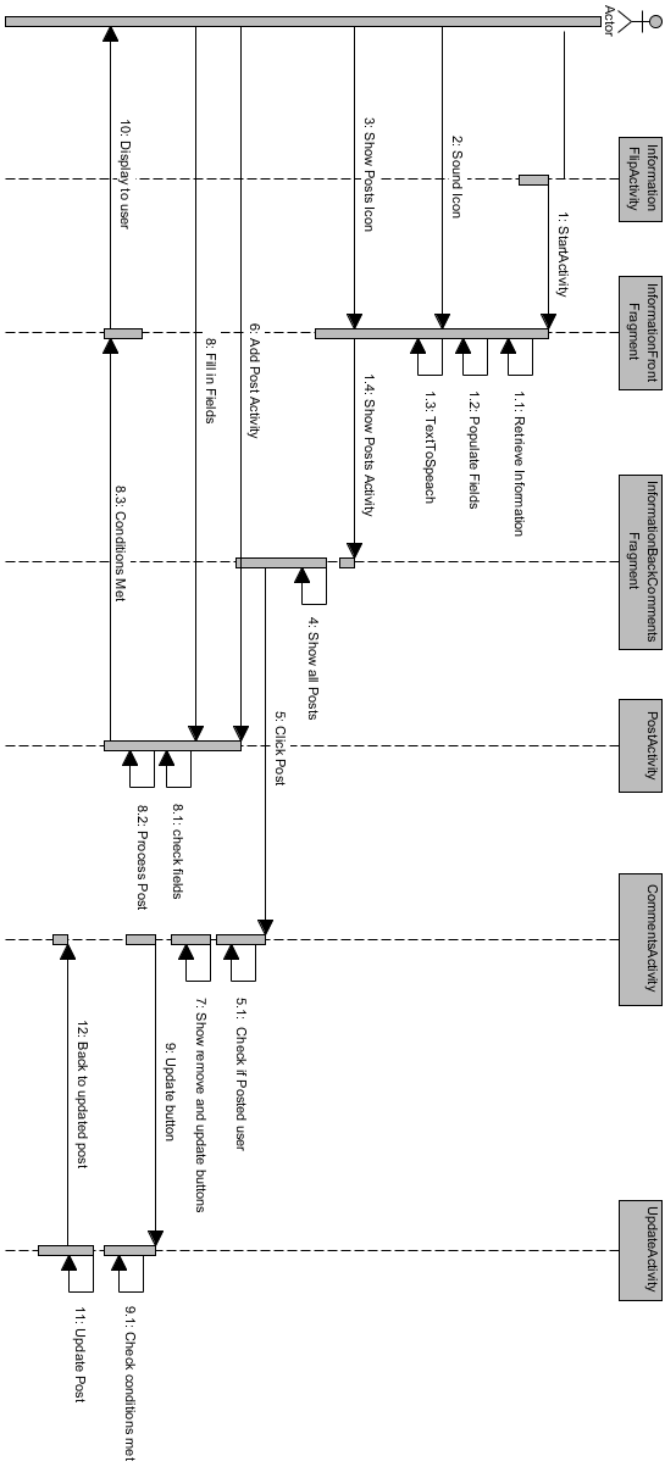

Figure 5.11: Information & Posting Sequence Diagram

5.9 Settings Diagrams

The following diagrams regarding the settings preferences are discussed throughout this section of the chapter. With hence to the settings activity, the user can change the application's preference with regards to setting the vibration of the notification which is sent on entering the geofence. The user can also change the notification sound as well as a battery save mode which utilises the proximity sensor to sense if the device's screen is on and if the sensor of the device is covered.

5.9.1 Settings Use Case Diagram

The following diagram shows the reader how the Settings and activities that are combined within the preference activities which allow the user to change specific settings for that application the user can change the settings as follows with the following activities as shown in the figure 5.12.

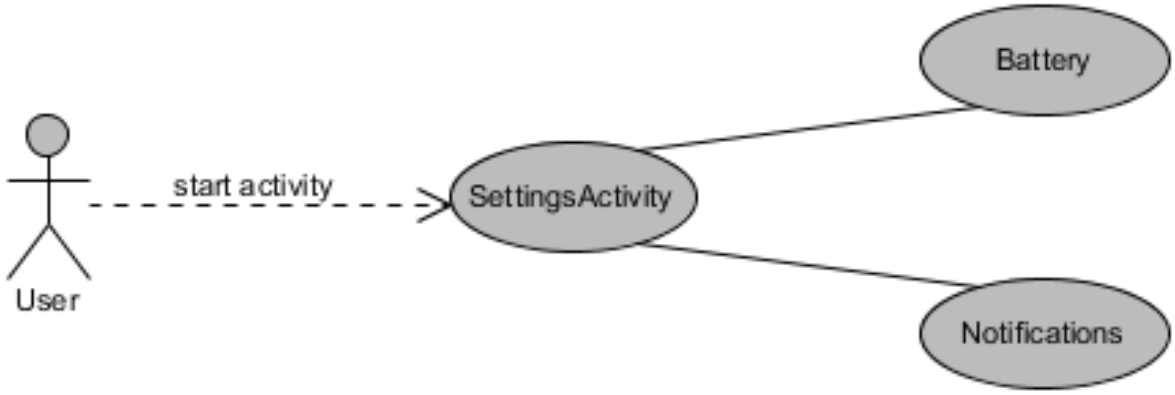

Figure 5.12: Settings Use Case Diagram

5.9.2 Settings Activity Diagram

The following diagram located in figure 5.13 shows the reader how the Settings and AppCompatActivity reference to each other when changing the preferences of the user. The CheckConnectivity class represents the application to see if the device is connected to the internet and if the Gps module is activated.

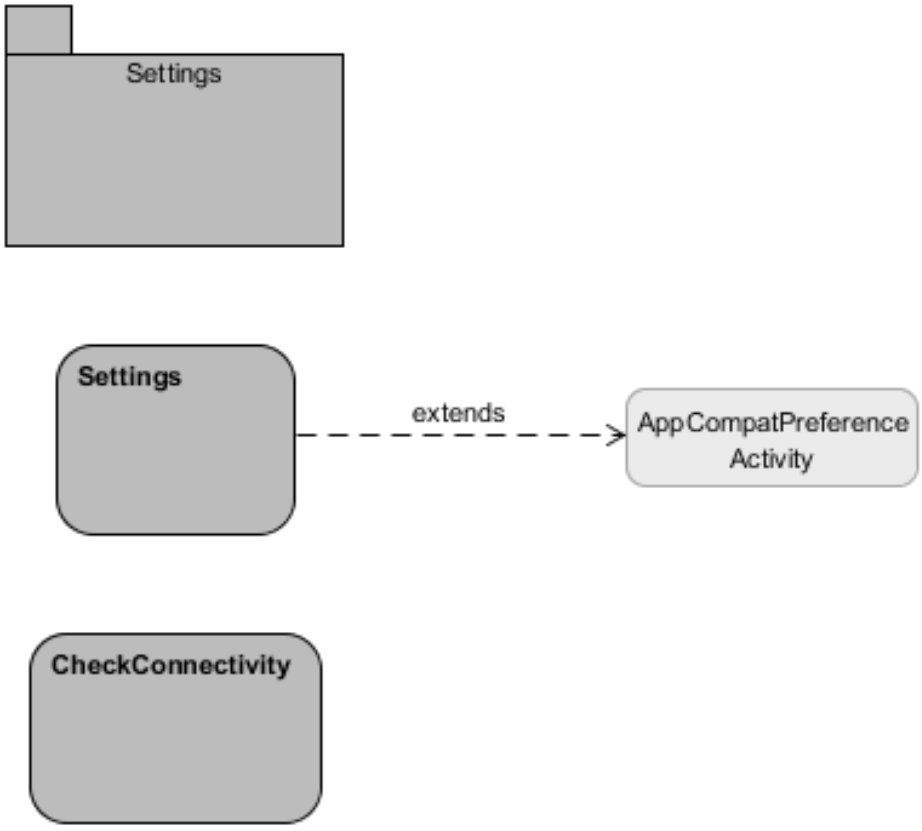

Figure 5.13: Settings Activity Diagram

5.9.3 Settings Sequence Diagram

While viewing the following diagram located in figure 5.14 shows how the user associates with the settings Activity while utilising the preference layouts as well as the AppCompat-Activity this then shows the preference which will be discussed in the next chapter of this thesis.

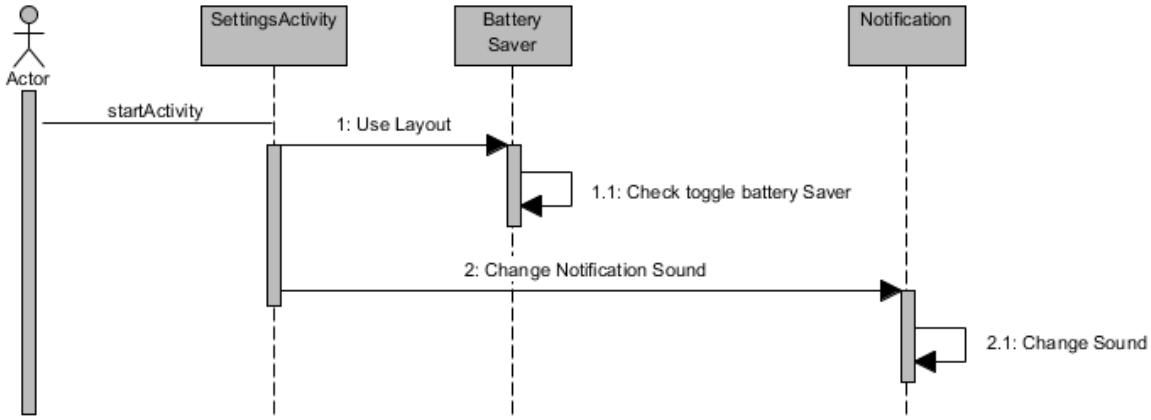

Figure 5.14: Settings Sequence Diagram

5.10 Overall System Use Case Diagram

The following diagram represents the InforMe@Dublin application as a whole. Concerning the following use case diagram located in figure 5.15, the reader can then see all of the different components on which a user can enter within the constraints of being signed in and with the submission of posts by the specific user being able to remove and update their specific posts. The reader can see how each component is connected to each other and on where the user can gain access to certain activities throughout the following application.

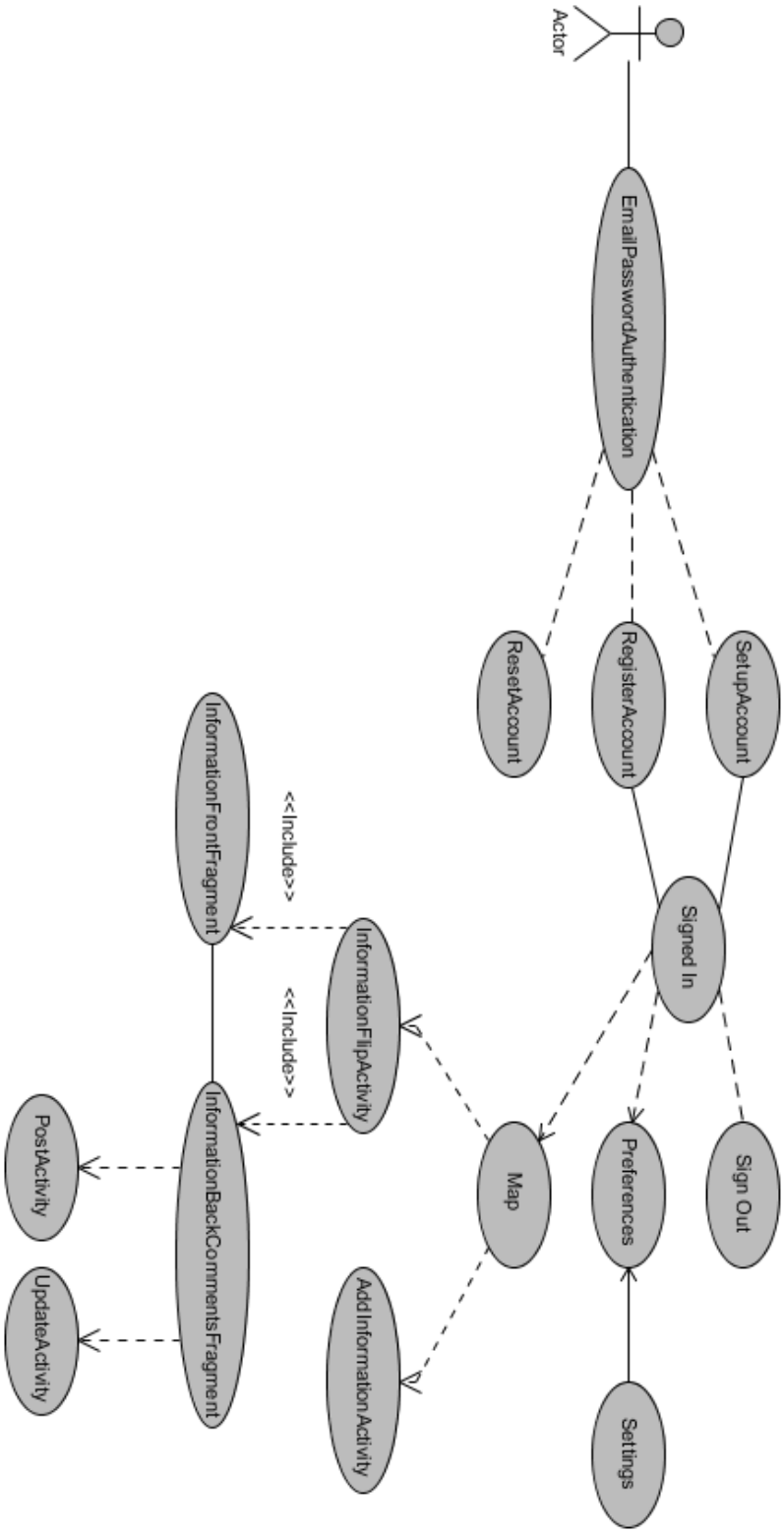

Figure 5.15: Overall Use Case Diagram

5.11 Activity Diagram with Overall System

Regarding the diagram figure 5.16, The activity diagram as a whole is to show the reader how each component is used throughout the system to grasp a better understanding of which Activities and classes are working and what exactly is each of these specific items doing throughout the Android application. When brought together as a whole with descriptions of each package and the associated classes and activities the reader may gain a better understanding of the following project. The InforMe@Dublin Android application is not meant to be complex in nature but descriptive enough to show the findings the developer designed within the application.

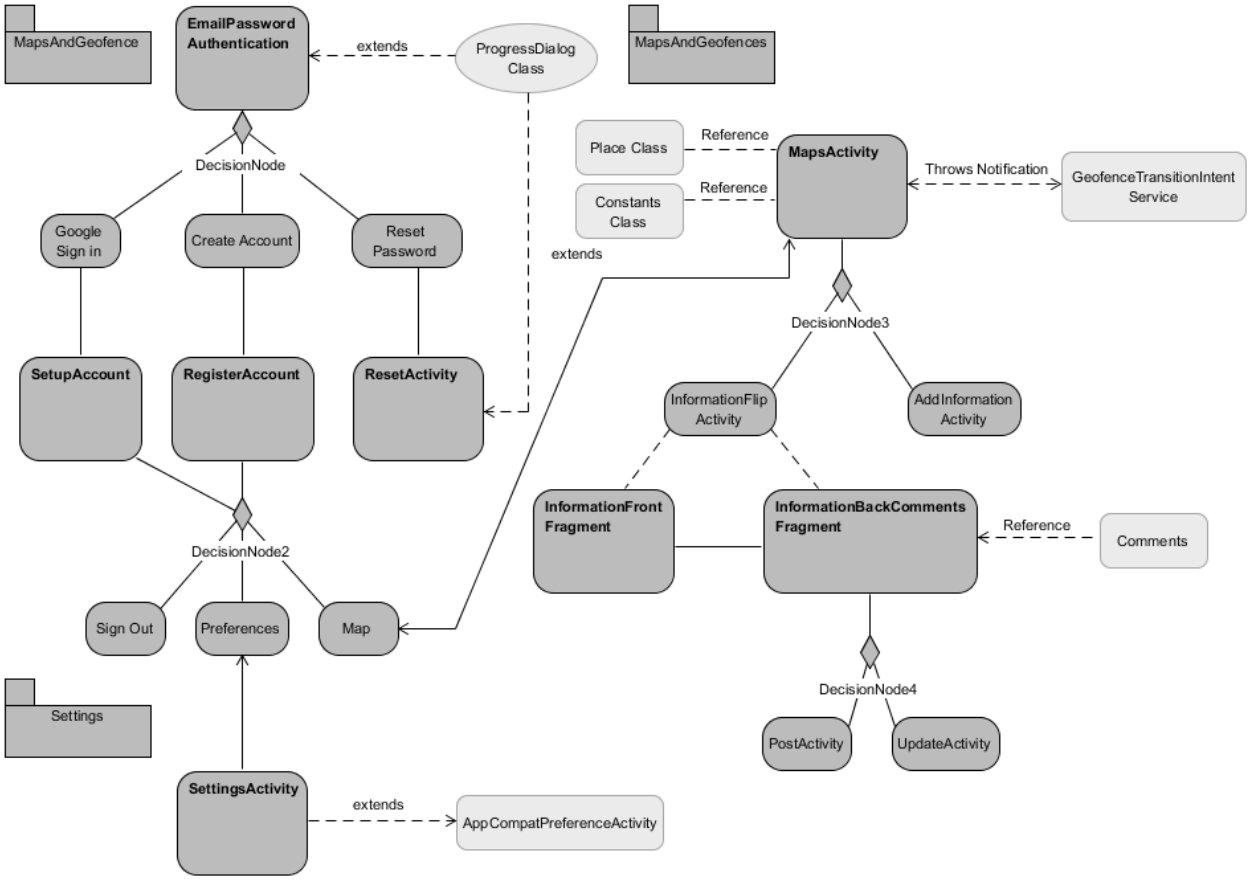

Figure 5.16: Overall Use Case Diagram

5.12 Summary of the Chapter

Throughout the following chapter, the developer has brought us through the different steps of how the user can interact with the application as a whole, and of which components of the app are connected to one another. Another aspect that the developer has to show us is the communication between the application within the packages of which are of a modular concern to the specific components of the app, concerning the MapsAndGeofence package holds all the classes and activities which are only used once a patron of the application is in that specific activity. With that in mind, the developer hopes that the reader understands how the application is mapped out and thoroughly understands of which the developer is trying to achieve. The next chapter of the following thesis is to show the wireframing and design aspects of the InforMe@Dublin Android application with hence to colour schemes and layouts of each specific activity.

Chapter 6

Application Design

6.1 Description of the Chapter

The following chapter discusses the design of the application, with regards to layout and colours used throughout each page. Within the chapter, the developer will provide wire-frames with regards to the thought process of the layout resource files. The Logo of the application will also be discussed with the reason behind the naming of the application provided below.

6.2 Creation of InforMe@Dublin's Logo

The inspiration for the InforMe@Dublin came from the using Google maps location markers of which is an iconic symbol to use of which ties into the following application quite well as it utilises maps and a mobile devices real-time location. The naming of the application is as the name state's to inform someone about something which in this case is the county of Dublin. The colours of which are used in the logo were just clean and crisp looking of which complimented the other parts of the logo. Other colours that were tried just didn't tie in with the look and feel of the application.

Figure 6.1: InforMe@Dublin Logo

6.3 Start Page

While designing the startup page of the application, certain permissions are requested in order to retrieve the device's location for the application being able to use the GPS module and to access the storage of the Android phone. The following wireframe represents the application after the installation has been completed and opened the following screen will be shown to the user. While looking at the following wireframe located in figure 6.2, The logo is the first aspect of the application on which the user should see which represents what the app is actually about the layout is also smooth and well round

Looking at the wireframe which was designed by the developer the reader can see where the user may log in with their credentials as well as using OAuth sign in by linking their personal Gmail account which continues to the setup of an individual account on the InforMe@Dublin app. The button is in the lower centre of the page with the button Google sign in. With regards to the login activity, the layout is a relatively appealing design.

Figure 6.2: Login Wireframe

6.4 Colour Scheme

After looking at the first screen login wireframe. The developer would like to display the colours of which will be used throughout the app the colours are as follows. The developer chooses these specific colours as it complements the project as a whole. Are used throughout the multiple pages with are incorporated throughout the app. The following colours make for a sleek look while users are utilising the app.

1. Theme of Application - R 43 / G 152 / B 212
2. Buttons of Application - R 249 / G 159 / B 27
3. Logo of Application - Blue: = R 0 / G 167 / B 157 Green: = 56 / G 180 / B 73

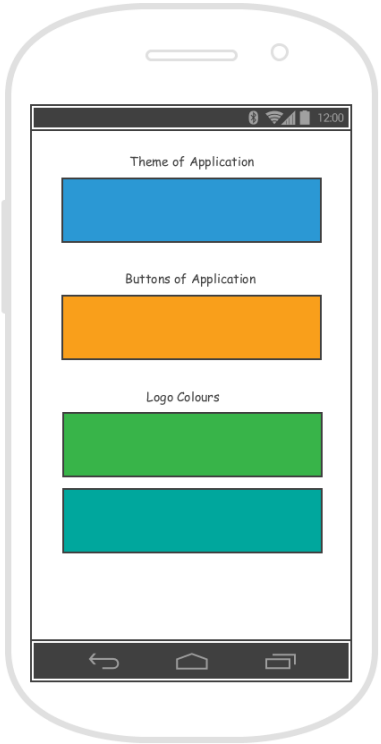

Figure 6.3: Colours Wireframe

6.5 Wire-frames

The wireframes in this section of the chapter are regarding the wireframes on which were created by the developer. These wireframes were meant as a guide on how the layouts of the different activities would look like on completion of the application in question. Each styling aspect of the application was followed throughout the development. All colours used were disclosed in the above section.

6.5.1 Login

The following logged in wireframe is to be used in conjunction with the login wireframe which will authenticate the user as well as sign in the user of the application. Different display buttons will be shown to the user on authentication with both firebases authentication and the database of the Firebase API.

Figure 6.4: Logged in user Wireframe

6.5.2 Map

With regards to the Maps Activity wireframe on which was designed by the developer, shows were the specific buttons are for the user throughout the using of the application. While viewing the wireframe, the reader can then distinguish how the real activity will be laid out. On the left-hand corner of the activity shows the compass area on which the user can choose to use. On the right-hand corner is used to toggle the focus of the user's location on which changes the camera animation to the devices current location utilising latitude and longitude. With regards to the bottom half of the screen on the bottom left-hand side of the display is where users can choose to send information to the InforMe@Dublin developer to update the Firebase database to add more historic locations for other users to enjoy.

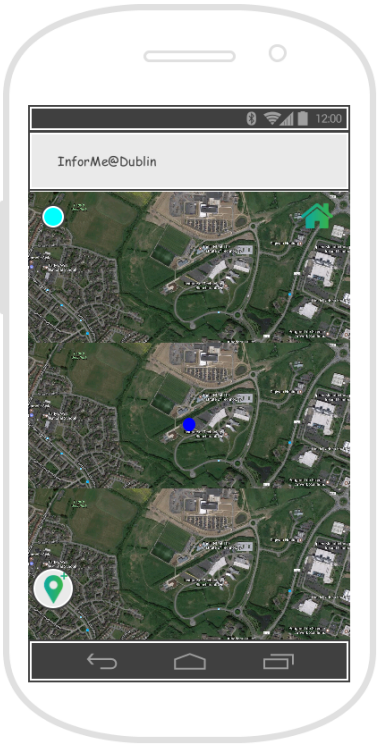

Figure 6.5: Map Activity Wireframe

The next wireframe in this section shows what happens when the user has entered a particular geofence a dialogue is displayed with the monuments name on which the user of the application can then continue to view the information on the specific location on which

the dialogue appeared for. A notification will also be sent on entering the location which is a general notification of the application notifying the user of a geofence on which they have entered.

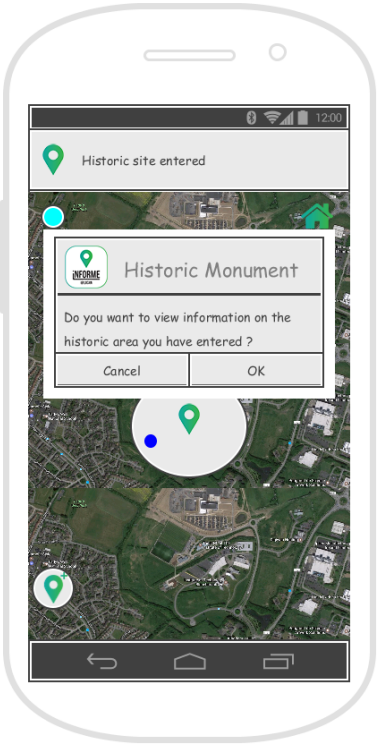

Figure 6.6: Map Activity Geofence Wireframe

6.5.3 Settings

The following section of this chapter represents the settings page of the application with a more particular reference to the preferences of the application. Within the following wire-frame, the following preferences can be changed in the activity that is related to the application.

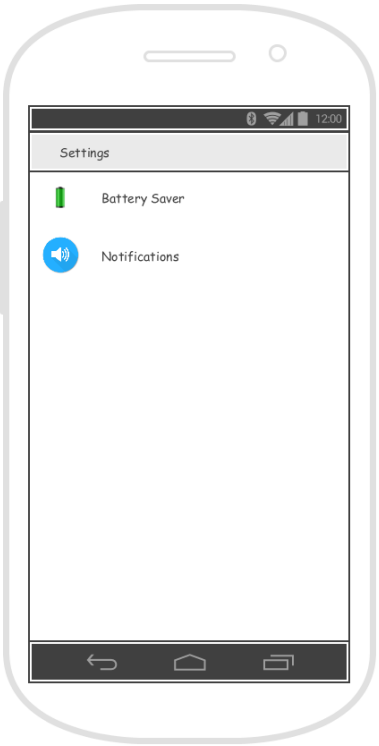

Figure 6.7: Settings Preferences Wireframe

6.5.4 Add Information

Concerning the following figure 6.8 shows the design on which will be implemented on the InforMe@Dublin app. with the wireframe in mind the user will be able to fill in the fields that are necessary for the app to create an email and send to the developer of the application. By pressing the select images button, the users will be brought to their personal gallery on which they can choose images to post to the developer as well as any information they might

have at hand such as the location of such monuments.

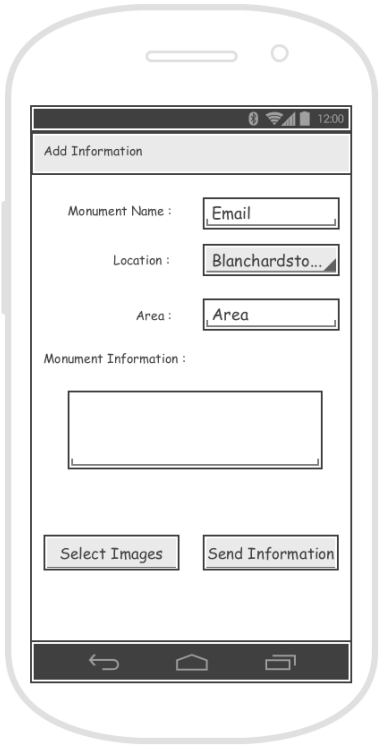

Figure 6.8: Add Information Wireframe

6.5.5 Show Information

With relation to the following section located in the system design. Two wireframes were developed in which the developer of the application could get a better understanding of what he wanted to achieve. The following wireframes were turned into resource layout files to complete the look and feel, and the developer thought that these following layout files would be the most important design in such an application. Information is the most important factor in the project which needs to be portrayed to the user in a specific manner. Creating a crisp clean and non-cluttered layout was the key ingredient which why the wireframes layout below was chosen.

While looking at both figures 6.9 and 6.10, the reader can see the distinctive feel of what the developer is trying to achieve by using both of the following activity wireframes as

Figure 6.9: Display Information

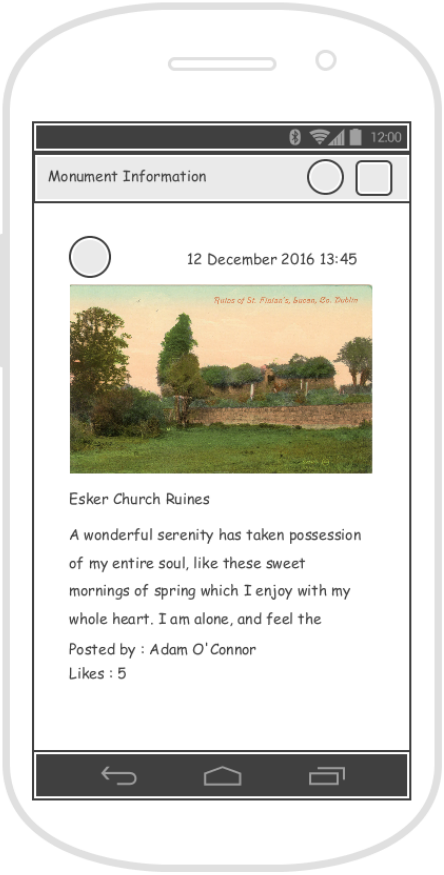

Figure 6.10: View Posts

fragments the developer can create a smooth turn around feel of the information on which the users want to see. While looking at figure 6.9, users can get to grasp with the images that have been used from multiple sources. The information that the reader can see is just filler text that was generated from a website. Were the filler text is in the wireframe is compiled of the information from that monument which was sourced from multiple sources on the internet. A button which is shown as a loudspeaker icon will then perform a text-to-speech on the information which was retrieved from the Firebase database.

6.6 Summary of the Chapter

To conclude the chapter as a whole, the reader can see that the developer has discussed the design of the InforMe@Dublin application. Each page of the app has been mocked up which allowed the developer to visualise how it should look at the end of development. While providing these wireframes in this current chapter, the reader can grasp a much better understanding of how the application prototype will evolve throughout the time of development and also with regards to the time scale allowed on which the final build and release of the project will be of the date 27th of April. The next chapter in this thesis discusses the coding of the application on the loading of geofences, providing permissions to the user on Android Marshmallow OS and above and many more feature on which were incorporated into the app. Each separate part of the next chapter will be discussed with regards to the different methods of which are associated with its specific method.

Chapter 7

Implementation of System

7.1 Initialization of the System

The installation of the application is available on the google play store, the name of the application is InforMe@Dublin and is created under the company Smooth Solutions which is the name of which was created by the developer Adam O'Connor.

7.2 Background to Android Devices Activity's Life-cycle

Regarding the activities on which each of these specific sections in the following chapter is based on a single activity page where all these methods are then utilised. The Android Activity Lifecycle is controlled by the use of seven individual methods of which are related to the android.app.Activity class which is a subclass of the ContextThemeWrapper class.

The activities in an Android app is just plainly a single screen in Android, or in another example just a window on a computer or a frame in creating Java applications. With the help of these activities, developers can then place all the UI components or widgets on this screen.

Below is a list of how each of the specific methods of an android application activity is used in figure - 7.1 as well as what each of the methods of the activity does which is located in the table - 7.1.

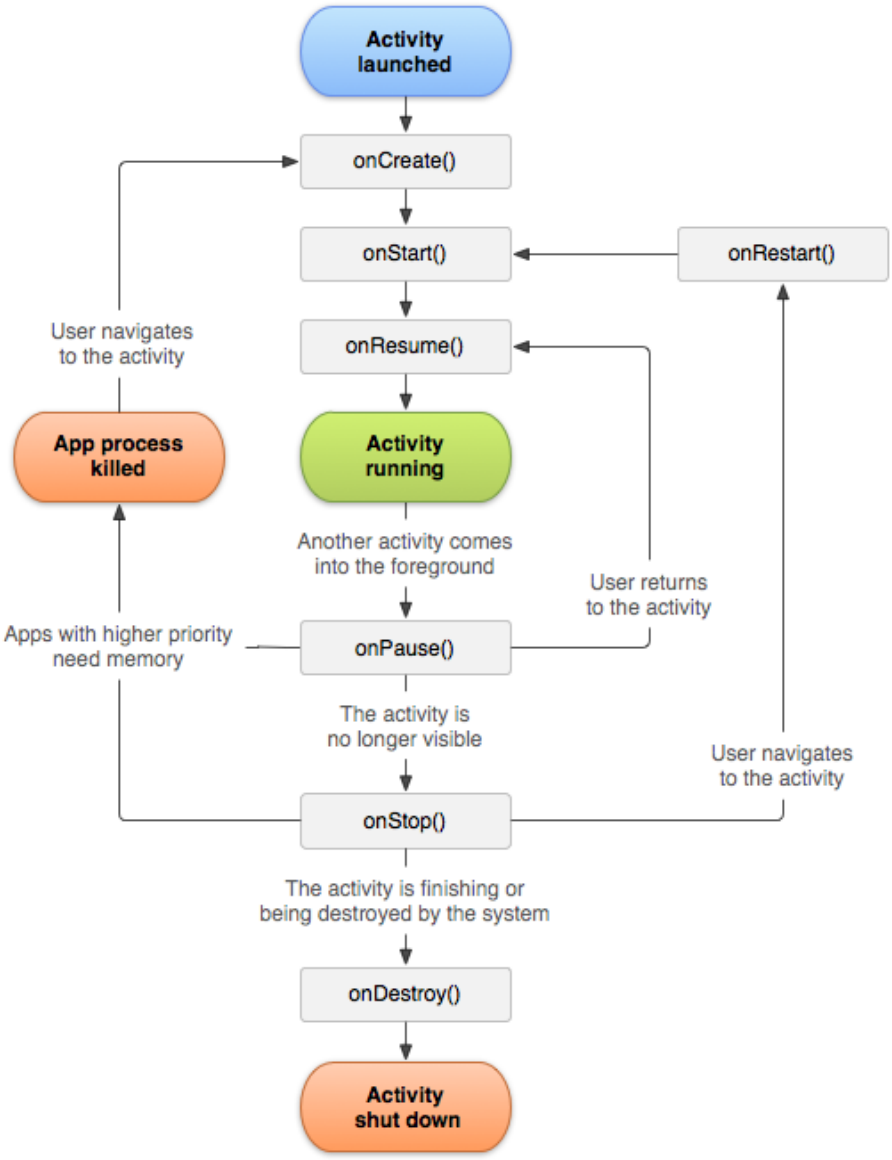

Figure 7.1: Activity Life-cycle JavatPoint.com (2013)

Table 7.1: Activity Lifecycle

Method	*Description*
onCreate	called when the activity is first created
onStart	called when the activity is becoming visible to the user.
onResume	called when the activity will start interacting with the user.
onPause	called when the activity is not visible to the user.
onStop	called when the activity is no longer visible to the user.
onRestart	called after the activity is stopped, prior to start.
onDestroy	called when the activity is destroyed.

7.3 Putting InforMe@Dublin Application on Google Play

The informe app is signed with the author's credentials so that no intellectual information can be stolen or the application cannot be deconstructed by an unauthorised person to get the property of the developer. By uploading the application for reproduction on the Google play store the developer of informe had to gain the necessary SHA-1 encryption keys which were produced by the NSA used as a cryptographic hash function. The related encryption key creates a 160 bit or 20-byte hash value or also known as a message digest.

The SHA-1 keys are needed to provide the related API keys on which are necessary for the usage of Google Maps as well as the OAuth used for the signing in of users authenticating with their Gmail account.
"C:\Program Files (x86)\Java\jdk1.8.0_65 \bin> keytool -list -v -keystore "C:\Users\Adam O'Connor\keyforInforMe\InforMe@Dublin.jks" -alias alias_name -storepass password -keypass password"

To obtain the SHA-1 key used in the generated signed APK the above console command is needed in order to retrieve the key which the developer needs to add to the Firebase database and OAuth 2 in both the Firebase console and Google API console. The Generated key store used to sign the application is needed to retrieve the SHA key.

Figure 7.2: Crypt Key Screenshot

7.4 A Way of allocating Permissions

With regards to the new permissions that are now needed by applications to run on Android applications, a code listing was provided by the developer below. The new permissions were introduced by the Android development team with the release of Android marshmallow instead of users accepting terms of the downloaded application from the Google play store, and users would need to accept these permissions when needed throughout the use of such an application with are requested one by one. The developer has to call each request manually which is below.

```
// check permission request of 99
public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;

/**
* used to check the location permission
* @return
* dialog if user has not set permissions
*/
public boolean checkLocationPermission() {
    if (ContextCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION)
    != PackageManager.PERMISSION_GRANTED) {
```

```
            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale(
                this,
            Manifest.permission.ACCESS_FINE_LOCATION)) {

                // Show an explanation to the user *asynchronously* --
                    don't block
                // this thread waiting for the user's response! After
                    the user
                // sees the explanation, try again to request the
                    permission.

                //Prompt the user once explanation has been shown
                //(just doing it here for now, note that with this
                    code, no explanation is shown)
                ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION
                    },
                MY_PERMISSIONS_REQUEST_LOCATION);


            } else {
                // No explanation needed, we can request the
                    permission.
                ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION
                    },
                MY_PERMISSIONS_REQUEST_LOCATION);
            }
                return false;
        } else {
            return true;
        }
    }
```

Listing 7.1: Permissions Example

The above code is used in conjunction with the requesting of the specific location permission of the device on which the app has been downloaded and opened. The location dialogue is the first permission on which would be shown to the user if the permission has not been set then the user cannot explicitly continue using the application. The dialogue module would simply keep showing until the user agrees to accept the location permissions. In conjunction

with the checking location permissions method, an OnRequestPermissionsResult method is also used for the retrieval of such a permission request. The following method checks if the specific user has chosen to allow the app to receive the location of that app.

```
/**
*
* @param requestCode
* start to see if user connected
* @param permissions
* permissions which are needed for the application
* @param grantResults
* if the user has granted access to sensors.
*/
@Override
public void onRequestPermissionsResult(int requestCode,
String permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_PERMISSIONS_REQUEST_LOCATION: {
            // If request is cancelled, the result arrays are
               empty.
            if (grantResults.length > 0
            && grantResults[0] == PackageManager.
               PERMISSION_GRANTED) {

                // permission was granted, yay! Do the
                // location-related task you need to do.
                if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED) {

                }

            } else {

                // permission denied, boo! Disable the
                // functionality that depends on this permission.
                checkLocationPermission();
            }
            return;
        }

        // other 'case' lines to check for other
        // permissions this app might request
```

```
        }
    }
```

Listing 7.2: Permissions Request Example

7.5 Utilising Firebase's Auth method

Regarding the Firebase Auth method on which includes a user management system. With the following in mind, the developer can compare some basic data against the Firebase Auth users. Utilising the following method allows the developer to offer multiple login methods such as email/password, Google, Facebook. The following lets users link their accounts into a single Firebase account. Using the Auth method provides pre-existing auth system so that the application can take advantage of Firebase's security rules which have been set by the developer.

7.6 Logging into Cloud services

The cloud service on which this application connects to is Firebase. Firebase controls the authentication between the application itself and allows the signed in person to retrieve information which is located in the database. The following code was used in coordinance with the authentication with the following application.

Email/password authentication has functions to register new users as well as signing them back in once registered. Firebase auth also provides a method of signing a user out of the application all functions return promises on which the new user registration as well as creating a profile on the database automatically signs the user in. The login information of the app is saved and holds the signing in information of the user if they have not signed out.

```
/**
* used for the signing in of a user which has logged in
* with their email address and password.
* @param email
* email address of the user which created an account.
* @param password
* password on which user wants to use.
```

```
 */
private void signIn(String email, String password) {
    Log.d(TAG, "signIn:" + email);
    // validate the textfields user has filled in.
    if (!validateForm()) {
        return;
    }

    showProgressDialog();

    // start the authentication of the email address and password
        of the user.
    final Task<AuthResult> authResultTask = mAuth.
        signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult
        >() {
         @Override
         public void onComplete(@NonNull Task<AuthResult> task) {
             Log.d(TAG, "signInWithEmail:onComplete:" + task.
                 isSuccessful());

             if(task.isSuccessful()) {
                 checkUserExistsDatabaseOnReEnter();
             }

             // If the sign in fails displays a message to the user
                 . If sign in succeeds
             // the auth state listener will be notified and logic
                 to handle the
             // signed in user can be handled in the listener.
             if (!task.isSuccessful()) {
                 Log.w(TAG, "signInWithEmail:failed", task.
                     getException());
                 Toast.makeText(EmailPasswordAuthentication.this, R
                     .string.auth_failed,
                 Toast.LENGTH_SHORT).show();
                 mStatusTextView.setText(R.string.auth_failed);
             }

            hideProgressDialog();
         }
    });
}
```

Listing 7.3: Authentication Example

The above method concludes the sign in of the application using the user's email and password to gain access to the following application. While choosing the sign in button after the user has entered their credentials the email and password will be used together with the use of Firebase Auth. The information is then sent to the Firebase project which is linked to the project utilising a google-services.json file of which is compromised of the client information of the app as well as the OAuth information which relates to the application. The google-services.json file will be added into the appendix in this chapter.

With the following in mind, the Google authentication of the users is used in the same way with some other methods used to map the OAuth 2 authentication together with the Google services API on which is automatically created in the Google console. OAuth with web authentication is needed as it is a requirement of Android if the app is released on the Google play store as some instances are authenticated with the web authentication.

```
// Configure Google Sign In
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(
    GoogleSignInOptions.DEFAULT_SIGN_IN)
     .requestIdToken(getString(R.string.default_web_client_id))//
        requesting the specific token the Google sign.
     .requestEmail() // requesting the email address of the user to
         sign in.
     .build();

// creation of a Google API client for the connection of a Google
    token of the OAuth 2 API in the Google Console.
mGoogleApiClient = new GoogleApiClient.Builder(this)
     .enableAutoManage(this, new GoogleApiClient.
        OnConnectionFailedListener() {
         @Override
         public void onConnectionFailed(@NonNull ConnectionResult
            connectionResult) {

         }
     })
     .addApi(Auth.GOOGLE_SIGN_IN_API, gso)// addtion of the Google
        sign in API and Google sign in options.
     .build();
```

Listing 7.4: Google Authentication Example

7.7 Creation of User Preferences

About the following section which utilises users preferences throughout the use of the InforMe@Dublin application. The developer will display some of the code of which is related to using user's preferences within an Android application environment the developer used preferences in informe by way of reusing the same activity but by producing different layouts to the user. Here is a following code listing of the preference method.

```
/**
* This fragment shows notification preferences only. It is used
   when the
* activity is showing a two-pane settings UI.
*/
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public static class NotificationPreferenceFragment extends
    PreferenceFragment {
     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         addPreferencesFromResource(R.xml.pref_notification);
         setHasOptionsMenu(true);

         // Bind the summaries of EditText/List/Dialog/Ringtone
            preferences
         // to their values. When their values change, their
            summaries are
         // updated to reflect the new value, per the Android Design
         // guidelines.

         bindPreferenceSummaryToValue(findPreference("
            notifications_new_message_ringtone"));
     }
}
```

Listing 7.5: Preferences Example

With the code listing above on which was provided by the developer shows how the specific preferences are bound within the app. As the user chooses to enter the settings activity, the following layout is provided upon request. Each of the two layout preference headers is used in conjunction with there specific layout files to produce to the user on request

```
<preference-headers xmlns:android="http://schemas.android.com/apk/
    res/android">

<!-- These settings headers are only used on tablets. -->

    // The following code snippet is used to produce the specific
       preference layout of Battery Saver
    <header
    android:fragment="adamoconnor.informe.Settings.
       SettingsActivity$DataSyncPreferenceFragment"
    android:icon="@drawable/batteryicon"
    android:title="Battery Saver" />

    // The following code snippet is used to produce the specific
       preference layout of Notification Preference.
    <header
    android:fragment="adamoconnor.informe.Settings.
       SettingsActivity$NotificationPreferenceFragment"
    android:icon="@drawable/ic_notifications_black_24dp"
    android:title="@string/pref_header_notifications" />

</preference-headers>
```

Listing 7.6: Preferences Header Layout Example

While the settings activity on which controls the following preference header's is extended to another class called AppCompatPreferenceActivity which itself extends PreferenceActivty is used to control and replace the view on which the user has chosen to use their personal preferences for the app. Users can choose the particular tone of the notification which will be sent to the user entering a Geofence. The other preferences the user of the app can choose to activate is the battery saver preference which will allow the screen of such a device to deactivate the screen display when the devices proximity sensor detects an object in front of it such as putting the device in the users pocket. This saves the battery life of the device which allows more time for exploring the InforMe@Dublin application.

7.8 Retrieving of Google's Mapping System

Within this following section of the implementation of the application is one of the most important tasks of the inforMe@Dublin application. Loading the map when the user chooses

they want to display their location and to view the geofences in the area in which they are in. The map loading quickly and smoothly with regarding the users quality of play with the app is of utmost importance.Below is two code snippets of which are used to load the Google map onto the activity of the device asynchronously. The mapping resource is used as a fragment which is displayed over the event in question.

```
mapFrag = (SupportMapFragment) getSupportFragmentManager().
    findFragmentById(map);
mapFrag.getMapAsync(this);
```

Listing 7.7: OnCreate Method Map Loading Example

Allowing for the integration of Google Maps into an Android application, a GoogleApiClient connection is needed on allowing the app to gain the necessary information to produce such a map fragment on the activity. The client of the ApiClient is initiated with the following code snippet below.

```
/**
* call the api client needed to retrieve information.
* from google such as maps and the location API.
*/
protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
    mGoogleApiClient.connect();

}
```

Listing 7.8: buildGoogleApiClient Code Snippet

While loading the map fragment which is used in the InforMe@Dublin application to set specific UI components within the application, the developer can choose whether to show specific items within the map fragment. Within this section of the implementation, the developer has chosen to disable certain settings which then enables developer's to add their buttons or UI components which are then shown on the app. Within the section which enables certain UI components, the map type can be distinguished with layouts such as:

```
mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
```

Listing 7.9: Map Types

Within the next code, listing below is how the map is asynchronously loaded within the application. The onMapReady method is called automatically when the actual map fragment is called upon.

```
/**
* creating the map on the creation of the activity.
* set each of the UI components of the map, such as
* compass, etc.
* @param googleMap
* pass the map on which is being shown on the activity.
*/
@Override
public void onMapReady(GoogleMap googleMap) {

    mGoogleMap = googleMap;
    // type of map needed to display.
    mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

    UiSettings uiSettings = googleMap.getUiSettings();
    uiSettings.setAllGesturesEnabled(true);
    uiSettings.setCompassEnabled(true);
    uiSettings.setMyLocationButtonEnabled(false);
    uiSettings.setMapToolbarEnabled(true);
    uiSettings.setZoomControlsEnabled(false);
    googleMap.setTrafficEnabled(false);

    //Initialize Google Play Services
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
            buildGoogleApiClient();
            mGoogleMap.setMyLocationEnabled(true);
        }
    } else {
        buildGoogleApiClient();
```

```
            mGoogleMap.setMyLocationEnabled(true);
        }
    }
```

Listing 7.10: Building the Map

7.9 Attaining Clients Correlative Location

Regarding the location manager of which is needed to gain access to the location of the device is as follows on Acquiring the location manager of the device in question is not done by instantiating a LocationManager directly but by rather requesting an instance from the device by calling the following method getSystemService(Context.LOCATION_SERVICE) which returns a new handler regarding a new LocationManager instance. While retrieving the user's location of the device permissions are needed to gain an accurate reading. With the following in mind, it is advised on using the GPS module together with an internet connection, With the application as follows an internet connection is needed by the user to use the app.

```
/**
* when requesting location, needs interval set and what accuracy
* the developer wants to achieve.
*/
protected void createLocationRequest() {

    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(3000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.
        PRIORITY_HIGH_ACCURACY);

    LocationSettingsRequest.Builder builder = new
        LocationSettingsRequest.Builder()
    .addLocationRequest(mLocationRequest);

    mLocationSettingsRequest = builder.build();
}
```

Listing 7.11: Location Request Settings

The code snippet above provides a reference to the location gathering of the user's device on connecting to the following activity on which the method has placed the setting of the interval with regards to the updating of the current position every couple of seconds.

```
/**
* call to location updates on when the device changes
* coordinates, it is then updated on the map.
* @param bundle
* the information of the activity.
*/
@Override
public void onConnected(Bundle bundle) {

    if (ContextCompat.checkSelfPermission(this,
    Manifest.permission.ACCESS_FINE_LOCATION)
    == PackageManager.PERMISSION_GRANTED) {
        createLocationRequest();
        LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
    }
}
```

Listing 7.12: OnConnected Location updates

7.10 Finding the Corresponding Positions of Historic areas

With regards to retrieving the geofences which are needed for the population of the UI components such as drawing a circle of the Geofence radius and the marker which is needed to see if the device has entered the Geofence in question. The snippet of code which is below shows the reference to the Firebase database which is then needed to retrieve a reference to the child nominator. The data on which is retrieved from the data snapshot is then split up and added into a HashMap.

```
/**
* creation of new firebase reference which populates an array of
    the specific,
* populates with the specific name of the area and the coordinates
*/
DatabaseReference database = FirebaseDatabase.getInstance().
    getReference();
```

```
final DatabaseReference myRef = database.child("geofences").child(
    town.toLowerCase());
myRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot alerts) {

        for (DataSnapshot alert : alerts.getChildren()) {
            String myLandmarks = alert.getValue().toString();
            //splitted name | long | lat
            String[] splited = myLandmarks.split("\\|");
            // send to the package com.example.adamoconnor.
                test02maps.LoginAndRegister; Constants Landmarks
                Hashmap.
            LANDMARKS.put(splited[0], new LatLng(Double.
                parseDouble(splited[1]), Double.parseDouble(splited
                [2])));
        }
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {

    }
});
```

Listing 7.13: Reteriving text Information of Monument

While the code above retrieves the Geofences and populates each of the historical places into a HashMap, the MapsActiviy is used in conjunction which displays each of these monuments coordinates on the map fragment. The snippet below is how these Geofences are populated. Keep in mind of which a reference of an ArrayList is then added which is populated with the information of the Landmarks Hashmap. The ArrayList in this instance is needed to see when the user's device in question is within the Geofence radius which in this case is 50 meters.

```
/*
* used for the loading of all geofences which where called from
    firebase when
* the user has signed in.
*/
for (Map.Entry<String, LatLng> entry : adamoconnor.informe.
    MapsAndGeofencing.Constants.LANDMARKS.entrySet()) {
```

```
        // creation of the geofence colour and border colour.
       // setting of each specific geofence.
        CircleOptions circleOptions = new CircleOptions()
              .center(new LatLng(entry.getValue().latitude,entry.
                 getValue().longitude))
              .strokeColor(Color.argb(50, 70,70,70))
              .fillColor(getResources().getColor(R.color.Geofence))
              .radius(50);
        mGoogleMap.addCircle( circleOptions );

        //Place current location marker
        LatLng geo = new LatLng(entry.getValue().latitude,entry.
           getValue().longitude);
        MarkerOptions markerOptions = new MarkerOptions();
         markerOptions.position(geo);
         markerOptions.title(entry.getKey());
         markerOptions.icon(BitmapDescriptorFactory.fromResource(R.
            drawable.informe));//BitmapDescriptorFactory.
            defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA)

        //adding marker to each geofence.
        mGeofenceMarker = mGoogleMap.addMarker(markerOptions);

        // add each to the geofence array list with the lat and long
           as well as the radius in meters.
        mGeofenceList.add(new Geofence.Builder()
              .setRequestId(entry.getKey())
              .setCircularRegion(
              entry.getValue().latitude,
              entry.getValue().longitude,
              adamoconnor.informe.MapsAndGeofencing.Constants.
                 GEOFENCE_RADIUS_IN_METERS)
              .setExpirationDuration(adamoconnor.informe.
                 MapsAndGeofencing.Constants.
                 GEOFENCE_EXPIRATION_IN_MILLISECONDS)
              .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
              Geofence.GEOFENCE_TRANSITION_EXIT)
              .build());
}
```

Listing 7.14: Displaying of Geofences

Concerning the above code located in listing 6.14. The reader can then understand on how each geofence is loaded and then decoded into its specific geo-location. Each location

is allocated a distinguishable marker from the Informe application with this in mind on selecting the marker of the area displays the name of the historical monument in question. The user can toggle to view these markers which open an option on getting directions of a monument of their choosing. Selecting this option will open up Google maps with the travel time etc. of the location.

7.11 A Look at the Geofencing Notification Service

When a user's device has received a location or the location has changed the developer has created a method to which calls the Geofencing request to see whether the user has entered that specific geofence on initiation. The following method is called in the code listing below.

```
/**
* used to see if user has entered a geofence.
*/
public void PopulateGeofences() {
    if (!mGoogleApiClient.isConnected()) {
        Toast.makeText(this, "Google API Client not connected!",
            Toast.LENGTH_SHORT).show();
        return;
    }

    try {
        LocationServices.GeofencingApi.addGeofences(
        mGoogleApiClient,
        getGeofencingRequest(),
        getGeofencePendingIntent()
        ).setResultCallback(this); // Result processed in onResult
            ().
    } catch (SecurityException securityException) {
        // Catch exception generated if the app does not use
            ACCESS_FINE_LOCATION permission.
    }
}
```

Listing 7.15: Population of Geofences

While reading through the different implementations of some of the most important parts of the application the following section represents how notifications are initiated when a

device enters such a location. While each of the HashMaps Geofences is loaded, they are also added to an ArrayList and sent to the GeofencingRequest method which allows for the checking if the device enters a geofence.

```
/**
* get the geofence which is requested.
* @return
* to the pending intent.
*/
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.
        Builder();
    builder.setInitialTrigger(GeofencingRequest.
        INITIAL_TRIGGER_ENTER);
    builder.addGeofences(mGeofenceList);
    return builder.build();
}

/**
* sending the notification to the user when geofence is entered.
* @return
*/
private PendingIntent getGeofencePendingIntent() {

    Log.d(TAG, "Geo fence pending intent");
    Intent intent = new Intent(this,
        GeofenceTransitionsIntentService.class);
    // We use FLAG_UPDATE_CURRENT so that we get the same pending
        intent back when calling addgeoFences()

    return PendingIntent.getService(this, 0, intent, PendingIntent.
        FLAG_UPDATE_CURRENT);
}
```

Listing 7.16: Geofence Request and Pending Intent

7.12 Retrieving Information of Historical Monuments

With regards to the implementation of loading the historical monuments within the application was coded as follows by the developer of the following project. While the user of the device enters a geofence location with the intention of a notification looming on entering the

geofence, that is produced with a pending intent. When a pending intent has then initiated a notification is then sent to the user's device of which is retrieved by the MapsActivity. If the user is using the app at the time or the user can simply choose to click the notification and be sent to the information activity which will show the subsequent history of that location. With that in mind, the name of the monument name is passed to a set variable which is located in the Place class.

```
/**
* load the specific data of the historic information
* this method retrieves the text for the activity
* with a reference to load images.
*/
private void LoadData() {

    //reference to Firebase database.
    DatabaseReference myRef = database.child("ruin");
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {

            //reference to load images method.
            informationImages();

            try {

                String info = (dataSnapshot.child(getMonumentName
                    ().trim()).getValue().toString());
                title.setText(monumentName.trim());
                String regex = "\\[|\\]";
                info = info.replaceAll(regex, "");
                information.setText(info);

            } catch(NullPointerException ex) {
            LoadData();
        }

    }

    @Override
    public void onCancelled(DatabaseError databaseError) {}

});
```

```
}
```

Listing 7.17: Loading Information of Monuments

With hence to the code located above the reader can see where the developer has provided a reference to the tree node of the Firebase database of which the following activity must have access to. After the reference connection has been completed the monument name of the intent is passed to the informationFrontFragment on which populates the information text field with the appropriate text allocated in the Firebase database. When the following information has been retrieved the developer then has to perform a regex to remove certain elements from the text on which the Firebase database add's brackets to multiple lines of text.

With regards to retrieving the images of which are related to a monument is compromised in a similar way to retrieving the child node of the parent in the database. For the population of each of the images, they are then compiled into a hash map and given a unique identity which is provided by a counter. These image URLs which are allocated by their monuments name and value of which were retrieved are then added onto a sliderLayout which will be changed every three seconds. The following code listing - 6.18 represents the loading of these images from the HashMap.

```
Hash_file_maps = new HashMap<>();

// load the images add numbers.
int count = 1;
for(DataSnapshot alert : alerts.getChildren()) {
    // pass the monument name
    Hash_file_maps.put(monumentName.trim()+", "+count, alert.
        getValue().toString());
    count++;
}

// add the images to the slider.
for(String name : Hash_file_maps.keySet()) {

    TextSliderView textSliderView = new TextSliderView(getContext
        ());

    textSliderView
```

```
            .description(name)
            .image(Hash_file_maps.get(name))
            .setScaleType(BaseSliderView.ScaleType.Fit)
            .setOnSliderClickListener(InformationFrontFragment.this);

        textSliderView.bundle(new Bundle());

        textSliderView.getBundle()
            .putString("extra",name);
        sliderLayout.addSlider(textSliderView);

    }
```

Listing 7.18: Loading Images on InformationFrontActivity

7.13 Storage of Data

With regards to the storing of the InforMe@Dublin information as well as each specific geofence location that is to be loaded onto a mapping application. Firebase is the answer to all of the questions while utilising the power of Firebase with regards to the security and anonymity of each specific user's data with reference to the users passwords and email address's all of these privacy issues are dealt with the Firebase console the developer doesnt have access to any password of which patrons of the application has set these are only reference by the unique identity key.

While the JSON data is added into the Firebase database the image below is what the InforMe@Dublin database actually look's like while the methods above show the retrieval of information from the database. Each line of these strings are associated with a corresponding parent.

Figure 7.3: Firebase Database

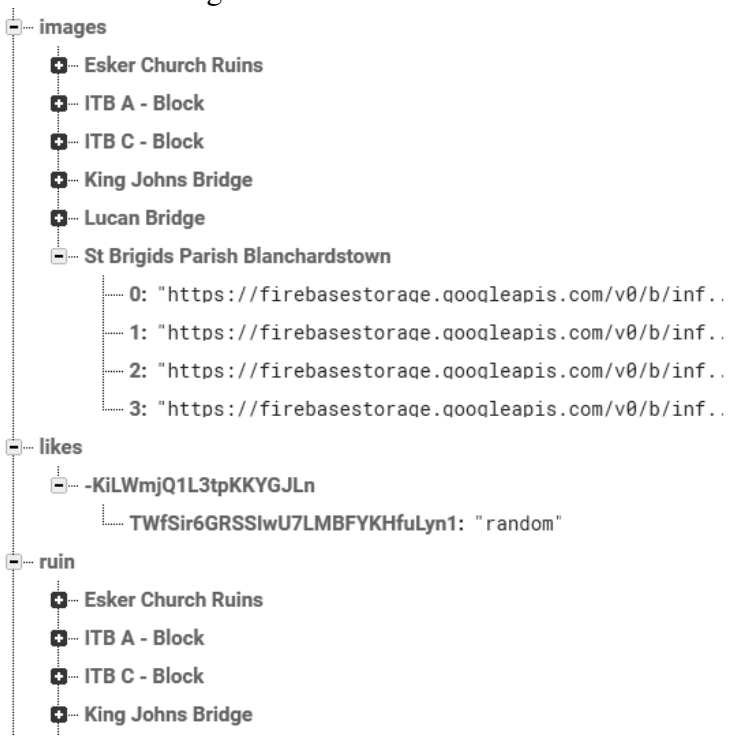

Figure 7.4: Firebase Database

7.14 Summary of the Chapter

Regarding the following chapter, the developer of the InforMe@Dublin Android application displayed and discussed some of the more important methods of the classes in the app. Most of the listings which are referenced by a unique identification number, regarding each of the methods and code snippets referenced above in the chapter the full code of the following application is supplied within the following thesis.

Chapter 8

Testing and Evaluation

8.1 Description of the Chapter

Regarding the following chapter, the developer will provide the testing of the application on which will prohibit errors running throughout at specific points and within the application. Some visual testing of the app has been conducted within which some guests were allowed to test upon the release of it to the Google play store. The smoothness of the application and latency issues, as well as any other problems which may have been discovered on completing such an app, would be delivered and results published.Testing any application which is due to be release needs vigorous testing some of the most tests related to application development are functional testing, usability testing, security testing as well as a not so complicated method of visually assessing the code of which was written. For an application to be release, a mobile application must be able to satisfy each of these testing methods.

8.2 Functional Correctness

Does the functionality of the following project meet the standards of what the project is intended for?

The answer to the question above is yes, the project functions above the expectation of the developer which is why the developer has chosen to publish the application on the Google Play Store. All parts of the program perform well and do what the developer has intended the app in question to do.

Each action of which a user will choose will do what it is supposed to do and will not receive any unknown errors unless something drastic or problematic is wrong with a device on which intends to use or install the following app. The graphical display is easily understood but could use a how-to tutorial after the creation of an account within the app for the not so experienced users.

With the exception of accepting permission's on the app as well as entering the user's details if these have not been entered an error on the text field would be shown graphically to the user.

8.3 Loading of the Application

Regarding the fast usability of the InforMe@Dublin application with regards to latency related issues, the following application has none of this kind. While the developer was coding the loading up of the application and authentication, the user just has to wait until the progress dialogue completes which means the user is in the process of being logged into the app which helps as a user is not trying to access certain UI elements while a long process is being completed. Overall the application has a smooth feel to it with no issues being related to latency issues. This may happen on some device with a low amount of ram available to the device.

8.4 Loading of the Map

While the map loads on the mobile device a camera animation is performed this allow's for a nice smooth feel to the user's location. Utilising these actions on the device leave's users with a great feel while accessing the app as it seems good quality overall with regards to scoping out the location of the device. Only errors will occur if the device cannot receive the geofences on which need to be received from the Firebase database.

8.5 Loading of the Information

Concerning loading the information of each specific Geo-lateral location while the developer was coding the application some latency issues were happening with regards to the API the developer was using which allows for the information to be justified and evenly spaced. Using the following class which is available to integrate with any Android project can be found on Github by bluejamesbond called TextJustify-Android. The following class although nice in styling provides text being loaded only at certain times which means that the development of this class has not been completed. With that in mind, the developer had found an alternative which is called JustifiedTextView by Navabi together while using the following class and using the Picasso API provided the needed processing power for the app not to become sluggish or not to display certain aspects of the activity.

8.6 Efficiency of the Application

The efficiency of the project as a whole is that is above that of what the developer expected. Considering the number of connections that could be going on at the same time regarding accessing the Firebase database and retrieving the authenticity of the users with their unique identity tags as well as processing each of the geo-locations which on opening the map activity is loaded asynchronously in such a fast and smooth manner. The efficiency of the app is produced by using worker threads in conjunction with the UI-thread of the Android device on which the application is installed this allows for multiple processes to be done simultaneously. When developing the application, the developer noticed some latency and concurrency issues while developing such an application concerning the loading of images which was done by using Picasso which allows for the downloading of images smoothly and in a synchronous type fashion, Picasso uses Automatic memory and disk caching for all images loaded with the following API. Also, async threads are also used to create a smooth feel while a user is utilising such an app.

While using the application, the device can use the mobile's data on which is fast enough for utilising an application of the following standard. Visiting different monuments within the application allows for the caching of these images and information this saves on the data

usage of the application, The developer tested this by using the application over a period of time and of which the application only used 30mb of data.

8.7 Usability of the Application

A set of tasks were given to each user of the following application's quality and usability with regards to the testing of the application. The developer compiled a set of task for each user to complete. Testing of the application needed to be throughout as the developer did not want users of the application to be hit with multiple app crashes. Also, the application needed to be easily enough to use as people of all ages could use the InforMe@Dublin app. Using the program should be easily read and perform as intended as the developer promised.

8.8 Tasks on using the Application for a Tester

1. Logging into the InforMe@Dublin app.
2. Creation of an Account.
3. Changing notification sounds.
4. Changing power saving option.
5. Toggle focusing on devices location.
6. Adding new geofences which will be sent to the developer.
7. Using the compass of the application.
8. Choosing a geofence marker on which users can get directions using Google Maps.
9. Entering a geofence to continue onto the information of a monument.
10. Choosing text-to-speech allowing the user to listen to the information of the monument.
11. Looking at users comments.

12. Liking users comments.
13. Adding comments.
14. Removing comments.
15. Updating comments.
16. Logging out of the application.

8.9 Tester 01

The followings testers background is as follows. Tester one's job title is that of a graphic designer who works within the mobile phone industry within Dublin city centre. The tester is 23 years old uses mobile phones every day and has a Samsung Galaxy 6S which boasts an astonishing number of high specifications such as a whopping 3GB of Ram. The tester downloaded and installed the InforMe@Dublin application on their device and a set of tasks where given.

8.9.1 Graphic Display

It is very professional looking. The layout is clear and informative. Very impressive for a student project.

8.9.2 Usability

I found the app user-friendly. Straight forward and easy to use. Anyone of any age would be able to navigate around the app successfully.

8.9.3 Additional Comments

Would love to see more locations. Keep up the good work

8.10 Tester 02

Tester two, the next tester is a male of 48 who's occupancy is a factory worker. Working for one of Ireland's most prestigious paint companies located in Celbridge but lives in Lucan Co.Dublin. The testers device is an Alcatel Idol 4, which is a budget mobile device. The tester downloaded and installed the InforMe@Dublin application on their device and a set of tasks where given.

8.10.1 Graphic Display

The display is quite nice. Overall there is a good feel to this app.

8.10.2 Usability

It works, the app does on what the developer intended it to do. I might even use it while going on walks to learn about our area.

8.10.3 Additional Comments

Best of luck in your future endeavours.

8.11 Tester 03

Tester three's background is of a professional fitness tutor working closely with the ncef and of which is the owner of their own business within the fitness sector. The device on which this tester has used the app on is a Sony Xperia XA. The tester downloaded and installed the InforMe@Dublin application on their device and a set of tasks where given.

8.11.1 Graphic Display

The Graphical display is a superb standard compared to other apps on the play store. I find this application to be very appealing.

8.11.2 Usability

The functionality of this app is great walking around and looking at different aspects of history really makes you think and realise of what little other people had compared to what is in the same location now. The feel of the layout is quite neat.

8.11.3 Additional Comments

I would like to wish the developer of InforMe@Dublin all the best with this app. As I see it going far into the future.

8.11.4 Security Testing

The following application InforMe@Dublin uses Firebase for authentication of each user of the application. While using their email and password or Google authentication, the user of the app must create an account on the app on which they must provide their name and profile image. On creation of an account, the user is then added to the Informe Firebase database. While the user has entered the app, a check is done on which compromises of finding the unique id of the applicant in the database, if no user is found then an account is needed to be created. While testing the application regarding the login of a user concerning email and password authentication works a treat as the developer can see the database updating with the user's credentials on login and creation of an account. With regards to Google authentication, while adding the following application onto the Google play store, the developer had to retrieve the SHA-1 key used for the app release. Each API used within the development of the app a Signing-certificate fingerprint or the SHA-1 is added to each API licence.

8.12 Visual Testing

Visual testing of the InforMe@Dublin application was used to verify that the program would work as intended by the developer. While developing different parts of the application, the author would run these specific parts to see if the following code would conduce with other parts of the project to produce the outcome on which the developer was trying to achieve.

While running the application, the developer could then see if the standard was paired with the code of which was written. Other standards were also needed which included exception classes to allow for errors to generate with an exception. Each part of the InforMe@Dublin project was overseen by the following method of testing on which was the most efficient, cost-effective and non-time consuming for the developer at that specific time.

While also using the following testing method with the following application be developed on an Android mobile device on which allowed for the developer to use the specific developers debugging tools of which showed the usage of the CPU. Using the application as well as the GPU renderings of the application regarding the graphics of which are needed or used within the app.

8.13 Summary of Chapter

Regarding the following chapter, the developer of the InforMe@Dublin Android application has discussed the different methods and problems of which arrived while completing the application. Different test methods used such as real user experience as well as using the developers own initiative in regards to which method does what and following the methodology of which was chosen by the developer. The selected methodology allows for only moving on from one correct coding function to another when it has been extensively validated. The next chapter in this thesis is about the future work which may be undertaken by the developer.

Chapter 9

Conclusions and Recommendations for Further Work

9.1 Summary and Main Conclusions

Following the review of literature which is located in (Chapter 3), four main research questions were identified regarding the project as a whole.

1. ***Is there a need for an application like this on the marketplace?***
2. ***Is there a way to save battery life while using GPS / Internet connection within an android application?***
3. ***Will location services, and mobile internet take much battery or data?***
4. ***Will the APP provide a wiki type interface where visitors can add to the information about a historic site?***

9.1.1 Developments and Findings Relating to Research Objective 1

With regards to the following question on which the author is to provide an answer to is yes there is quite a market for this application. The developer of InforMe@Dublin has received many words of praise before even releasing an app of the following standard. Researching application's of this type and standard throughout the Google Play market are of non-existence the developer of which had released a small survey to the general public of Lucan

as a pilot scheme on distinguishing if the idea of an application within its category would fit in with their need's. Utilising a survey website called Survey Monkey was used in the creation of the survey which was published to the people of Lucan.

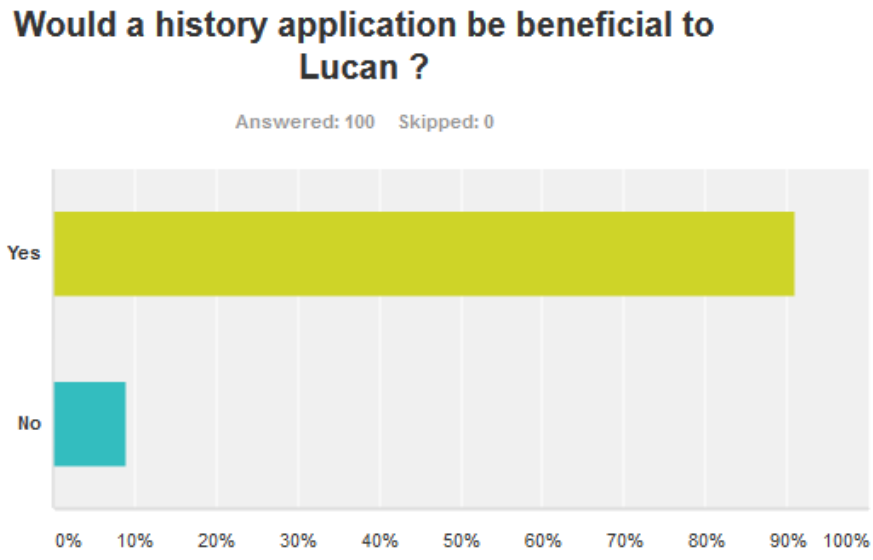

Figure 9.1: Need for InforMe@Dublin

Regarding figure 9.1 the developer has shown one of the results on which were retrieved from the survey of which suggests that the people of Lucan would love to see an application of the following stature.

Answer Choices	Responses	
Yes	91.00%	91
No	9.00%	9
Total		100

Figure 9.2: Number of Patrons who Completed Survey

The number of patrons who took part in the survey with relation to seeing if the InforMe@Dublin application was a good choice. 91 out of the 100 patrons who completed the survey said that they would be interested in such an application while nine said no. This

could be that some of the patrons who took part in the survey were not actually from Dublin and may live in other area's or that the specific audience of which was targeted wasn't the best choice such as 15 - 19 year old's.

An application which is related to the InforMe@Dublin that the developer has found was produced for the telling of information of the 1916 rising located here in Dublin City Centre the app of which was developed by Failte Ireland. The Dublin Discovery Trails application is for the use of guided tours within the city centre itself which tells the story of Dublin over 1000 years. The app which use's devices location but stores all necessary files on the application itself and is only useful in Dublin city before going on these walks the application prompts user's to download a specific audio and image files which isn't much use if the patron is on the go.

9.1.2 Developments and Findings Relating to Research Objective 2

Regarding objective two which is related to the life of a battery while using location services as well as the internet connectivity. The battery life can be saved by using Google Play Services of which provide a battery manager of which is provided within the application to provide the best and most suitable power/accuracy ratio on gaining the location of an Android device.

The answer is yes. The battery life of an Android phone can be optimised by being smart with the development of the application. Breaking each part into specific smaller pieces allowing threads to do individual tasks will help the life of the users battery, Also by using less services such as not calling all of the Google services packages will assist the battery life by using only a particular package. Developers (2016)

As the following application is to be used for educational purposes and doesn't ambit the need for adverts of which provides 75% of application related battery drain. The following study was provided by Purdue University in partnership with Microsoft. These studies were concluded by looking at free application's located in the Google Play Store and the Apple store. As ad-serving provides its advertisers with valuable data from many patrons of many applications. In a nutshell advertiser's cover the cost of the application which is why it is delivered free to the user. The problem of these Ad-serving SDK's are that they are in fact

poorly coded with regards to sending the information back to servers as well as the activity on the screen.Donnelly (2012)

Also while using Firebase on which is incorporated within the Android application allows for the saving of some data on which has been loaded from the database this allows for the device to show this data which doesn't need to be parsed and downloaded again.

9.1.3 Developments and Findings Relating to Research Objective 3

While retrieving the location of a mobile device the following information was retrieved from a source which provides each of the location techniques, as well as the amount of power each of these, utilise while in use. This information was gathered when the author was researching different technologies that would be used or potentially used in the application.

Table 9.1: Battery Usage of Positioning Techniques Bareth and Kupper (2011)

Technologies	*Accuracy*	*Precision*	*Energy*
GPS	10m	95%	6.616Ws
WiFi	50m	90%	2.852Ws
Cellular-ID	5km	65%	1.013Ws

The following screen shot located in figure 9.3 was taken from the test device of which the InforMe@Dublin application was developed on. In the figure below shows how much data of which the application had consumed over the hour test. Below shows that on average the application does not exactly use that much data compared to other applications over an hour interval.

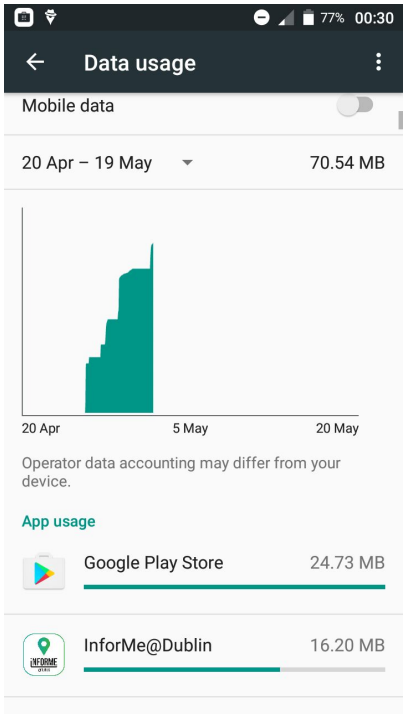

Figure 9.3: Data usage of InforMe@Dublin

9.1.4 Developments and Findings Relating to Research Objective 4

During the development of such an application, the developer decided on introducing a different type Wiki feel to the application which allows the users to contribute to the information on such areas by posting different images of that area whether that being new or old images. While each patron has added their individual information of the area. Other users can then like posts of which have been created. Only that specific user who created the post can remove and update it on the application if they wish to re-visit the Geofence location. A Screen-shot of the applications wiki type feel is shown below in figure - 9.4

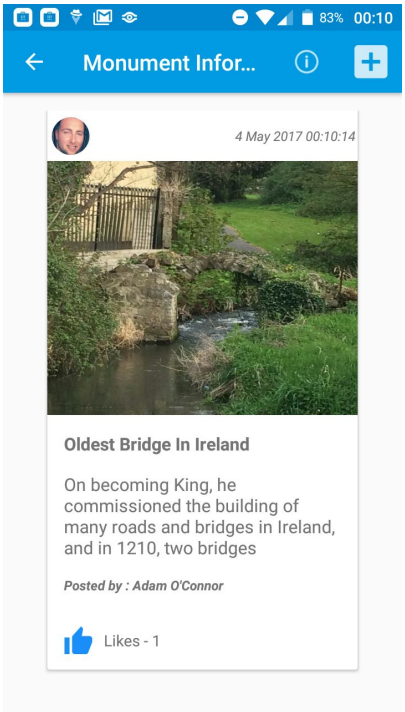

Figure 9.4: Wiki Type Activity

9.2 Recommendations for Further Work

With regards to the developer continuing with the development of the Android application InforMe@Dublin. The developer as follows thinks that the project as a whole is worthy of more development bringing the app to even more devices as it is only limited to Android Marshmallow devices and above. With the future of technology where it is today, there is a significant opportunity for providing an application to other users not just specifically to Dublin so the Informe app can be expanded and provided to people all over Ireland.

the following future work objectives can be completed.

1. Providing more historical monuments to the application.
2. Adding a new way of users to add a Geofence.
3. Creation of specific profile pages of user's.

4. Providing more cost effective ways on users using the application with a connection to the internet.

5. Provide more power efficient code which allows users to explore more.

6. Adding a how to page or tutorial on how to use the InforMe@Dublin application.

Chapter 10

Personal Reflections

Regarding the following project InforMe@Dublin, the developer has expressed Great enjoyment on creating an application of the following standard. The application was a joy on completing which allowed the developer to show the skills on which he has learnt over the past four years studying computer science at the Institute of Technology Blanchardstown. Regarding the skills needed for completing such an app of a high standard and the progressing to the release of an app which is the only one of its kind located in the Google play store. Some problems faced throughout the development of such an application where the concurrency issues of which related to the accessing of the Firebase database and populating the monument fields, Geofence locations and data on which other users have added. Some of these methods on which needed access to the database of which had to be slowed down on processing the data within the app. It was fun creating an app from scratch seeing a theory in the developers head into becoming a fully functional Android application. The developer would like to update the app into the distant future continually and updating the geofences of different area's but also to include a website on which geofence's can be added and updated as well as uploading the associated images.

10.1 Fingal County Council Feedback

Concerning sending the InforMe@Dublin application to Fingal county council, the developer received some feedback which was good news. The person who is an editor and project manager of Dublin.ie thought that the application was quite good considering it's a student

project. On first installing the application the user had a problem with signing into the app which was a weird error considering the installation was tested on a few devices. Anthony Mc Guinness wished the developer all the best of luck on developing the InforMe@Dublin Android Application.

10.2 Dublin City County Council Feedback

Regarding receiving feedback from Dublin City Council, the developer was directed to the Mobile Device section, of which was notified of the application due to a contract that is between the council and their work mobiles they cannot install unknown applications. Because this application doesn't have enough recognition with regards to known security risk factors they cannot download these unknown applications even though the application is on the Google Play Store and passed all security screening. While the application uses the mobile's location and access to photographs and media files, it cannot be installed on their device hence forfeiting a contract of which they have made with the mobile provider.

Chapter 11

Appendix A

11.1 InforMe@Dublin Screen Shots

In the following section of the appendices, the developer has submitted screen shots of the InforMe@Dublin. Each page of the application is presented below just as an aid to the reader on understanding what the app presents to the users that download it from the Google Play Store.

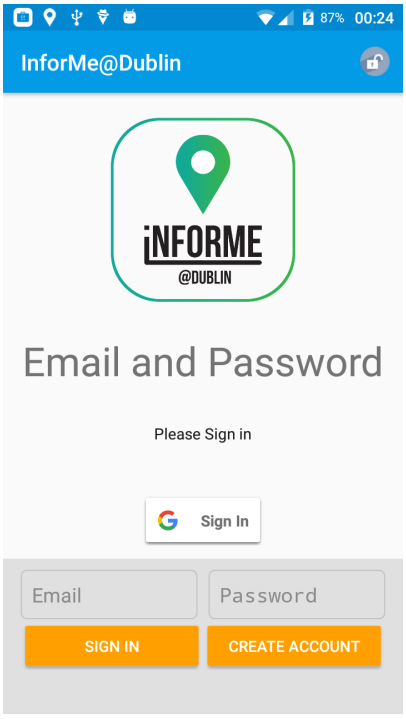

Figure 11.1: Login Page

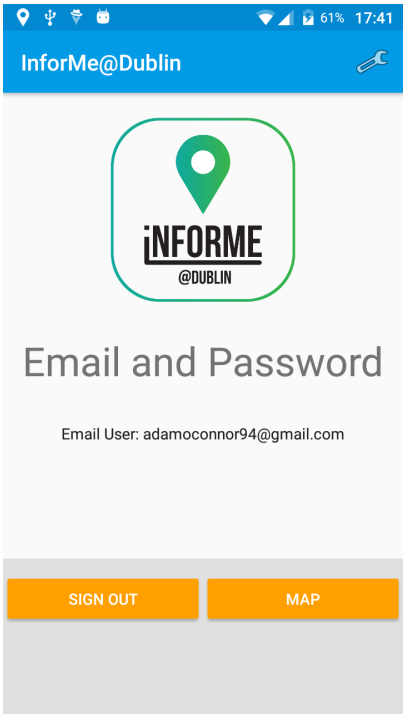

Figure 11.2: Logged in Page

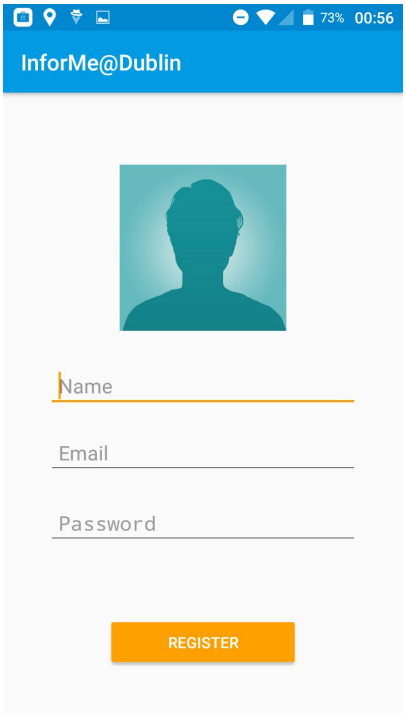

Figure 11.3: Register

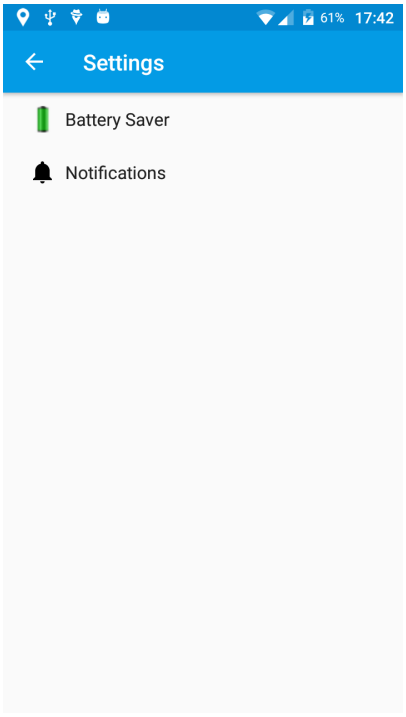

Figure 11.4: Settings Preference

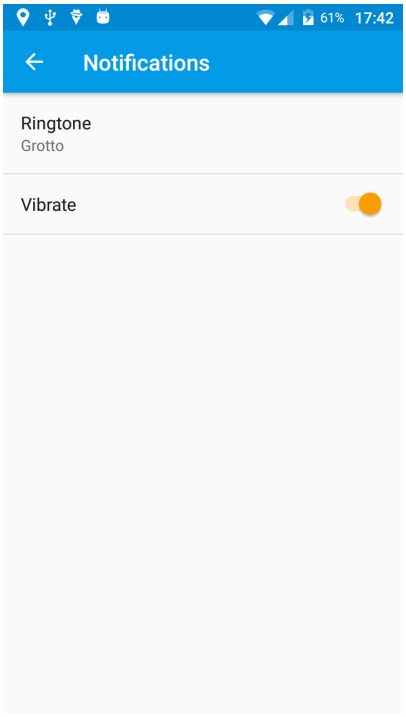

Figure 11.5: Notification Settings

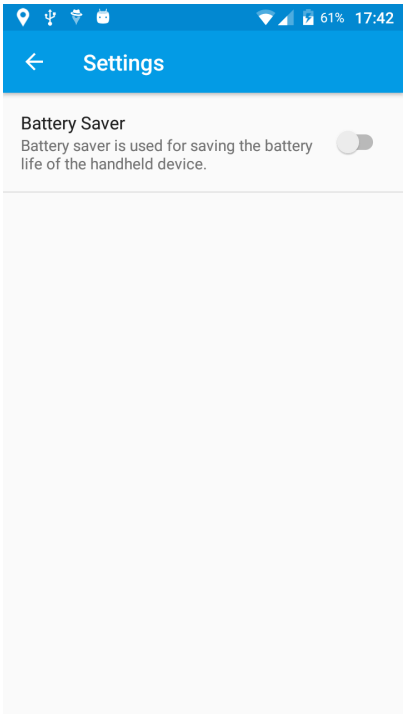

Figure 11.6: Battery Settings

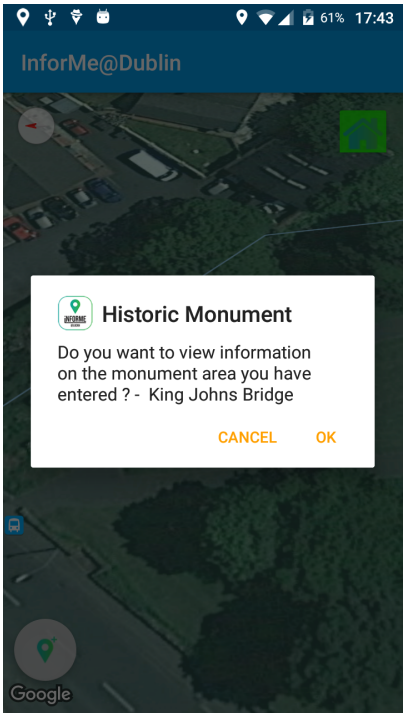

Figure 11.7: Geofence Enter

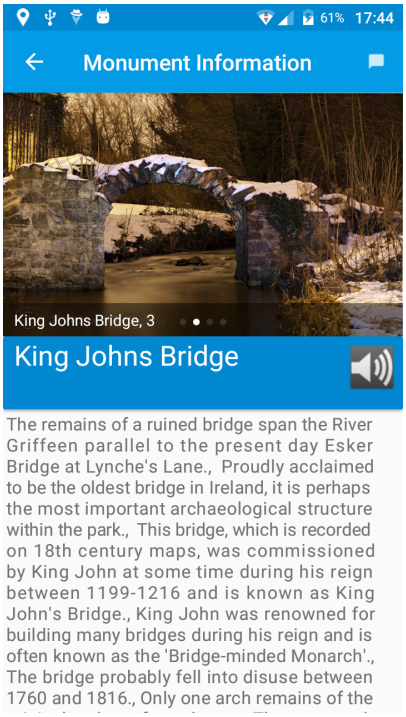

Figure 11.8: Information Page

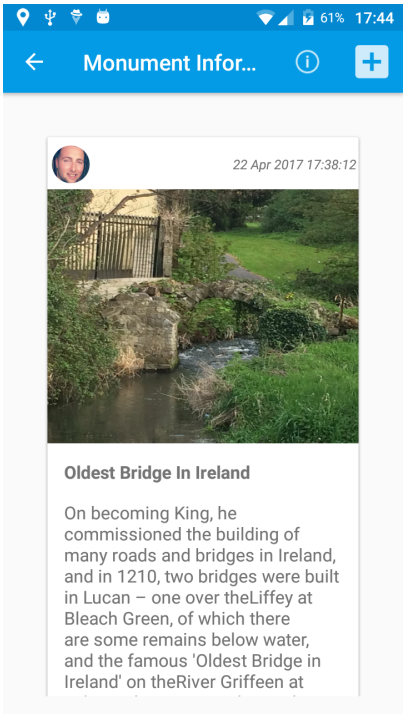

Figure 11.9: Display Posts

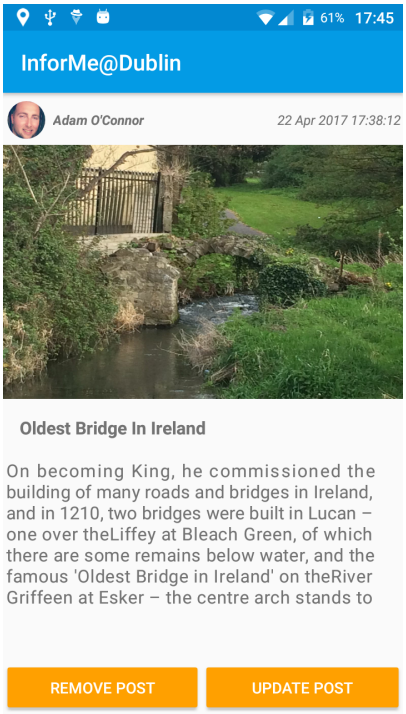

Figure 11.10: Entering Post

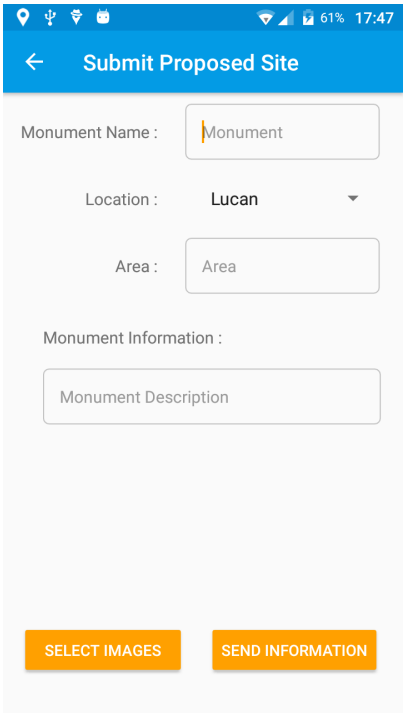

Figure 11.11: Adding Geofence

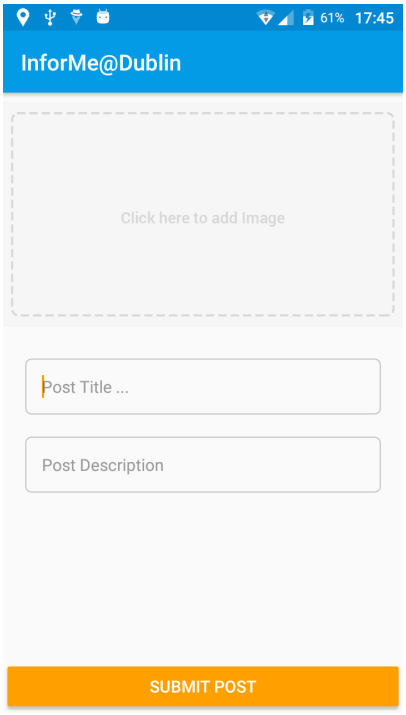

Figure 11.12: Add Post

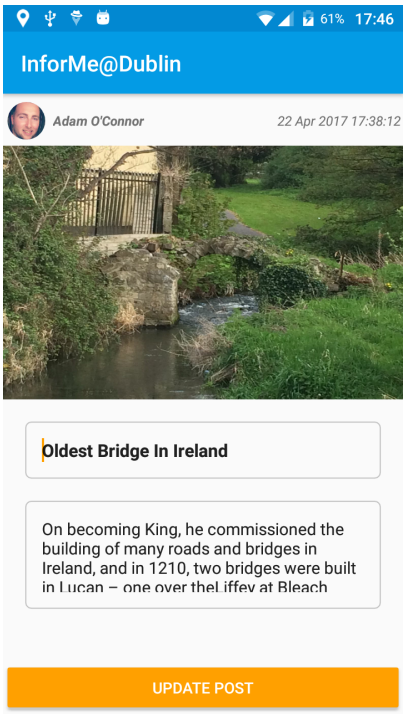

Figure 11.13: Updating Post

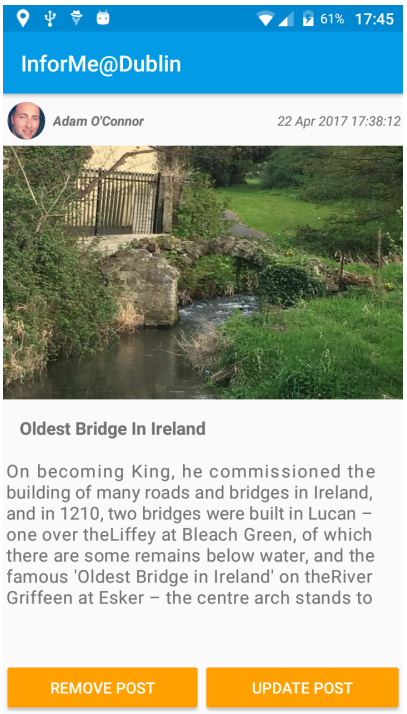

Figure 11.14: Entering Post

Chapter 12

Appendix B

12.1 InforMe@Dublin Code Listings

In the following section of the appendices, the developer has submitted the code on which the InforMe@Dublin application was developed each section is split into its specific packages which are only related to its section such as Logging into the app, SettingsPreferences, Map&Geofencing and also adding posts on a monument in the area.

12.2 Login&Register

12.2.1 EmailPasswordAuthentication

```
package adamoconnor.informe.LoginAndRegister;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
```

```
import android.support.v4.content.ContextCompat;
import android.text.TextUtils;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import adamoconnor.informe.MapsAndGeofencing.MapsActivity;
import adamoconnor.informe.R;
import adamoconnor.informe.Settings.CheckConnectivity;
import adamoconnor.informe.Settings.SettingsActivity;
import com.google.android.gms.auth.api.Auth;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.auth.api.signin.GoogleSignInResult;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.List;
import java.util.Locale;
import static adamoconnor.informe.MapsAndGeofencing.Constants.
    LANDMARKS;
import static adamoconnor.informe.R.id.resetPassword;
import static adamoconnor.informe.R.id.settings_prefs;

public class EmailPasswordAuthentication extends Progress
    implements
        View.OnClickListener {
```

```
private static final String TAG = "EmailPassword";

// fields located in design
private TextView mStatusTextView;
private EditText mEmailField;
private EditText mPasswordField;
private SignInButton mGoogleLogin;

//states used for the settings bar when user or out.
private String mState = null; // setting state
private String userState = null;
private CheckConnectivity checkConnectivity;

// declare auth.
private FirebaseAuth mAuth;

//used for firebase database of users.
private DatabaseReference mDatabaseUsers;

// declare auth listener.
private FirebaseAuth.AuthStateListener mAuthListener;

// used to see only one signed in.
private static final int RC_SIGN_IN = 1;
private GoogleApiClient mGoogleApiClient;

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_emailpassword);

    //setting screen orientation to stop fragments view showing
        on eachother.
    this.setRequestedOrientation(ActivityInfo.
       SCREEN_ORIENTATION_PORTRAIT);

    // declare progress dialog
    mProgressDialog = new ProgressDialog(this);
    // declare connectivity checker
    checkConnectivity = new CheckConnectivity();

    // Views
    mStatusTextView = (TextView) findViewById(R.id.status);
    mEmailField = (EditText) findViewById(R.id.field_email);
    mPasswordField = (EditText) findViewById(R.id.
       field_password);
```

```
// Buttons
findViewById(R.id.email_sign_in_button).setOnClickListener(
    this);
findViewById(R.id.email_create_account_button).
    setOnClickListener(this);
findViewById(R.id.sign_out_button).setOnClickListener(this)
    ;
findViewById(R.id.to_map_button).setOnClickListener(this);
findViewById(R.id.googleSignIn).setOnClickListener(this);

// initialization of firebase authentication.
mAuth = FirebaseAuth.getInstance();

// instanciation of database instance in the users area.
mDatabaseUsers = FirebaseDatabase.getInstance().
    getReference().child("users");
// keep database synced.
mDatabaseUsers.keepSynced(true);

// authentication state listener
mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth
        firebaseAuth) {
        FirebaseUser user = firebaseAuth.getCurrentUser();
        if (user != null) {
            // User is signed in
            Log.d(TAG, "onAuthStateChanged:signed_in:" +
                user.getUid());
            checkUserExistsDatabaseOnReEnter();

        } else {
            // User is signed out
            Log.d(TAG, "onAuthStateChanged:signed_out");
        }
        // update the UI
        updateUI(user);
    }
};

checkLocationPermission();

// Configure Google Sign In
```

```
        GoogleSignInOptions gso = new GoogleSignInOptions.Builder(
            GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.
                    default_web_client_id))
                .requestEmail()
                .build();

        // creation of a google api client for the connection of a
            google token.
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .enableAutoManage(this, new GoogleApiClient.
                    OnConnectionFailedListener() {
                     @Override
                     public void onConnectionFailed(@NonNull
                         ConnectionResult connectionResult) {

                     }
                })
                .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
                .build();
    }

    /**
     * Used for the signing in of a google authentication
     * token.
     */
    private void signIn() {
        mProgressDialog.setMessage("Signing in ...");
        mProgressDialog.show();
        Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(
            mGoogleApiClient);
        startActivityForResult(signInIntent, RC_SIGN_IN);
    }

    /**
     *
     * @param requestCode
     * checking the allocated code on which allows for the sign in.
     * @param resultCode
     *  not really used in this instance.
     * @param data
     *requesting the information needed to retrieve the information
     * used for the signing in to the application.
     */
    @Override
```

```
public void onActivityResult(int requestCode, int resultCode,
    Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from
       GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        // Get the intent of the data used for the sign in.
        GoogleSignInResult result = Auth.GoogleSignInApi.
           getSignInResultFromIntent(data);
        if (result.isSuccess()) {
            // Google Sign In was successful, authenticate with
                Firebase
            GoogleSignInAccount account = result.
               getSignInAccount();
            firebaseAuthWithGoogle(account);
        } else {
            // Google Sign In failed, update UI appropriately
            Toast.makeText(EmailPasswordAuthentication.this, "
               Authentication failed.",
                    Toast.LENGTH_SHORT).show();
            mProgressDialog.dismiss();

        }
        mProgressDialog.dismiss();
    }
}

/**
 * @param acct
 * retrieving credentials used in the sign in
 * and to authenticate the user on which will
 * be using the application.
 */
private void firebaseAuthWithGoogle(GoogleSignInAccount acct) {
    Log.d(TAG, "firebaseAuthWithGoogle:" + acct.getId());

    AuthCredential credential = GoogleAuthProvider.
       getCredential(acct.getIdToken(), null);
    mAuth.signInWithCredential(credential)
            .addOnCompleteListener(this, new OnCompleteListener
               <AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult
                   > task) {
```

```
                        Log.d(TAG, "signInWithCredential:onComplete
                            :" + task.isSuccessful());

                        // If sign in fails, display a message to
                            the user. If sign in succeeds
                        // the auth state listener will be notified
                             and logic to handle the
                        // signed in user can be handled in the
                            listener.
                        if (!task.isSuccessful()) {
                            Log.w(TAG, "signInWithCredential", task
                                .getException());
                            Toast.makeText(
                                EmailPasswordAuthentication.this, "
                                Authentication failed.",
                                    Toast.LENGTH_SHORT).show();
                        }else {
                            checkUserExistsDatabaseOnReEnter();

                        }
                    }
                });
        // stop progress dialog.
        mProgressDialog.dismiss();
    }

    // check permission request of 99
    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;

    /**
     * used to check the location permission
     * @return
     * dialog if user has not set permissions
     */
    public boolean checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale
                (this,
                        Manifest.permission.ACCESS_FINE_LOCATION)) {
```

```
                // Show an expanation to the user *asynchronously*
                   -- don't block
                // this thread waiting for the user's response!
                   After the user
                // sees the explanation, try again to request the
                   permission.

                //Prompt the user once explanation has been shown
                //(just doing it here for now, note that with this
                   code, no explanation is shown)
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.
                           ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);


            } else {
                // No explanation needed, we can request the
                   permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.
                           ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);
            }
            return false;
        } else {
            return true;
        }
    }

    /**
     *
     * @param requestCode
     * start to see if user connected
     * @param permissions
     * permissions which are needed for the application
     * @param grantResults
     * if the user has granted access to sensors.
     */
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[],
                                             int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_LOCATION: {
```

```
                // If request is cancelled, the result arrays are
                    empty.
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.
                            PERMISSION_GRANTED) {

                    // permission was granted, yay! Do the
                    // location-related task you need to do.
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.
                                ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {
                    }

                } else {

                    // permission denied, boo! Disable the
                    // functionality that depends on this
                        permission.
                    checkLocationPermission();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

    /**
     * used to get the location of the device in order
     * to populate geofences for the specific area...
     * @return
     * used to return the last known coordinates of the
     * device.
     */
    public Location getLocation() {
        //check permission before using.
        checkLocationPermission();
        LocationManager locationManager = (LocationManager) this.
            getSystemService(Context.LOCATION_SERVICE);
        if (locationManager != null) {
            Location lastKnownLocationGPS = locationManager.
                getLastKnownLocation(LocationManager.GPS_PROVIDER);
            if (lastKnownLocationGPS != null) {
```

```
                return lastKnownLocationGPS;
            } else {
                Location loc =  locationManager.
                    getLastKnownLocation(LocationManager.
                    PASSIVE_PROVIDER);

                return loc;
            }
        } else {
            return null;
        }
    }

    /**
     * used for the adding of geofences to the map activity.
     * populates the landmarks array, which gets geofences ready
        for
     * the map when a user has been authenticated
     */
    public void addGeofences() {

        Geocoder gcd = new Geocoder(getApplicationContext(), Locale
            .getDefault());
        List<Address> addresses = null;
        String town = "dublin";

        // used in further work for specific areas around Ireland.
       /* try {
            addresses = gcd.getFromLocation(getLocation().
                getLatitude(), getLocation().getLongitude(), 1);
        }catch (IOException ex) {
            ex.getStackTrace();
        }

        if (addresses.size() > 0)
        {
            town = addresses.get(0).getLocality();

        }
        */
        //town.toLowerCase() used to get current users location in
            the future of InforMe@Ireland

        /**
```

```
         * creation of new firebase reference which populates an
            array of the specific,
         * populates with the specific name of the area and the
            coordinates
         */
        DatabaseReference database = FirebaseDatabase.getInstance()
            .getReference();
        final DatabaseReference myRef = database.child("geofences")
            .child(town.toLowerCase());
        myRef.addListenerForSingleValueEvent(new ValueEventListener
            () {
            @Override
            public void onDataChange(DataSnapshot alerts) {

                for (DataSnapshot alert : alerts.getChildren()) {
                    String myLandmarks = alert.getValue().toString
                        ();
                    //splitted name | long | lat
                    String[] splited = myLandmarks.split("\\|");
                    // send to the package com.example.adamoconnor.
                        test02maps.LoginAndRegister; Constants
                        Landmarks Hashmap.
                    LANDMARKS.put(splited[0], new LatLng(Double.
                        parseDouble(splited[1]), Double.parseDouble(
                        splited[2])));
                }
            }
            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });

    }

    // creating an options menu for settings page
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();

        // resetting password when user not signed in.
        if(id == resetPassword) {

            Intent intent = new Intent(this, ResetActivity.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
            startActivity(intent);
            return true;

        }

        // setting preferences when user signed in.
        if(id == settings_prefs) {

            Intent intent = new Intent(this, SettingsActivity.class
                );
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            return true;

        }
        return false;
    }

    //create options menu when user is signed in or signed out.
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if
            it is present.
        MenuInflater inflater = getMenuInflater();

        if (mState.equals("show")) {

            // set the previous action view to non visible.
            for (int i = 0; i < menu.size(); i++)
                menu.getItem(i).setVisible(false);

            inflater.inflate(R.menu.pref_custom, menu);

            //set a new action to visible.
            for (int i = 0; i < menu.size(); i++)
                menu.getItem(i).setVisible(true);
        }
        if(userState.equals("SHOWUSER_SETTINGS")) {

            // set the previous action view to non visible.
            for (int i = 0; i < menu.size(); i++)
                menu.getItem(i).setVisible(false);

            inflater.inflate(R.menu.settings_custom, menu);

            //set a new action to visible.
```

```
            for (int i = 0; i < menu.size(); i++)
                menu.getItem(i).setVisible(true);
        }

        return true;
    }

    //START on_start_add_listener
    @Override
    public void onStart() {
        super.onStart();
        mAuth.addAuthStateListener(mAuthListener);
    }

    //START on_stop_remove_listener
    @Override
    public void onStop() {
        super.onStop();
        if (mAuthListener != null) {
            mAuth.removeAuthStateListener(mAuthListener);
        }
    }

    /**
     * used for the changing of activity to the
     * Map Activity.
     */
    private void myMap() {

        // check if the internet and location is turned on.
        if(!checkConnectivity.isInternetOn() && !checkConnectivity.
            isLocationOn()) {
            finish();
        }

        // adding of geofences.
        addGeofences();

        // send user to map activity.
        Intent intent = new Intent(EmailPasswordAuthentication.this
            , MapsActivity.class);
        startActivity(intent);
    }

    /**
```

```
     * used for the signing in of a user which has logged in
     * with their email address and password.
     * @param email
     * email address of the user which created an account.
     * @param password
     * password on which user wants to use.
     */
    private void signIn(String email, String password) {
        Log.d(TAG, "signIn:" + email);
        // validate the textfields user has filled in.
        if (!validateForm()) {
            return;
        }

        showProgressDialog();

        // start the authentication of the email address and
           password of the user.
        final Task<AuthResult> authResultTask = mAuth.
           signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, new OnCompleteListener
                   <AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult
                       > task) {
                        Log.d(TAG, "signInWithEmail:onComplete:" +
                            task.isSuccessful());

                        if(task.isSuccessful()) {
                            checkUserExistsDatabaseOnReEnter();
                        }
                        // If sign in fails, display a message to
                           the user. If sign in succeeds
                        // the auth state listener will be notified
                            and logic to handle the
                        // signed in user can be handled in the
                           listener.
                        if (!task.isSuccessful()) {
                            Log.w(TAG, "signInWithEmail:failed",
                                task.getException());
                            Toast.makeText(
                                EmailPasswordAuthentication.this, R.
                                string.auth_failed,
                                      Toast.LENGTH_SHORT).show();
```

```
                        mStatusTextView.setText(R.string.
                            auth_failed);
                    }

                    hideProgressDialog();
                }
            });
    }

    /**
     * used for the signing out of the user.
     */
    private void signOut() {
        mAuth.signOut();
        updateUI(null);
    }

    /**
     * check on whether the user that has been authenticated is
     * in the database and has created an account.
     */
    private void checkUserExistsDatabaseOnReEnter() {

        // see if user is authenticated.
        if(mAuth.getCurrentUser() != null) {

            // get the id of the authenticated user.
            final String user_id = mAuth.getCurrentUser().getUid();

            // see if the authenticated user is found in the
               firebase database.
            mDatabaseUsers.addValueEventListener(new
               ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot)
                    {
                    if(dataSnapshot.hasChild(user_id)) {

                    }
                    else
                    {

                        Intent intent = new Intent(
                            EmailPasswordAuthentication.this,
                            SetupActivity.class);
```

```
                        intent.setFlags(Intent.
                            FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(intent);

                        Toast.makeText(EmailPasswordAuthentication.
                            this, "Please setup your account...",
                                Toast.LENGTH_SHORT).show();
                    }
                }

                @Override
                public void onCancelled(DatabaseError databaseError
                    ) {

                }
            });
    }

}

/**
 * validating the text-fields on which the user
 * must enter their personal information.
 * @return
 * return the error on screen for the text-field.
 */
private boolean validateForm() {
    boolean valid = true;

    String email = mEmailField.getText().toString();
    if (TextUtils.isEmpty(email)) {
        mEmailField.setError("Required.");
        valid = false;
    } else {
        mEmailField.setError(null);
    }

    String password = mPasswordField.getText().toString();
    if (TextUtils.isEmpty(password)) {
        mPasswordField.setError("Required.");
        valid = false;
    } else {
        mPasswordField.setError(null);
    }
```

```
        return valid;
    }

    /**
     * show the login details to next activity !!
     * @param user
     * the following updates the login UI which sees
     * if the user is signed in or not.
     */
    private void updateUI(FirebaseUser user) {
        hideProgressDialog();
        if (user != null) {
            mStatusTextView.setText(getString(R.string.
                emailpassword_status_fmt, user.getEmail()));

            //check the connectivity of the internet and location.
            checkConnectivity.startInternetEnabled(this);
            checkConnectivity.startLocationEnabled(this);

            mState = "show";
            userState = "HIDE_MENU";

            // refresh the action bar menu.
            invalidateOptionsMenu();

            // reset buttons.
            findViewById(R.id.email_password_buttons).setVisibility
                (View.GONE);
            findViewById(R.id.email_password_fields).setVisibility(
                View.GONE);
            findViewById(R.id.googleSignIn).setVisibility(View.GONE
                );
            findViewById(R.id.sign_out_button).setVisibility(View.
                VISIBLE);
            findViewById(R.id.to_map_button).setVisibility(View.
                VISIBLE);

        } else {
            mStatusTextView.setText(R.string.signed_out);


            mState = "HIDE_MENU";
            userState = "SHOWUSER_SETTINGS";

            // refresh the action bar menu.
```

```
            invalidateOptionsMenu();

            // reset buttons.
            findViewById(R.id.email_password_buttons).setVisibility
                (View.VISIBLE);
            findViewById(R.id.email_password_fields).setVisibility(
                View.VISIBLE);
            findViewById(R.id.googleSignIn).setVisibility(View.
                VISIBLE);
            findViewById(R.id.sign_out_button).setVisibility(View.
               GONE);
            findViewById(R.id.to_map_button).setVisibility(View.
               GONE);
        }
    }

    /**
     * used for the checking of which buttons have been clicked
     * by the user.
     * @param v
     * The view used to see which has been selected.
     */
    @Override
    public void onClick(View v) {
        int i = v.getId();
        if (i == R.id.email_create_account_button) {
            Intent intent = new Intent(EmailPasswordAuthentication.
               this, RegisterAccount.class);
            startActivity(intent);
        } else if (i == R.id.email_sign_in_button) {
            signIn(mEmailField.getText().toString(), mPasswordField
               .getText().toString());
        } else if (i == R.id.sign_out_button) {
            signOut();
        }else if(i == R.id.googleSignIn) {
            signIn();
        } else if (i == R.id.to_map_button) {
            myMap();
        }
    }
}
```

Listing 12.1: EmailPasswordAuthentication

12.2.2 Progress

```
package adamoconnor.informe.LoginAndRegister;

import android.app.ProgressDialog;
import android.support.annotation.VisibleForTesting;
import android.support.v7.app.AppCompatActivity;

import adamoconnor.informe.R;

public class Progress extends AppCompatActivity {

    // declare progress dialog.
    @VisibleForTesting
    public static ProgressDialog mProgressDialog;

    /**
     * used for showing the progress dialog.
     */
    public void showProgressDialog() {
        if (mProgressDialog == null) {
            mProgressDialog = new ProgressDialog(this);
            mProgressDialog.setMessage(getString(R.string.loading))
                ;
            mProgressDialog.setIndeterminate(true);
        }

        mProgressDialog.show();
    }

    /**
     * used for dismissing the progress dialog.
     */
    public void hideProgressDialog() {
        if (mProgressDialog != null && mProgressDialog.isShowing())
            {
            mProgressDialog.dismiss();
        }
    }

    @Override
    public void onStop() {
        super.onStop();
        hideProgressDialog();
    }
```

```
}
```

Listing 12.2: Progress

12.2.3 RegisterAccount

```
package adamoconnor.informe.LoginAndRegister;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;
import adamoconnor.informe.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.squareup.picasso.Picasso;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

import java.util.Random;

public class RegisterAccount extends Progress {
```

```
private static final String TAG = "RegisterAccount";

//declare boolean
private boolean valid;

// declare the textviews etc...
private EditText mNameField;
private EditText mEmailField;
private EditText mPasswordField;
private ImageButton mSetupImageButton;

// declare button to register.
private Button mRegisterButton;

// declare progress bar.
private ProgressDialog mProgress;

// declaring firebase authentication, database and storage
   reference.
private FirebaseAuth mAuth;
private DatabaseReference mDatabase;
private StorageReference mStorageImage;

//declare an image URI.
private Uri mImageUri = null;

// declaring GALLERY_REQUEST result.
private static final int GALLERY_REQUEST = 1;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.register_account);

    //setting screen orientation to stop fragments view showing
        on eachother.
    this.setRequestedOrientation(ActivityInfo.
       SCREEN_ORIENTATION_PORTRAIT);

    isStoragePermissionGranted();

    // getting users authentication of application.
    mAuth = FirebaseAuth.getInstance();

    // database and storage reference's.
```

```
mDatabase = FirebaseDatabase.getInstance().getReference().
    child("users");
mStorageImage = FirebaseStorage.getInstance().getReference
    ().child("Profile_images");

// textfield and button declarations to the layout file.
mNameField = (EditText) findViewById(R.id.nameField);
mEmailField = (EditText) findViewById(R.id.emailField);
mPasswordField = (EditText) findViewById(R.id.passwordField
    );
mRegisterButton = (Button) findViewById(R.id.registerButton
    );
mSetupImageButton = (ImageButton) findViewById(R.id.
    profileSetup);

// used for selecting an image for the user's profile.
mSetupImageButton.setOnClickListener(new View.
    OnClickListener() {
     @Override
     public void onClick(View v) {
         new Thread() {
             @Override
             public void run() {

                 Intent galleryIntent = new Intent(Intent.
                     ACTION_GET_CONTENT);
                 galleryIntent.setType("image/*");
                 startActivityForResult(galleryIntent,
                     GALLERY_REQUEST);

             }
         }.start();

     }
});

// used for registering user.
mRegisterButton.setOnClickListener(new View.OnClickListener
    () {
     @Override
     public void onClick(View v) {
         validateForm();
         if(valid){
             startRegister();
         }else {
```

```
                    Toast.makeText(RegisterAccount.this, "
                        conditions are not correct",
                            Toast.LENGTH_SHORT).show();
                }
            }
        });

    }

    /**
     * used to get a random string needed to apply to the
     * profile image of user to combat clashes with image names.
     * @return
     * return a random string.
     */
    public static String random() {
        Random generator = new Random();
        StringBuilder randomStringBuilder = new StringBuilder();
        int randomLength = generator.nextInt(10);
        char tempChar;
        for (int i = 0; i < randomLength; i++){
            tempChar = (char) (generator.nextInt(96) + 32);
            randomStringBuilder.append(tempChar);
        }
        return randomStringBuilder.toString();
    }

    /**
     * start to register the user with an email address, password,
        name
     * and profile image.
     */
    private void startRegister() {

        // used for getting the information from the user.
        final String name = mNameField.getText().toString().trim();

        //set progress bar.
        mProgress = new ProgressDialog(RegisterAccount.this);
        mProgress.setMessage("Registering user ...");
        mProgress.show();

        // used for getting the information from the user.
        final String email = mEmailField.getText().toString().trim
            ();
```

```
final String password = mPasswordField.getText().toString()
    .trim();

//check if image is not selected.
if(mImageUri != null) {

        // START create_user_with_email and password.
        mAuth.createUserWithEmailAndPassword(email,
            password)
                .addOnCompleteListener(new
                    OnCompleteListener<AuthResult>() {
                     @Override
                     public void onComplete(@NonNull Task<
                         AuthResult> task) {
                         Log.d(TAG, "createUserWithEmail:
                             onComplete:" + task.isSuccessful
                             ());
                         if (task.isSuccessful()) {

                             // getting current users
                                 authenticated id.
                             final String user_id = mAuth.
                                 getCurrentUser().getUid();

                             StorageReference filepath =
                                 mStorageImage.child(random()
                                 );
                             filepath.putFile(mImageUri).
                                 addOnSuccessListener(new
                                 OnSuccessListener<UploadTask
                                 .TaskSnapshot>() {
                                  @Override
                                  // adding users information
                                      to the database.
                                  public void onSuccess(
                                      UploadTask.TaskSnapshot
                                      taskSnapshot) {

                                      @SuppressWarnings("
                                          VisibleForTests")
                                      String downloadUri =
                                          taskSnapshot.
                                          getDownloadUrl().
                                          toString();
```

```
                DatabaseReference
                    currentUserDb =
                    mDatabase.child(
                    user_id);
                currentUserDb.child("
                    name").setValue(name
                    );
                currentUserDb.child("
                    image").setValue(
                    downloadUri);

            }
        });

        // allow a delay for the
            database being updated
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    Thread.sleep(4000);

                    mProgress.dismiss()
                        ;
                    Intent mainIntent =
                         new Intent(
                        RegisterAccount.
                        this,
                        EmailPasswordAuthentication
                        .class);
                    mainIntent.setFlags
                        (Intent.
                        FLAG_ACTIVITY_CLEAR_TOP
                        );
                    startActivity(
                        mainIntent);

                } catch (
                    InterruptedException
                     e) {
                    e.printStackTrace()
                        ;
                }
            }
        }).start();
```

```
                                }
                                // If sign in fails, display a
                                   message to the user. If sign in
                                   succeeds
                                // the auth state listener will be
                                   notified and logic to handle the
                                // signed in user can be handled in
                                    the listener.
                                if (!task.isSuccessful()) {
                                    Toast.makeText(RegisterAccount.
                                       this, R.string.auth_failed +
                                        "Please check internet
                                       connection",
                                            Toast.LENGTH_SHORT).
                                               show();
                                }

                            }
                        });

        } else {
            Toast.makeText(RegisterAccount.this, "Please select an
               Image for your profile.",
                    Toast.LENGTH_SHORT).show();
        }

    }

    /**
     * used to validate the registration form.
     * @return
     * return the error if user has not entered text.
     */
    private boolean validateForm() {

        valid = true;

        String email = mEmailField.getText().toString();
        if (TextUtils.isEmpty(email)) {
            mEmailField.setError("Required.");
            valid = false;
        } else {
            mEmailField.setError(null);
        }
```

```
        String name = mNameField.getText().toString();
        if (TextUtils.isEmpty(name)) {
            mNameField.setError("Required.");
            valid = false;
        } else {
            mNameField.setError(null);
        }

        String password = mPasswordField.getText().toString();
        int size = password.length();
        if (TextUtils.isEmpty(password)) {
            mPasswordField.setError("Required.");
            valid = false;
        } else
            if(size < 6) {
                Toast.makeText(RegisterAccount.this, "password is
                    too short the minimum length is 6 characters",
                        Toast.LENGTH_LONG).show();
                valid = false;
            }
            else {
                mPasswordField.setError(null);
            }

        return valid;
    }

    /**
     *
     * @param requestCode
     * code on which was requested as the gallery intent.
     * @param resultCode
     * if the request was successful continue.
     * @param data
     * the data on which was selected, profile image.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode
        , Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if(requestCode == GALLERY_REQUEST && resultCode ==
            RESULT_OK) {
```

```
            // get the image data send to the declared URI
            mImageUri = data.getData();

            // crop the image to specific size.
            CropImage.activity(mImageUri)
                    .setGuidelines(CropImageView.Guidelines.ON)
                    .setAspectRatio(1,1)
                    .start(this);
        }

        // on cropped image display to the user in the image button
            .
        if (requestCode == CropImage.
           CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
            CropImage.ActivityResult result = CropImage.
               getActivityResult(data);
            if (resultCode == RESULT_OK) {
                Uri resultUri = result.getUri();

                Picasso.with(this)
                        .load(resultUri)
                        .centerCrop()
                        .resize(300,300)
                        .into(mSetupImageButton);

                mImageUri = resultUri;

            } else if (resultCode == CropImage.
               CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
                Exception error = result.getError();
            }
        }

    }

    //storage reference.
    final int MY_STORAGE = 1;
    /**
     * getting android permission to access storage for the device.
     * @return
     * return true or false
     */
    public  boolean isStoragePermissionGranted() {
        if (Build.VERSION.SDK_INT >= 23) {
            if (ContextCompat.checkSelfPermission(this,
```

```
                    Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    != PackageManager.PERMISSION_GRANTED) {

                // Should we show an explanation?
                if (ActivityCompat.
                    shouldShowRequestPermissionRationale(this,
                        Manifest.permission.WRITE_EXTERNAL_STORAGE)
                            ) {

                    // Show an expanation to the user *
                       asynchronously* -- don't block
                    // this thread waiting for the user's response!
                        After the user
                    // sees the explanation, try again to request
                       the permission.

                    //Prompt the user once explanation has been
                       shown
                    //(just doing it here for now, note that with
                       this code, no explanation is shown)
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.
                               WRITE_EXTERNAL_STORAGE},
                            MY_STORAGE);


                } else {
                    // No explanation needed, we can request the
                       permission.
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.
                               WRITE_EXTERNAL_STORAGE},
                            MY_STORAGE);
                }
            }
        }
        else {
            //permission is automatically granted on sdk<23 upon
               installation
            return true;
        }
        return false;
    }

    /**
```

```
     * request the permission that is needed.
     * @param requestCode
     * request the code needed
     * @param permissions
     * pass the permission that is needed
     * @param grantResults
     * retrieve the result the needs to be achieved
     */
    @Override
    public void onRequestPermissionsResult(int requestCode, String
       permissions[], int[] grantResults) {
        switch (requestCode) {
            case MY_STORAGE: {
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.
                           PERMISSION_GRANTED) {
                    // permission was granted, yay! Do the
                } else {
                    isStoragePermissionGranted();
                }
                return;
            }
        }
    }

}
```

Listing 12.3: RegisterAccount

12.2.4 SetupActivity

```
package adamoconnor.informe.LoginAndRegister;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import adamoconnor.informe.R;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.squareup.picasso.Picasso;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

import java.util.Random;

public class SetupActivity extends AppCompatActivity {

    // declared textfield and image button.
    private ImageButton mSetupImageButton;
    private EditText mNameField;

    // declare the setup button.
    private Button mSetupButton;

    //declare the brogress bar.
    private ProgressDialog mProgress;

    // declare firebasebase database and storage reference.
    private DatabaseReference mDatabaseUsers;
    private StorageReference mStorageImage;

    //declare image URI
    private Uri mImageUri = null;

    //declare the gallery request value.
    private static final int GALLERY_REQUEST = 1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_setup);
```

```
//setting screen orientation to stop fragments view showing
    on eachother.
this.setRequestedOrientation(ActivityInfo.
   SCREEN_ORIENTATION_PORTRAIT);

//initiate progress bar.
mProgress = new ProgressDialog(this);

// get instances of database and storage.
mStorageImage = FirebaseStorage.getInstance().getReference
   ().child("Profile_images");
mDatabaseUsers = FirebaseDatabase.getInstance().
   getReference().child("users");

// getting reference to the buttons and textfields.
mSetupImageButton = (ImageButton) findViewById(R.id.
   profileSetup);
mNameField = (EditText) findViewById(R.id.nameSetup);
mSetupButton = (Button) findViewById(R.id.submitSetup);

// profile image which needs to be selected
mSetupImageButton.setOnClickListener(new View.
   OnClickListener() {
    @Override
    public void onClick(View v) {
        new Thread() {
            @Override
            public void run() {

                Intent galleryIntent = new Intent(Intent.
                   ACTION_GET_CONTENT);
                galleryIntent.setType("image/*");
                startActivityForResult(galleryIntent,
                   GALLERY_REQUEST);

            }
        }.start();
    }
});

//setup the profile of the user.
mSetupButton.setOnClickListener(new View.OnClickListener()
   {
    @Override
    public void onClick(View v) {
```

```
                startSetupAccount();
            }
        });

        isStoragePermissionGranted();
    }

    //storage reference.
    final int MY_STORAGE = 1;
    /**
     * getting android permission to access storage for the device.
     * @return
     * return true or false
     */
    public  boolean isStoragePermissionGranted() {
        if (Build.VERSION.SDK_INT >= 23) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    != PackageManager.PERMISSION_GRANTED) {

                // Should we show an explanation?
                if (ActivityCompat.
                    shouldShowRequestPermissionRationale(this,
                          Manifest.permission.WRITE_EXTERNAL_STORAGE)
                             ) {

                    // Show an expanation to the user *
                       asynchronously* -- don't block
                    // this thread waiting for the user's response!
                        After the user
                    // sees the explanation, try again to request
                       the permission.

                    //Prompt the user once explanation has been
                       shown
                    //(just doing it here for now, note that with
                       this code, no explanation is shown)
                    ActivityCompat.requestPermissions(this,
                            new String[]{Manifest.permission.
                               WRITE_EXTERNAL_STORAGE},
                            MY_STORAGE);


                } else {
```

```
                // No explanation needed, we can request the
                    permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.
                            WRITE_EXTERNAL_STORAGE},
                        MY_STORAGE);
            }
        }
    }
    else {
        //permission is automatically granted on sdk<23 upon
            installation
        return true;
    }
    return false;
}

/**
 * request the permission that is needed.
 * @param requestCode
 * request the code needed
 * @param permissions
 * pass the permission that is needed
 * @param grantResults
 * retrieve the result the needs to be achieved
 */
@Override
public void onRequestPermissionsResult(int requestCode, String
    permissions[], int[] grantResults) {
    switch (requestCode) {
        case MY_STORAGE: {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.
                        PERMISSION_GRANTED) {
                // permission was granted, yay! Do the
            } else {
                isStoragePermissionGranted();
            }
            return;
        }
    }
}

/**
 * used to get a random string needed to apply to the
```

```
     * profile image of user to combat clashes with image names.
     * @return
     * return a random string.
     */
    public static String random() {
        Random generator = new Random();
        StringBuilder randomStringBuilder = new StringBuilder();
        int randomLength = generator.nextInt(10);
        char tempChar;
        for (int i = 0; i < randomLength; i++){
            tempChar = (char) (generator.nextInt(96) + 32);
            randomStringBuilder.append(tempChar);
        }
        return randomStringBuilder.toString();
    }

    /**
     * setting up of an account when user signs in by google.
     */
    private void startSetupAccount() {

        // getting name of user.
        final String name = mNameField.getText().toString().trim();
        // get the authenticated users id.
        final String user_id = FirebaseAuth.getInstance().
           getCurrentUser().getUid();

        if(!TextUtils.isEmpty(name) && mImageUri != null) {

            mProgress.setMessage("Setting up Account ...");
            mProgress.show();

            StorageReference filepath = mStorageImage.child(random
               ());
            filepath.putFile(mImageUri).addOnSuccessListener(new
               OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot
                   taskSnapshot) {

                    @SuppressWarnings("VisibleForTests")
                    String downloadUri = taskSnapshot.
                       getDownloadUrl().toString();
```

```
                    mDatabaseUsers.child(user_id).child("name").
                        setValue(name);
                    mDatabaseUsers.child(user_id).child("image").
                        setValue(downloadUri);

                    mProgress.dismiss();

                    Intent sendToMainIntent = new Intent(
                        SetupActivity.this,
                        EmailPasswordAuthentication.class);
                    sendToMainIntent.addFlags(Intent.
                        FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(sendToMainIntent);
                }
            });

        }

    }

    /**
     *
     * @param requestCode
     * code on which was requested as the gallery intent.
     * @param resultCode
     * if the request was successful continue.
     * @param data
     * the data on which was selected, profile image.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode
        , Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if(requestCode == GALLERY_REQUEST && resultCode ==
            RESULT_OK) {

            mImageUri = data.getData();

            CropImage.activity(mImageUri)
                    .setGuidelines(CropImageView.Guidelines.ON)
                    .setAspectRatio(1,1)
                    .start(this);

        }
```

```
        if (requestCode == CropImage.
           CROP_IMAGE_ACTIVITY_REQUEST_CODE) {
            CropImage.ActivityResult result = CropImage.
               getActivityResult(data);
            if (resultCode == RESULT_OK) {
                Uri resultUri = result.getUri();

                Picasso.with(this)
                        .load(resultUri)
                        .centerCrop()
                        .resize(300,300)
                        .into(mSetupImageButton);

                mImageUri = resultUri;

            } else if (resultCode == CropImage.
               CROP_IMAGE_ACTIVITY_RESULT_ERROR_CODE) {
                Exception error = result.getError();
            }
        }

    }

}
```

Listing 12.4: SetupActivity

12.2.5 ResetActivity

```
package adamoconnor.informe.LoginAndRegister;

import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v4.app.NavUtils;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import adamoconnor.informe.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;

public class ResetActivity extends AppCompatActivity {

    //declared inputs.
    private EditText emailField;

    // declare reset button.
    private Button resetButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reset);

        //setting screen orientation to stop fragments view showing
            on eachother.
        this.setRequestedOrientation(ActivityInfo.
           SCREEN_ORIENTATION_PORTRAIT);

        // action bar with back button.
        setupActionBar();

        //find view id for the design fields.
        emailField = (EditText) findViewById(R.id.mEmailField);
        resetButton = (Button) findViewById(R.id.resetButton);

        // when the reset button has been selected send to the
           forgotten password method.
        resetButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                // validate the form before processing.
                if(validateForm()) {

                    forgottenPassword();

                }
            }
        });
```

```
}

/**
 * setting up the navigation back button.
 * @param item
 * the image on whihc users select the back button.
 * @return
 * return the item to the options menu.
 */
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        NavUtils.navigateUpFromSameTask(this);
        return true;
    }
    return super.onOptionsItemSelected(item);
}

/**
 * setting up the top action bar.
 */
private void setupActionBar() {
    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        // Show the Up button in the action bar.
        actionBar.setDisplayHomeAsUpEnabled(true);

    }
}

/**
 * validate the form
 * @return
 * return the error if data wasn't entered.
 */
private boolean validateForm() {
    boolean valid = true;

    String email = emailField.getText().toString();
    if (TextUtils.isEmpty(email)) {
        emailField.setError("Required.");
        valid = false;
    } else {
        emailField.setError(null);
```

```
        }

        return valid;
    }

    /**
     * method used to send email to user with a
     * forgotten password.
     */
    public void forgottenPassword() {

        FirebaseAuth auth = FirebaseAuth.getInstance();
        final String emailAddress = emailField.getText().toString()
            .trim();

        if(android.util.Patterns.EMAIL_ADDRESS.matcher(emailAddress
            ).matches()) {
             auth.sendPasswordResetEmail(emailAddress)
                     .addOnCompleteListener(new OnCompleteListener<
                        Void>() {
                      @Override
                      public void onComplete(@NonNull Task<Void>
                         task) {
                          if (task.isSuccessful()) {
                              new AlertDialog.Builder(
                                  ResetActivity.this)
                                      .setTitle("Password Reset")
                                      .setMessage("An Email has
                                         been sent to the
                                         following email address
                                         "+emailAddress)
                                      .setPositiveButton(android.
                                         R.string.yes, new
                                         DialogInterface.
                                         OnClickListener() {
                                          public void onClick(
                                              DialogInterface
                                              dialog, int which) {


                                          }
                                      })
```

```
                .setNegativeButton(android.
                    R.string.no, new
                    DialogInterface.
                    OnClickListener() {
                     public void onClick(
                         DialogInterface
                         dialog, int which) {
                          // do nothing
                     }
                })
                .setIcon(R.drawable.
                    informe4)
                .show();
        }
        else {
            new AlertDialog.Builder(
                ResetActivity.this)
                .setTitle("Email Account")
                .setMessage("Sorry there is
                     no account using this
                    email address, would you
                     like to create an
                    account.")
                .setPositiveButton(android.
                    R.string.yes, new
                    DialogInterface.
                    OnClickListener() {
                     public void onClick(
                         DialogInterface
                         dialog, int which) {

                          Intent
                              createAccount =
                              new Intent(
                              ResetActivity.
                              this,
                              EmailPasswordAuthentication
                              .class);
                          createAccount.
                              setFlags(Intent.
                              FLAG_ACTIVITY_CLEAR_TOP
                              );
                          startActivity(
                              createAccount);
```

```
                                        }
                                    })
                                    .setNegativeButton(android.
                                       R.string.no, new
                                       DialogInterface.
                                       OnClickListener() {
                                        public void onClick(
                                           DialogInterface
                                           dialog, int which) {
                                            // do nothing
                                        }
                                    })
                                    .setIcon(R.drawable.
                                       informe4)
                                    .show();
                            }
                        }
                    });
        }
        else {
            new AlertDialog.Builder(ResetActivity.this)
                    .setTitle("Email Address")
                    .setMessage("Please enter a valid email address
                       ")
                    .setPositiveButton(android.R.string.yes, new
                       DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                            int which) {

                        }
                    })
                    .setNegativeButton(android.R.string.no, new
                       DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                            int which) {
                            // do nothing
                        }
                    })
                    .setIcon(R.drawable.informe4)
                    .show();
        }
    }
```

```
}
```

Listing 12.5: ResetActivity

12.3 Maps&Geofencing

12.3.1 AddInformation

```
package adamoconnor.informe.MapsAndGeofencing;

import android.content.ClipData;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v4.app.NavUtils;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;
import java.io.File;
import java.util.ArrayList;
import java.util.List;

import adamoconnor.informe.R;

public class AddInformation extends AppCompatActivity {

    private static final String TAG = "Add Information";

    //declare edit texts
    private EditText monumentName;
    private EditText areaName;
    private EditText monumentInformation;

    //declare spinner.
    private Spinner location;
```

```
// declare multiple image picker.
private int PICK_IMAGE_MULTIPLE = 1;
// declare image string.
private String imageEncoded;
//declare image list for multiple images
private static List<String> imagesEncodedList;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_newmonument);

    //setting screen orientation to stop fragments view showing
        on eachother.
    this.setRequestedOrientation(ActivityInfo.
       SCREEN_ORIENTATION_PORTRAIT);

    //setup back button action bar.
    setupActionBar();

    // declare gallery and send email buttons.
    Button gallery = (Button)findViewById(R.id.resetButton);
    Button sendEmail = (Button)findViewById(R.id.submitButton);

    //declare the textviews of the layout.
    monumentName = (EditText)findViewById(R.id.monumentName);
    areaName = (EditText)findViewById(R.id.area);
    monumentInformation = (EditText)findViewById(R.id.
       monumentInformation);
    location = (Spinner)findViewById(R.id.locationSpinner);


    // set the gallery OnClick listener.
    gallery.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent intent = new Intent();
            // Show only images, no videos or anything else
            intent.setType("image/*");
            intent.putExtra(Intent.EXTRA_ALLOW_MULTIPLE, true);
            intent.setAction(Intent.ACTION_GET_CONTENT);
            // Always show the chooser (if there are multiple
               options available)
```

```
                startActivityForResult(Intent.createChooser(intent,
                    "Select Picture"), PICK_IMAGE_MULTIPLE);
            }
        });

        // set OnClick listener for sending of email.
        sendEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                if(validateForm()) {
                    email("informedublinproject@gmail.com","",
                            "Add New Monument - adamoconnor.informe
                               ","Monument Name : "+monumentName.
                               getText()+"\n"
                                    +"Location : "+location.
                                       getSelectedItem().toString()
                                    +"\n"+"Area name : "+areaName.
                                       getText()+"\n"+"Monument
                                       Information : "+
                                       monumentInformation.getText
                                       ());
                }
            }
        });
    }

    /**
     * validating the text-fields on which the user
     * must enter their personal information.
     * @return
     * return the error on screen for the text-field.
     */
    private boolean validateForm() {
        boolean valid = true;

        String monument = monumentName.getText().toString();
        if (TextUtils.isEmpty(monument)) {
            monumentName.setError("Required.");
            valid = false;
        } else {
            monumentName.setError(null);
        }

        String area = areaName.getText().toString();
```

```
        if (TextUtils.isEmpty(area)) {
            areaName.setError("Required.");
            valid = false;
        } else {
            areaName.setError(null);
        }

        String description = monumentInformation.getText().toString
            ();
        if (TextUtils.isEmpty(description)) {
            areaName.setError("Required.");
            valid = false;
        } else {
            areaName.setError(null);
        }

        return valid;
    }

    /**
     * set back button for action bar
     * @param item
     * find items on options menu
     * @return
     * return the item selected.
     */
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            NavUtils.navigateUpFromSameTask(this);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * used for choosing images for sending by email.
     * @param requestCode
     * picking multiple or single images.
     * @param resultCode
     * see if everything went ok.
     * @param data
     * get the data on which was chosen.
     */
    @Override
```

```
protected void onActivityResult(int requestCode, int resultCode
    , Intent data) {
     try {
         // When an Image is picked
         if (requestCode == PICK_IMAGE_MULTIPLE && resultCode ==
             RESULT_OK
                 && null != data) {
             // Get the Image from data

             String[] filePathColumn = { MediaStore.Images.Media
                 .DATA };
             imagesEncodedList = new ArrayList<>();
             if(data.getData()!=null){

                 Uri mImageUri=data.getData();

                 // Get the cursor
                 Cursor cursor = getContentResolver().query(
                     mImageUri,
                             filePathColumn, null, null, null);
                 // Move to first row
                 cursor.moveToFirst();

                 int columnIndex = cursor.getColumnIndex(
                     filePathColumn[0]);
                 imageEncoded  = cursor.getString(columnIndex);
                 cursor.close();

             }else {
                 if (data.getClipData() != null) {
                     ClipData mClipData = data.getClipData();
                     ArrayList<Uri> mArrayUri = new ArrayList
                         <>();
                     for (int i = 0; i < mClipData.getItemCount
                         (); i++) {

                         ClipData.Item item = mClipData.
                             getItemAt(i);
                         Uri uri = item.getUri();
                         mArrayUri.add(uri);
                         // Get the cursor
                         Cursor cursor = getContentResolver().
                             query(uri, filePathColumn, null,
                             null, null);
                         // Move to first row
```

```
                        cursor.moveToFirst();

                        int columnIndex = cursor.getColumnIndex
                            (filePathColumn[0]);
                        imageEncoded  = cursor.getString(
                            columnIndex);
                        imagesEncodedList.add(imageEncoded);
                        cursor.close();
                    }

                }
            }
        } else {
            Toast.makeText(this, "You haven't picked an Image",
                    Toast.LENGTH_LONG).show();
        }
    } catch (Exception e) {
        Toast.makeText(this, "Something went wrong", Toast.
            LENGTH_LONG)
                .show();
    }

    super.onActivityResult(requestCode, resultCode, data);
}

/**
 * setup the above action bar.
 */
private void setupActionBar() {
    ActionBar actionBar = getSupportActionBar();
    if (actionBar != null) {
        // Show the Up button in the action bar.
        actionBar.setDisplayHomeAsUpEnabled(true);

    }
}

/**
 * sending the following email to inforMe@Dublin.
 * @param emailTo
 * send email to.
 * @param emailCC
 * used for carbon copy of email.
 * @param subject
 * setting the subject of the email.
```

```
     * @param emailText
     * the text which will be in the email.
     */
    public void email(String emailTo, String emailCC,
                      String subject, String emailText)
    {
        //need to "send multiple" to get more than one attachment
        final Intent emailIntent = new Intent(Intent.
            ACTION_SEND_MULTIPLE);
        emailIntent.setType("text/xml");
        emailIntent.putExtra(Intent.EXTRA_EMAIL,
                new String[]{emailTo});
        emailIntent.putExtra(Intent.EXTRA_CC,
                new String[]{emailCC});
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
        emailIntent.putExtra(Intent.EXTRA_TEXT, emailText);
        //has to be an ArrayList
        ArrayList<Uri> uris = new ArrayList<>();
        //convert from paths to Android friendly Parcelable Uri's
        try {
            for (String file : imagesEncodedList)
            {
                File fileIn = new File(file);
                Uri u = Uri.fromFile(fileIn);
                uris.add(u);
            }
        } catch (NullPointerException ex) {
            ex.getStackTrace();
        }

        emailIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM
            , uris);
        this.startActivity(Intent.createChooser(emailIntent, "Send
            mail..."));
    }
}
```

Listing 12.6: AddInformation

12.3.2 Constants

```
package adamoconnor.informe.MapsAndGeofencing;


import com.google.android.gms.location.Geofence;
import com.google.android.gms.maps.model.LatLng;
```

```
import java.util.HashMap;

public class Constants {

    // used to allow geofence to never expire.
    public static final long GEOFENCE_EXPIRATION_IN_MILLISECONDS =
        Geofence.NEVER_EXPIRE;

    // show the radius of the geofence
    public static final float GEOFENCE_RADIUS_IN_METERS = 50;

    // hashmap to hold the geo-fences.
    public static final HashMap<String, LatLng> LANDMARKS = new
        HashMap<>();

}
```

Listing 12.7: Constants

12.3.3 GeofenceTransitionsIntentService

```
package adamoconnor.informe.MapsAndGeofencing;

import android.app.IntentService;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.SystemClock;
import android.preference.PreferenceManager;
import android.support.v4.app.NotificationCompat;
import android.text.TextUtils;
import android.util.Log;
import adamoconnor.informe.PostingInformationAndComments.
    InformationFlipActivity;
import adamoconnor.informe.R;
import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofenceStatusCodes;
import com.google.android.gms.location.GeofencingEvent;
import java.util.ArrayList;
```

```
import java.util.List;

public class GeofenceTransitionsIntentService extends IntentService
    {

    protected static final String TAG = "GeofenceTransitionsIS";

    // getting description of area.
    public static String description;

    public GeofenceTransitionsIntentService() {
        super(TAG);  // use TAG to name the IntentService worker
            thread
    }

    @Override
    protected void onHandleIntent(Intent intent) {

        // geofence event triggered when entering geofence.
        GeofencingEvent event = GeofencingEvent.fromIntent(intent);
        if (event.hasError()) {
            Log.e(TAG, "GeofencingEvent Error: " + event.
                getErrorCode());
        }

        // get the description of the geofence.
        description = getGeofenceTransitionDetails(event);

        // allow a delay for the notification.
        SystemClock.sleep(3000);

        // call showNotifiation method to send notification when
            entering geofence.
        showNotification(description);
    }

    private static String getGeofenceTransitionDetails(
        GeofencingEvent event) {

        // getting the code of the geofence.
        String transitionString = GeofenceStatusCodes.
            getStatusCodeString(event.getGeofenceTransition());

        // getting trigger id.
        List triggeringIDs = new ArrayList();
```

```
        for (Geofence geofence : event.getTriggeringGeofences()) {
            triggeringIDs.add(geofence.getRequestId());
        }

        // return the strings of the geofence.
        return String.format("%s| %s", transitionString, TextUtils.
            join(", ", triggeringIDs));
    }

    public void showNotification(String text) {

        // getting the preferences which the user has set.
        SharedPreferences preferences = PreferenceManager.
            getDefaultSharedPreferences(this);

        //declare string
        String sound = null;

        //declare URI for sound.
        Uri notificationSound = null;

        //declare int for the vibration.
        int vibrationSet;

        // set the sound of the preferences.
        try {
            notificationSound = Uri.parse(preferences.getString("
                notifications_new_message_ringtone",sound));
        }catch (NullPointerException ex) {

        }

        if(notificationSound == null) {
            notificationSound = RingtoneManager.getDefaultUri(
                RingtoneManager.TYPE_NOTIFICATION);

        }


        //set the vibration.
        if(preferences.getBoolean("
            notifications_new_message_vibrate",true) == true) {
            vibrationSet = Notification.DEFAULT_VIBRATE;
        }else {
            vibrationSet = 0;
```

```
        }

        // 1. Create a NotificationManager
        NotificationManager notificationManager =
                (NotificationManager) this.getSystemService(Context
                    .NOTIFICATION_SERVICE);
        String[] splited = text.split("\\|");
        // 2. Create a PendingIntent for MapsActivity
        Intent intent = new Intent(this, InformationFlipActivity.
            class);
        intent.putExtra("1", splited[1]);
        intent.addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.
            FLAG_ACTIVITY_CLEAR_TOP); //FLAG_UPDATE_CURRENT
        PendingIntent pendingNotificationIntent = PendingIntent.
            getActivity(this, 0, intent, PendingIntent.
            FLAG_UPDATE_CURRENT);



        // 3. Create and send a notification
        Notification notification = new NotificationCompat.Builder(
            this)
                .setLargeIcon(BitmapFactory.decodeResource(this.
                    getResources(), R.drawable.informe))
                .setSmallIcon(R.drawable.informe)
                .setContentTitle("Historic Site Entered")
                .setStyle(new NotificationCompat.BigTextStyle().
                    bigText("Hey you have entered a site of historic
                     significance - "+splited[1]))
                .setContentIntent(pendingNotificationIntent)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setSound(notificationSound)
                .setDefaults(vibrationSet)
                .setAutoCancel(true)
                .build();

        notificationManager.notify(0, notification);

        //create a thread.
        MyThread myThread = new MyThread();
        myThread.start();
    }

    final static String MY_ACTION = "MY_ACTION";
```

```
    /**
     * create thread to send the data to the MapsActivity to
        populate dialog etc.
     */
    private class MyThread extends Thread{

        @Override
        public void run() {

            try {
                Thread.sleep(3000);
                String[] splited = description.split("\\|");
                Intent intent = new Intent();
                intent.setAction(MY_ACTION);
                intent.putExtra("DATAPASSED", splited[1]);

                sendBroadcast(intent);

            }catch(InterruptedException e){

            }
            stopSelf();
        }



    }

}
```

Listing 12.8: GeofenceTransitionsIntentService

12.3.4 MapsActivity

```
package adamoconnor.informe.MapsAndGeofencing;

import android.Manifest;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.IntentSender;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
```

```
import android.content.pm.PackageManager;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Location;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.preference.PreferenceManager;
import android.support.design.widget.FloatingActionButton;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AlertDialog;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.CompoundButton;
import android.widget.Toast;
import android.widget.ToggleButton;
import adamoconnor.informe.LoginAndRegister.Progress;
import adamoconnor.informe.PostingInformationAndComments.
    InformationFlipActivity;
import adamoconnor.informe.R;
import adamoconnor.informe.Settings.CheckConnectivity;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.Geofence;
import com.google.android.gms.location.GeofencingRequest;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsRequest;
import com.google.android.gms.location.LocationSettingsResult;
import com.google.android.gms.location.LocationSettingsStatusCodes;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.LocationSource;
import com.google.android.gms.maps.OnMapReadyCallback;
```

```
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.UiSettings;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.CircleOptions;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.util.ArrayList;
import java.util.Map;

import static adamoconnor.informe.R.id.map;

public class MapsActivity extends Progress
        implements OnMapReadyCallback,
        GoogleApiClient.ConnectionCallbacks,
        GoogleApiClient.OnConnectionFailedListener,
        LocationSource,
        LocationListener,
        ResultCallback<Status>,
        SensorEventListener {

    Marker mGeofenceMarker;

    //declare TAG.
    private static final String TAG = MapsActivity.class.
       getSimpleName();
    //declare GoogleMap for fragment
    private GoogleMap mGoogleMap;
    //used for populating of geofences
    boolean populate = true;
    //used for the map fragment
    private SupportMapFragment mapFrag;
    //used to get last location of user.
    protected Location mLastLocation;
    // used to recieve location updates
    private myReceiver myReceiver;
    //geofence arraylist
    protected ArrayList<Geofence> mGeofenceList;
    //declare googleAPIClient
    protected GoogleApiClient mGoogleApiClient;
    //declare sensor
    private SensorManager mSensorManager;
    //declare type of sensor used
```

```
private Sensor mProximity;
//sensitivity of sensor
private static final int SENSOR_SENSITIVITY = 4;
//declare location manager
protected LocationManager locationManager;
//declare location changed listener
protected OnLocationChangedListener mListener;
// checking for gps connection.
protected static final int REQUEST_CHECK_SETTINGS = 0x1;
// declare the location request.
protected LocationRequest mLocationRequest;
//declare handler.
protected Handler h;
//declare runnable.
protected Runnable myrunnable;
//declare check connectivity class.
private CheckConnectivity checkConnectivity;

/**
 * Tracks the status of the location updates request. Value
    changes when the user presses the
 * Start Updates and Stop Updates buttons.
 */
protected Boolean mRequestingLocationUpdates;

/**
 * Stores the types of location services the client is
    interested in using. Used for checking
 * settings to determine if the device has optimal location
    settings.
 */
protected LocationSettingsRequest mLocationSettingsRequest;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_maps);

    // stop the activity resetting don't allow landscape mode.
    this.setRequestedOrientation(ActivityInfo.
       SCREEN_ORIENTATION_PORTRAIT);

    // check permission if build is marshmallow or over.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
```

```
        checkLocationPermission();
    }

    //check location on.
    checkConnectivity = new CheckConnectivity();
    checkConnectivity.startInternetEnabled(this);
    checkConnectivity.startLocationEnabled(this);

    // getting preferences of the user to see if battery saver
       is on.
    SharedPreferences preferences = PreferenceManager.
       getDefaultSharedPreferences(this);

    if(preferences.getBoolean("battery_switch",true) == true) {

        mSensorManager = (SensorManager) getSystemService(
           Context.SENSOR_SERVICE);
        mProximity = mSensorManager.getDefaultSensor(Sensor.
           TYPE_PROXIMITY);
    }

    // used to send email of new monument to adamoconnor.
       informe.
    FloatingActionButton addInfo = (FloatingActionButton)  this
       .findViewById(R.id.floatingAddInfoButton);
    addInfo.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(MapsActivity.this ,
               AddInformation.class);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
        }
    });

    // used to toggle focus on and off.
    final ToggleButton toggle = (ToggleButton) findViewById(R.
       id.toggleFocus);
    toggle.setOnCheckedChangeListener(new CompoundButton.
       OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
            if (!isChecked) {
                // The toggle is enabled
```

```
                    toggle.setBackgroundTintList(ColorStateList.
                        valueOf(getResources().getColor(R.color.
                        TransparetFocus)));
                    h = new Handler();
                    final int delay = 2000; //milliseconds

                    h.postDelayed(myrunnable = new Runnable(){
                        public void run(){
                            startLocationUpdates();
                            h.postDelayed(this, delay);
                        }
                    }, delay);

                } else {
                    // The toggle is disabled
                    toggle.setBackgroundTintList(ColorStateList.
                        valueOf(getResources().getColor(R.color.
                        TransparetNonFocus)));
                    try {
                        h.removeCallbacks(myrunnable);
                        h.removeCallbacks(myrunnable);
                        h.removeCallbacks(myrunnable);
                    }catch (NullPointerException ex) {

                    }
                    stopLocationUpdates();
                }
            }
        });

        // geofence array list.
        mGeofenceList = new ArrayList<>();

        mapFrag = (SupportMapFragment) getSupportFragmentManager().
            findFragmentById(map);
        mapFrag.getMapAsync(this);
        createLocationRequest();

    }

    /**
     * creating the map on the creation of the activity.
     * set each of the ui components of the map, such as
     * compass etc.
     * @param googleMap
```

```
     * pass the map on which is being shown on the activity.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {

        mGoogleMap = googleMap;
        // type of map needed to display.
        mGoogleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);

        UiSettings uiSettings = googleMap.getUiSettings();
        uiSettings.setAllGesturesEnabled(true);
        uiSettings.setCompassEnabled(true);
        uiSettings.setMyLocationButtonEnabled(false);
        uiSettings.setMapToolbarEnabled(true);
        uiSettings.setZoomControlsEnabled(false);
        googleMap.setTrafficEnabled(false);

        //Initialize Google Play Services
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)
                    == PackageManager.PERMISSION_GRANTED) {
                buildGoogleApiClient();
                mGoogleMap.setMyLocationEnabled(true);
            }
        } else {
            buildGoogleApiClient();
            mGoogleMap.setMyLocationEnabled(true);
        }
    }

    /**
     * call the api client needed to retrieve information.
     * from google such as maps and the location API.
     */
    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();

    }
```

```
/**
 * call to location updates on when the device changes
 * coordinates, its then updated on the map.
 * @param bundle
 * the information of the activity.
 */
@Override
public void onConnected(Bundle bundle) {

    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
        createLocationRequest();
        LocationServices.FusedLocationApi.
            requestLocationUpdates(mGoogleApiClient,
            mLocationRequest, this);
    }
}

/**
 * when requesting location, needs interval set and what
    accuracy
 * the developer wants to achieve.
 */
protected void createLocationRequest() {

    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(3000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.
        PRIORITY_HIGH_ACCURACY);

    LocationSettingsRequest.Builder builder = new
        LocationSettingsRequest.Builder()
            .addLocationRequest(mLocationRequest);

    mLocationSettingsRequest = builder.build();
}

/**
 * used to focus the map view when the user moves.
 * camera is then animated to find users coordinates.
 */
protected void startLocationUpdates() {
    LocationServices.SettingsApi.checkLocationSettings(
```

```
            mGoogleApiClient,
            mLocationSettingsRequest
    ).setResultCallback(new ResultCallback<
        LocationSettingsResult>() {
        @Override
        public void onResult(LocationSettingsResult
            locationSettingsResult) {
            final Status status = locationSettingsResult.
                getStatus();
            switch (status.getStatusCode()) {
                case LocationSettingsStatusCodes.SUCCESS:
                    Log.i(TAG, "All location settings are
                        satisfied.");
                    checkLocationPermission();
                    LocationServices.FusedLocationApi.
                        requestLocationUpdates(
                            mGoogleApiClient, mLocationRequest,
                                MapsActivity.this);
                    break;
                case LocationSettingsStatusCodes.
                    RESOLUTION_REQUIRED:
                    Log.i(TAG, "Location settings are not
                        satisfied. Attempting to upgrade " +
                            "location settings ");
                    try {
                        // Show the dialog by calling
                            startResolutionForResult(), and
                            check the
                        // result in onActivityResult().
                        status.startResolutionForResult(
                            MapsActivity.this,
                            REQUEST_CHECK_SETTINGS);
                    } catch (IntentSender.SendIntentException e
                        ) {
                        Log.i(TAG, "PendingIntent unable to
                            execute request.");
                    }
                    break;
                case LocationSettingsStatusCodes.
                    SETTINGS_CHANGE_UNAVAILABLE:
                    String errorMessage = "Location settings
                        are inadequate, and cannot be " +
                            "fixed here. Fix in Settings.";
                    Log.e(TAG, errorMessage);
```

```
                        Toast.makeText(MapsActivity.this,
                            errorMessage, Toast.LENGTH_LONG).show();
                        mRequestingLocationUpdates = false;
                    }
                }
            });

    }

    /**
     * It is a good practice to remove location requests when the
        activity is in a paused or
     * stopped state. Doing so helps battery performance and is
        especially
     * recommended in applications that request frequent location
        updates.
     */
    protected void stopLocationUpdates() {

        LocationServices.FusedLocationApi.removeLocationUpdates(
                mGoogleApiClient,
                this
        ).setResultCallback(new ResultCallback<Status>() {
            @Override
            public void onResult(Status status) {
                mRequestingLocationUpdates = false;
            }
        });
    }

    /** Activates this provider. This provider will notify the
       supplied listener
     * periodically, until you call deactivate().
     * This method is automatically invoked by enabling my-location
         layer.
     */
    @Override
    public void activate(OnLocationChangedListener listener) {
        // We need to keep a reference to my-location layer's
            listener so we can push forward
        // location updates to it when we receive them from
            Location Manager.
        mListener = listener;
        String bestAvailableProvider = LocationManager.GPS_PROVIDER
            ;
```

```
        long minTime = 1000;
        float minDistance = 10;

        // Request location updates from Location Manager
        if (bestAvailableProvider != null) {
            checkLocationPermission();
            locationManager.requestLocationUpdates(
                bestAvailableProvider, minTime, minDistance, (
                android.location.LocationListener) this);
        } else {
            // (Display a message/dialog) No Location Providers
                currently available.
        }
    }

    /** Deactivates this provider.
     *  This method is automatically invoked by disabling my-
        location layer.
     */
    @Override
    public void deactivate() {
        // Remove location updates from Location Manager
        locationManager.removeUpdates((android.location.
            LocationListener) this);

        mListener = null;
    }

    /**
     * when the location of the user is changed the method as
        follows is
     * initiated.
     * @param location
     */
    @Override
    public void onLocationChanged(Location location) {

        // get the last location which was found on device.
        mLastLocation = location;
        //declare LatLng for coordinates of camera animation.
        LatLng latLng;

        if (mGeofenceMarker != null) {
            mGeofenceMarker.remove();
        }
```

```
double latitude = 0;
double longitude = 0;
try {
    // find the location from the Gps on the map.
    Location findMe = mGoogleMap.getMyLocation();
    latitude = findMe.getLatitude();
    longitude = findMe.getLongitude();

}catch(Exception ex) {
    ex.getLocalizedMessage();
}finally {
    if(latitude != 0 && longitude != 0) {
        latLng = new LatLng(latitude, longitude);
    }
    else {
        // use the location lat and long if map location
           cannot be found
        latLng = new LatLng(location.getLatitude(),
           location.getLongitude());
    }
}

// animation of the camera.
CameraPosition cameraPosition = new CameraPosition.Builder
   ()
        .target(latLng)      // Sets the center of the map
           to Mountain View
        .zoom(20)                    // Sets the zoom
        .bearing(90)                 // Sets the orientation
            of the camera to east
        .tilt(45)                    // Sets the tilt of the
            camera to 30 degrees
        .build();                    // Creates a
           CameraPosition from the builder
mGoogleMap.animateCamera(CameraUpdateFactory.
   newCameraPosition(cameraPosition));

// clear anything not needed.
mGoogleMap.clear();

/*
* used for the loading of all geofences which where called
   from firebase when
* the user has signed in.
```

```
*/
for (Map.Entry<String, LatLng> entry : adamoconnor.informe.
    MapsAndGeofencing.Constants.LANDMARKS.entrySet()) {

    // creation of the geofence colour and border colour.
    // setting of each specific geofence.
    CircleOptions circleOptions = new CircleOptions()
            .center(new LatLng(entry.getValue().latitude,
                entry.getValue().longitude))
            .strokeColor(Color.argb(50, 70,70,70))
            .fillColor(getResources().getColor(R.color.
                Geofence))
            .radius(50);
    mGoogleMap.addCircle( circleOptions );

    //Place current location marker
    LatLng geo = new LatLng(entry.getValue().latitude,entry
        .getValue().longitude);
    MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(geo);
    markerOptions.title(entry.getKey());
    markerOptions.icon(BitmapDescriptorFactory.fromResource
        (R.drawable.informe));//BitmapDescriptorFactory.
        defaultMarker(BitmapDescriptorFactory.HUE_MAGENTA)

    //adding marker to each geofence.
    mGeofenceMarker = mGoogleMap.addMarker(markerOptions);

    // add each to the geofence array list with the lat and
        long as well as the radius in meters.
    mGeofenceList.add(new Geofence.Builder()
            .setRequestId(entry.getKey())
            .setCircularRegion(
                    entry.getValue().latitude,
                    entry.getValue().longitude,
                    adamoconnor.informe.MapsAndGeofencing.
                        Constants.GEOFENCE_RADIUS_IN_METERS
            )
            .setExpirationDuration(adamoconnor.informe.
                MapsAndGeofencing.Constants.
                GEOFENCE_EXPIRATION_IN_MILLISECONDS)
            .setTransitionTypes(Geofence.
                GEOFENCE_TRANSITION_ENTER |
                    Geofence.GEOFENCE_TRANSITION_EXIT)
            .build());
```

```
        }

        //stop location updates
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates
               (mGoogleApiClient, this);
        }

        // used to populate the check if user is in specific
           geofence.
        if(populate) {
            try{
                PopulateGeofences();
            }catch (IllegalStateException ex) {

            }

            populate = false;
        }

    }

    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 99;

    /**
     * used to check the location permission
     * @return
     * dialog if user has not set permissions
     */
    public boolean checkLocationPermission() {
        if (ContextCompat.checkSelfPermission(this,
                Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {

            // Should we show an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale
               (this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {

                // Show an expanation to the user *asynchronously*
                   -- don't block
                // this thread waiting for the user's response!
                   After the user
                // sees the explanation, try again to request the
                   permission.
```

```
                //Prompt the user once explanation has been shown
                //(just doing it here for now, note that with this
                    code, no explanation is shown)
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.
                            ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);


            } else {
                // No explanation needed, we can request the
                    permission.
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.
                            ACCESS_FINE_LOCATION},
                        MY_PERMISSIONS_REQUEST_LOCATION);
            }
            return false;
        } else {
            return true;
        }
    }

    /**
     *
     * @param requestCode
     * start to see if user connected
     * @param permissions
     * permissions which are needed for the application
     * @param grantResults
     * if the user has granted access to sensors.
     */
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           String permissions[],
                                             int[] grantResults) {
        switch (requestCode) {
            case MY_PERMISSIONS_REQUEST_LOCATION: {
                // If request is cancelled, the result arrays are
                    empty.
                if (grantResults.length > 0
                        && grantResults[0] == PackageManager.
                            PERMISSION_GRANTED) {
```

```
                    // permission was granted, yay! Do the
                    // location-related task you need to do.
                    if (ContextCompat.checkSelfPermission(this,
                            Manifest.permission.
                                ACCESS_FINE_LOCATION)
                            == PackageManager.PERMISSION_GRANTED) {

                        if (mGoogleApiClient == null) {
                            buildGoogleApiClient();
                        }
                        mGoogleMap.setMyLocationEnabled(true);
                    }

                } else {

                    // permission denied, boo! Disable the
                    // functionality that depends on this
                        permission.
                    checkLocationPermission();
                }
                return;
            }

            // other 'case' lines to check for other
            // permissions this app might request
        }
    }

    /**
     * used to see if user has entered a geofence.
     */
    public void PopulateGeofences() {
        if (!mGoogleApiClient.isConnected()) {
            Toast.makeText(this, "Google API Client not connected!"
                , Toast.LENGTH_SHORT).show();
            return;
        }

        try {
            LocationServices.GeofencingApi.addGeofences(
                    mGoogleApiClient,
                    getGeofencingRequest(),
                    getGeofencePendingIntent()
            ).setResultCallback(this); // Result processed in
                onResult().
```

```
        } catch (SecurityException securityException) {
            // Catch exception generated if the app does not use
               ACCESS_FINE_LOCATION permission.
        }
    }

    /**
     * get the geofence which is requested.
     * @return
     * to the pending intent.
     */
    private GeofencingRequest getGeofencingRequest() {
        GeofencingRequest.Builder builder = new GeofencingRequest.
            Builder();
        builder.setInitialTrigger(GeofencingRequest.
            INITIAL_TRIGGER_ENTER);
        builder.addGeofences(mGeofenceList);
        return builder.build();
    }

    /**
     * sending the notification to the user when geofence is
        entered.
     * @return
     */
    private PendingIntent getGeofencePendingIntent() {

        Log.d(TAG, "Geo fence pending intent");
        Intent intent = new Intent(this,
            GeofenceTransitionsIntentService.class);
        // We use FLAG_UPDATE_CURRENT so that we get the same
            pending intent back when calling addgeoFences()

        return PendingIntent.getService(this, 0, intent,
            PendingIntent.FLAG_UPDATE_CURRENT);
    }

    /**
     * used for when the activity has started.
     */
    @Override
    protected void onStart() {
        // used as a broadcast receiver
        myReceiver = new myReceiver();
        IntentFilter intentFilter = new IntentFilter();
```

```
        intentFilter.addAction(adamoconnor.informe.
            MapsAndGeofencing.GeofenceTransitionsIntentService.
            MY_ACTION);
        registerReceiver(myReceiver, intentFilter);

        // create a new thread to start the focusing method for the
            view on the map.
        new Thread(new Runnable() {
            @Override

            public void run() {
                Looper.prepare();
                try {
                    Thread.sleep(5000);
                    h = new Handler();
                    final int delay = 2000; //milliseconds

                    h.postDelayed(myrunnable = new Runnable(){
                        public void run(){
                            startLocationUpdates();
                            h.postDelayed(this, delay);
                        }
                    }, delay);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                Looper.loop();
            }
        }).start();

        super.onStart();
    }

    /**
     * used to register the proximity sensor of the device.
     */
    @Override
    public void onResume() {
        super.onResume();
        try {
            mSensorManager.registerListener(this, mProximity,
                SensorManager.SENSOR_DELAY_NORMAL);
        }catch (NullPointerException ex) {

        }
```

```
}

/**
 * used to stop the proximity sensor as well as the location
    provider.
 */
@Override
public void onPause() {
    super.onPause();

    h.removeCallbacks(myrunnable);
    h.removeCallbacks(myrunnable);

    //stop location updates when Activity is no longer active
    try {
        mSensorManager.unregisterListener(this);
    }catch(NullPointerException ex) {

    }

    if (mGoogleApiClient != null) {
        try {
            LocationServices.FusedLocationApi.
               removeLocationUpdates(mGoogleApiClient, this);
        }catch(IllegalStateException ex ){

        }

    }
}

/**
 * used to check wheather the sensor is blocked or not
 * which changes the brightness of the screen when on the
 * map.
 * @param event
 * change of the screen brightness.
 */
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_PROXIMITY) {
        if (event.values[0] >= -SENSOR_SENSITIVITY && event.
           values[0] <= SENSOR_SENSITIVITY) {
            //near
```

```
                WindowManager.LayoutParams layout = getWindow().
                    getAttributes();
                layout.screenBrightness = 0F;
                getWindow().setAttributes(layout);
            } else {
                //far
                WindowManager.LayoutParams layout = getWindow().
                    getAttributes();
                layout.screenBrightness = 0.7F;
                getWindow().setAttributes(layout);
            }
        }
    }

    /**
     * needed to call the sensor but not needed in this case.
     * @param sensor
     * what sensor are we using.
     * @param accuracy
     * the accuracy that is needed.
     */
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }

    /**
     * unregister the broadcast receiver.
     */
    @Override
    public void onStop() {
        h.removeCallbacks(myrunnable);
        h.removeCallbacks(myrunnable);
        unregisterReceiver(myReceiver);
        super.onStop();
    }

    /**
     * when the connection has failed.
     * @param result
     * result returned when an error has been found.
     */
    @Override
    public void onConnectionFailed(ConnectionResult result) {
       result.getErrorCode(); }
```

```
/**
 * when the connection of the application has become suspended
 * @param cause
 * the cause of the activity becoming suspended.
 */
@Override
public void onConnectionSuspended(int cause) {
    mGoogleApiClient.connect();
}

/**
 * when the geofences have been added a toast will appear.
 * @param status
 * see if the geofences have been added.
 */
public void onResult(Status status)
{
    if (status.isSuccess()) {
        Toast.makeText(this, "Geofences Added", Toast.
            LENGTH_SHORT).show();
    } else {
        Toast.makeText(this,"Sorry an Error has occurred",Toast
            .LENGTH_SHORT).show();
    }
}

/**
 * broadcast receiver used to produce the dialog on the map
    activity if user has entered
 * a geofence.
 */
private class myReceiver extends BroadcastReceiver {

    /**
     * when an notification has been initiated it will be sent
        to the following
     * dialog and will be sent to the users map to display.
     * @param arg0
     * the context of the intent
     * @param arg1
     * the intent
     */
    @Override
    public void onReceive(Context arg0, Intent arg1) {
```

```
            // getting the information from the
               GeofenceTransitionIntentService.
            final String datapassed = arg1.getStringExtra("
               DATAPASSED");

            // display dialog to the specific user.
            new AlertDialog.Builder(MapsActivity.this)
                    .setTitle("Historic Monument")
                    .setMessage("Do you want to view information on
                        the monument area you have entered ? - "+
                       datapassed)
                    .setPositiveButton(android.R.string.yes, new
                       DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                            int which) {

                            // send user to the specific
                               Information activity.
                            Intent monumentInformationIntent = new
                               Intent(getApplicationContext(),
                               InformationFlipActivity.class);
                            // pass the name of the monument to the
                                activity to retrieve information
                               from firebase.
                            monumentInformationIntent.putExtra("
                               monumentInformation", datapassed);
                            startActivity(monumentInformationIntent
                               );

                        }
                    })
                    .setNegativeButton(android.R.string.no, new
                       DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                            int which) {
                            // do nothing
                        }
                    })
                    .setIcon(R.drawable.informe4)
                    .show();

        }

    }
```

```
}
```

Listing 12.9: MapsActivity

12.3.5 Place

```
package adamoconnor.informe.MapsAndGeofencing;

public class Place {

    // monument name of the historical place
    private static String MonumentName;

    /**
     * get the name of the monument which was set.
     * @return
     * the monument name.
     */
    public static String getMonumentName() {
        return MonumentName;
    }

    /**
     * set the name of the monument
     * @param monumentName
     * setting of the monument name.
     */
    public static void setMonumentName(String monumentName) {
        MonumentName = monumentName;
    }
}
```

Listing 12.10: Place

12.4 PostingInformation&Comments

12.4.1 Comments

```
package adamoconnor.informe.PostingInformationAndComments;


public class Comments {
```

```
// declare strings of the firebase monument posts
private String title;
private String description;
private String image;
private String username;
private String profile;

// declare timestamp for monument post.
private Long timestamp;

// needed constructor.
public Comments() {}

/**
 * used as the overall constructor
 * @param title
 * setting title.
 * @param description
 * setting description.
 * @param image
 * setting image.
 * @param username
 * setting username.
 * @param timestamp
 * setting timestamp.
 * @param profile
 * setting profile image.
 */
public Comments(String title, String description, String image,
    String username, Long timestamp, String profile) {

    this.title = title;
    this.description = description;
    this.image = image;
    this.username = username;
    this.timestamp = timestamp;
    this.profile = profile;
}

/**
 * getting timestamp
 * @return
 * timestamp value
 */
```

```
public Long getTimestamp() {
    return timestamp;
}

/**
 * setting timestamp
 * @param timestamp
 * timestamp value
 */
public void setTimestamp(Long timestamp) {
    this.timestamp = timestamp;
}

/**
 * getting profile Image
 * @return
 * profile image value
 */
public String getProfile() {
    return profile;
}

/**
 * setting profile image
 * @param profile
 * profile image value
 */
public void setProfile(String profile) {
    this.profile = profile;
}

/**
 * getting username
 * @return
 * username value
 */
public String getUsername() {
    return username;
}

/**
 * setting timestamp
 * @param username
 * username value
 */
```

```
public void setUsername(String username) { this.username =
    username;}

/**
 * getting title
 * @return
 * title value
 */
public String getTitle() {
    return title;
}

/**
 * setting titke
 * @param title
 * title value
 */
public void setTitle(String title) {
    this.title = title;
}

/**
 * getting description
 * @return
 * description value
 */
public String getDescription() {
    return description;
}

/**
 * setting description
 * @param description
 * description value
 */
public void setDescription(String description) {
    this.description = description;
}

/**
 * getting image
 * @return
 * image value
 */
public String getImage() {
```

```
        return image;
    }

    /**
     * setting image
     * @param image
     * image value
     */
    public void setImage(String image) {
        this.image = image;
    }


}
```

Listing 12.11: Comments

12.4.2 CommentsActivity

```
package adamoconnor.informe.PostingInformationAndComments;

import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import adamoconnor.informe.R;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import de.hdodenhof.circleimageview.CircleImageView;

public class CommentsActivity extends AppCompatActivity {
```

```
// strings needed for the post id and monument name id.
private String mPost_key = null;
private String mLocation_key = null;

// used for the post unique id.
private String post_uid = null;

//database reference.
private DatabaseReference mDatabase;

// textviews for the posted information.
private TextView postTitle;
private TextView postDescription;
private TextView postTimestamp;
private TextView postUsername;

// declare image views for the posted image and posted user.
private CircleImageView postProfile;
private ImageView postImage;

// declare post remove and update buttons.
private Button postRemove;
private Button postUpdate;

//firebase authentication.
private FirebaseAuth mAuth;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_comments);

    //setting screen orientation to stop fragments view showing
        on eachother.
    this.setRequestedOrientation(ActivityInfo.
       SCREEN_ORIENTATION_PORTRAIT);

    // getting the reference of the specific post.
    mPost_key = getIntent().getExtras().getString("post_id");
    mLocation_key = getIntent().getExtras().getString("
       location_id");

    //get instance of the database.
    mDatabase = FirebaseDatabase.getInstance().getReference().
       child("comments").child(mLocation_key);
```

```
// get the authentication instance.
mAuth = FirebaseAuth.getInstance();

// getting the textviews reference form the layout.
postUsername = (TextView) findViewById(R.id.commentUsername
    );
postTimestamp = (TextView) findViewById(R.id.commentTime);
postTitle = (TextView) findViewById(R.id.commentTitle);
postDescription = (TextView) findViewById(R.id.
    commentDescription);

// getting the imageView reference.
postImage = (ImageView) findViewById(R.id.commentImage);
postProfile = (CircleImageView) findViewById(R.id.
    commentProfile);

// getting the button references
postRemove = (Button) findViewById(R.id.commentRemove);
postUpdate = (Button) findViewById(R.id.updatePost);

// getting the information from firebase and display on the
     activity.
mDatabase.child(mPost_key).addValueEventListener(new
    ValueEventListener() {
     @Override
     public void onDataChange(DataSnapshot dataSnapshot) {

         String post_title = (String)dataSnapshot.child("
             title").getValue();
         String post_desc = (String)dataSnapshot.child("
             description").getValue();
         String post_image = (String)dataSnapshot.child("
             image").getValue();

         String post_profile = (String)dataSnapshot.child("
             profile").getValue();
         String post_username = (String)dataSnapshot.child("
             username").getValue();
        Long post_timestamp = (Long)dataSnapshot.child("
             timestamp").getValue();
         post_uid = (String)dataSnapshot.child("uid").
             getValue();

         postTitle.setText(post_title);
         postDescription.setText(post_desc);
```

```
            postUsername.setText(post_username);

            //used to get the correct date and time format from
                timestamp.
            try {

                DateFormat sdf = new SimpleDateFormat().
                    getDateTimeInstance();
                Date netDate = (new Date(post_timestamp));
                postTimestamp.setText(sdf.format(netDate));

            } catch(NullPointerException ex) {
                ex.getStackTrace();
            }

            // load the images from the reference with picasso
            Picasso.with(CommentsActivity.this).load(post_image
                ).resize(1200,750).into(postImage);
            Picasso.with(CommentsActivity.this).load(
                post_profile)
                    .placeholder(R.drawable.defaulticon).error(
                        R.drawable.defaulticon)
                    .into(postProfile);

            //only when the same user is authenticated my they
                have the option to delete or update post.
            if(mAuth.getCurrentUser().getUid().equals(post_uid)
                ) {

                postRemove.setVisibility(View.VISIBLE);
                postUpdate.setVisibility(View.VISIBLE);
            }
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });

    //used for removing the post.
    postRemove.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
```

```
                if(mAuth.getCurrentUser().getUid().equals(post_uid)
                    ) {

                    mDatabase.child(mPost_key).removeValue();

                    Intent InformationIntent = new Intent(
                        CommentsActivity.this,
                        InformationFlipActivity.class);
                    InformationIntent.addFlags(Intent.
                        FLAG_ACTIVITY_CLEAR_TOP);
                    InformationIntent.addFlags(Intent.
                        FLAG_ACTIVITY_NEW_TASK);
                    startActivity(InformationIntent);
                }
            }
        });

        //used for updating a post.
        postUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                if(mAuth.getCurrentUser().getUid().equals(post_uid)
                    ) {
                    Intent InformationIntent = new Intent(
                        CommentsActivity.this, UpdateActivity.class)
                        ;
                    InformationIntent.putExtra("post_id", mPost_key
                        );
                    InformationIntent.putExtra("location_id",
                        mLocation_key);
                    startActivity(InformationIntent);
                }
            }
        });
    }
}
```

Listing 12.12: CommentsActivity

12.4.3 InformationFlipActivity

```
package adamoconnor.informe.PostingInformationAndComments;

import android.app.FragmentManager;
import android.content.Intent;
```

```
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Handler;
import android.support.v4.app.NavUtils;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;
import adamoconnor.informe.MapsAndGeofencing.MapsActivity;
import adamoconnor.informe.R;
import static adamoconnor.informe.MapsAndGeofencing.Place.
    setMonumentName;

/**
 * Demonstrates a "card-flip" animation using custom fragment
     transactions ({@link
 * android.app.FragmentTransaction#setCustomAnimations(int, int)}).
 *
 * <p>This sample shows an "info" action bar button that shows the
     back of a "card", rotating the
 * front of the card out and the back of the card in. The reverse
     animation is played when the user
 * presses the system Back button or the "photo" action bar button
     .</p>
 */
public class InformationFlipActivity extends AppCompatActivity
        implements FragmentManager.OnBackStackChangedListener {
    /**
     * A handler object, used for deferring UI operations.
     */
    private Handler mHandler = new Handler();
    private String monumentName = null;

    /**
     * Whether or not we're showing the back of the card (otherwise
         showing the front).
     */
    private boolean mShowingBack = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_card_flip);
```

```
        //setting screen orientation to stop fragments view showing
            on eachother.
        this.setRequestedOrientation(ActivityInfo.
           SCREEN_ORIENTATION_PORTRAIT);

        // get the specific monument entered.
        Bundle extras = null;
        if (savedInstanceState == null) {
            extras = getIntent().getExtras();
            if(extras == null) {
                monumentName = null;
            } else {
                monumentName = extras.getString("1");
                setMonumentName(monumentName);
                if(monumentName == null) {
                    monumentName = extras.getString("
                       monumentInformation");
                    setMonumentName(monumentName);
                }
            }
        } else {
            try {
                monumentName = extras.getString("
                   monumentInformation");
                setMonumentName(monumentName);
            } catch (NullPointerException ex) {

            }
        }

        // send to new thread
        MyThread myThread = new MyThread();
        myThread.start();

        // Monitor back stack changes to ensure the action bar
           shows the appropriate
        // button (either "photo" or "info").
        getFragmentManager().addOnBackStackChangedListener(this);
    }

    /**
     * start new thread to use fragment manager
     * to speed things up and allow less computation on
     * the UI thread.
     */
```

```
public class MyThread extends Thread{

    @Override
    public void run() {

        // If there is no saved instance state, add a fragment
           representing the
        // front of the card to this activity. If there is
           saved instance state,
        // this fragment will have already been added to the
           activity.
        getFragmentManager()
                .beginTransaction()
                .add(R.id.container, new
                   InformationFrontFragment())
                .commit();

    }

}

/**
 * used to set the configuration of the orientation of the
 * activity
 * @param newConfig
 * configuration which has been set.
 */
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    //here you can handle orientation change
}

/**
 * create the options menu when activity is created.
 * @param menu
 * imenu reference
 * @return
 * return the action.
 */
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
```

```
        // Add either a "photo" or "finish" button to the action
           bar, depending on which page
        // is currently selected.
        MenuItem item = menu.add(Menu.NONE, R.id.action_flip, Menu.
           NONE,
                mShowingBack
                        ? R.string.action_photo
                        : R.string.action_info);
        item.setIcon(mShowingBack
                ? R.drawable.ic_action_info
                : R.mipmap.posts);
        item.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM);
        return true;
    }

    /**
     * find which item has been selected like an
     * onClickListener.
     * @param item
     * items which can be selected.
     * @return
     * return the boolean.
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                // Navigate "up" the demo structure to the
                   launchpad activity.
                // See http://developer.android.com/design/patterns
                   /navigation.html for more.
                NavUtils.navigateUpTo(this, new Intent(this,
                   MapsActivity.class));
                return true;

            case R.id.action_flip:
                flipCard();
                return true;
        }

        return super.onOptionsItemSelected(item);
    }

    /**
     * used to flip the fragments when user wants to change.
```

```
     */
    private void flipCard() {
        if (mShowingBack) {
            getFragmentManager().popBackStack();
            return;
        }

        // Flip to the back.
        mShowingBack = true;

        // Create and commit a new fragment transaction that adds
           the fragment for the back of
        // the card, uses custom animations, and is part of the
           fragment manager's back stack.
        getFragmentManager()
                .beginTransaction()

                // Replace the default fragment animations with
                   animator resources representing
                // rotations when switching to the back of the card
                   , as well as animator
                // resources representing rotations when flipping
                   back to the front (e.g. when
                // the system Back button is pressed).
                .setCustomAnimations(
                        R.animator.card_flip_right_in, R.animator.
                            card_flip_right_out,
                        R.animator.card_flip_left_in, R.animator.
                            card_flip_left_out)

                // Replace any fragments currently in the container
                    view with a fragment
                // representing the next page (indicated by the
                   just-incremented currentPage
                // variable).
                .replace(R.id.container, new
                   InformationBackCommentsFragment())

                // Add this transaction to the back stack, allowing
                    users to press Back
                // to get to the front of the card.
                .addToBackStack(null)

                // Commit the transaction.
                .commit();
```

```
        // Defer an invalidation of the options menu (on modern
            devices, the action bar). This
        // can't be done immediately because the transaction may
            not yet be committed. Commits
        // are asynchronous in that they are posted to the main
            thread's message loop.
        mHandler.post(new Runnable() {
            @Override
            public void run() {
                invalidateOptionsMenu();
            }
        });
    }

    /**
     * check when fragments on the back of the stack.
     */
    @Override
    public void onBackStackChanged() {
        mShowingBack = (getFragmentManager().getBackStackEntryCount
            () > 0);

        // When the back stack changes, invalidate the options menu
            (action bar).
        invalidateOptionsMenu();
    }

    /**
     * onResume method for activity.
     */
    @Override
    public void onResume() {
        super.onResume();  // Always call the superclass method
            first

    }

    /**
     * onPause method used for activity.
     */
    @Override
    public void onPause() {
        super.onPause();  // Always call the superclass method
            first
```

```
    }

}
```

Listing 12.13: InformationFlipActivity

12.4.4 InformationFrontFragment

```
package adamoconnor.informe.PostingInformationAndComments;

import android.app.Fragment;
import android.app.ProgressDialog;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
import com.daimajia.slider.library.Animations.DescriptionAnimation;
import com.daimajia.slider.library.SliderLayout;
import com.daimajia.slider.library.SliderTypes.BaseSliderView;
import com.daimajia.slider.library.SliderTypes.TextSliderView;
import com.daimajia.slider.library.Tricks.ViewPagerEx;
import adamoconnor.informe.MapsAndGeofencing.Place;
import adamoconnor.informe.R;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import java.util.HashMap;
import java.util.Locale;
import static adamoconnor.informe.MapsAndGeofencing.Place.
    getMonumentName;

public class InformationFrontFragment extends Fragment implements
    BaseSliderView.OnSliderClickListener, ViewPagerEx.
    OnPageChangeListener {

    //declare text to speech.
    private TextToSpeech repeatText;
```

```
private Boolean soundButton = true;

//declare textView's
private TextView information;
private TextView title;

//declare Image button
private ImageButton listenButton;

//declare slider.
private SliderLayout sliderLayout;

//declare hash-map for loading images etc.
private HashMap<String,String> Hash_file_maps ;

//declare monumentName string
private String monumentName = null;

//declare progress dialog.
private ProgressDialog infoProgress;

//declare database reference
private DatabaseReference database;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
   container,
                         Bundle savedInstanceState) {

    // get reference to the Place class
    Place activity = new Place();
    //retrieve the monument name of the monument.
    monumentName = activity.getMonumentName();

    // get the instance to the firebase reference.
    database = FirebaseDatabase.getInstance().getReference();
    // keep the database synced.
    database.keepSynced(true);

    final View view = inflater.inflate(R.layout.
       activity_information, container, false);

    //declare reference to the layout buttons.
```

```
listenButton = (ImageButton)view.findViewById(R.id.
    informationListen);
listenButton.setImageResource(R.drawable.soundicon);

//declare reference to the textView's.
information = (TextView)view.findViewById(R.id.
    informationText);
title = (TextView)view.findViewById(R.id.title);

//declare the sliderLayout.
sliderLayout = (SliderLayout)view.findViewById(R.id.slider)
    ;

//declare the progress bar to load up.
infoProgress = ProgressDialog.show(getActivity(), "Loading
    ...", "Please wait...", true);

// load the data for the information of the monument.
LoadData();

//listen button start textToSpeech
listenButton.setOnClickListener(new View.OnClickListener()
    {
     public void onClick(View v) {
         if(soundButton) {
             listenButton.setImageResource(R.drawable.
                 soundicon);

         }
         else {
             listenButton.setImageResource(R.drawable.
                 muteicon);
         }
         TextToSpeech(v);
     }
});

//start getting the information needed to speak.
repeatText=new TextToSpeech(getContext(), new TextToSpeech.
    OnInitListener() {

     @Override
     public void onInit(int status) {

         if(status == TextToSpeech.SUCCESS){
```

```
                int result=repeatText.setLanguage(Locale.
                    ENGLISH);
                if(result==TextToSpeech.LANG_MISSING_DATA ||
                        result==TextToSpeech.LANG_NOT_SUPPORTED
                            ){
                    Log.e("error", "This Language is not
                        supported");
                }
            }
            else
                Log.e("error", "Initilization Failed!");
        }
    });
    return view;
}

/**
 * when the activity is paused shutdown the textToSpeech.
 */
@Override
public void onPause() {

    if(repeatText != null){
        repeatText.stop();
        repeatText.shutdown();
    }
    super.onPause();
}

/**
 * text to speech method used when using
 * specific sound button.
 */
public void TextToSpeech(View view){

    if(repeatText.isSpeaking()) {

        repeatText.stop();
        soundButton = true;
    }
    // If it's not playing
    else {
        // Resume the music player
        ConvertTextToSpeech();
        soundButton = false;
```

```
        }

    }

    /**
     * convert the following text to speech
     * that was retrieved from firebase.
     */
    private void ConvertTextToSpeech() {
        String text = information.getText().toString();
        if(!TextUtils.isEmpty(text))
        {
            repeatText.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }else {
            repeatText.speak(text, TextToSpeech.QUEUE_FLUSH, null);
        }
    }

    /**
     * load the specific data of the historic information
     * this method retrieves the text for the activity
     * with a reference to load images.
     */
    private void LoadData() {

        //reference to firebase database.
        DatabaseReference myRef = database.child("ruin");
        myRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {

                //reference to load images method.
                informationImages();

                try {

                    String info = (dataSnapshot.child(
                       getMonumentName().trim()).getValue().
                       toString());
                    title.setText(monumentName.trim());
                    String regex = "\\[|\\]";
                    info = info.replaceAll(regex, "");
                    information.setText(info);

                } catch(NullPointerException ex) {
```

```
                LoadData();
            }

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {}

    });
}

/**
 * loading the images and add to the slider which will keep
 * spinning around.
 */
public void informationImages() {

    // pass the name of the monument
    //get the reference to the database.
    DatabaseReference myRef = database.child("images").child(
        getMonumentName().trim());
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot alerts) {

            Hash_file_maps = new HashMap<>();

            // load the images add numbers.
            int count = 1;
            for(DataSnapshot alert : alerts.getChildren()) {
                // pass the monument name
                Hash_file_maps.put(monumentName.trim()+", "+
                    count, alert.getValue().toString());
                count++;
            }

            // add the images to the slider.
            for(String name : Hash_file_maps.keySet()){

                TextSliderView textSliderView = new
                    TextSliderView(getContext());
                textSliderView
                        .description(name)
                        .image(Hash_file_maps.get(name))
```

```
                        .setScaleType(BaseSliderView.ScaleType.
                            Fit)
                        .setOnSliderClickListener(
                            InformationFrontFragment.this);
                textSliderView.bundle(new Bundle());
                textSliderView.getBundle()
                        .putString("extra",name);
                sliderLayout.addSlider(textSliderView);
            }
            // set the slider transform and duration of the
                slider.
            sliderLayout.setPresetTransformer(SliderLayout.
                Transformer.Accordion);
            sliderLayout.setPresetIndicator(SliderLayout.
                PresetIndicators.Center_Bottom);
            sliderLayout.setCustomAnimation(new
                DescriptionAnimation());
            sliderLayout.setDuration(3000);
            sliderLayout.addOnPageChangeListener(
                InformationFrontFragment.this);

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {}

    });
    // cancel the progress.
    infoProgress.cancel();
}

/**
 * stop the slider when activity closed.
 */
@Override
public void onStop() {
    sliderLayout.stopAutoCycle();
    super.onStop();
}

/**
 * used when the user clicks the image.
 * @param slider
 * reference to the slider.
 */
```

```
    @Override
    public void onSliderClick(BaseSliderView slider) {
        Toast.makeText(getContext(),slider.getBundle().get("extra")
            + "", Toast.LENGTH_SHORT).show();
    }

    /**
     * scroll of the slider
     * @param position
     * position of image
     * @param positionOffset
     * offset of image
     * @param positionOffsetPixels
     * offset of pixels
     */
    @Override
    public void onPageScrolled(int position, float positionOffset,
       int positionOffsetPixels) {}

    /**
     * which page is selected.
     * @param position
     * position of the page.
     */
    @Override
    public void onPageSelected(int position) {}

    /**
     * page which has changed.
     * @param state
     * which state is the page in.
     */
    @Override
    public void onPageScrollStateChanged(int state) {}
}
```

Listing 12.14: InformationFrontFragment

12.4.5 InformationBackCommentsFragment

```
package adamoconnor.informe.PostingInformationAndComments;

import android.app.Fragment;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;
import adamoconnor.informe.R;
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.like.LikeButton;
import com.squareup.picasso.Picasso;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import static adamoconnor.informe.MapsAndGeofencing.Place.
    getMonumentName;


public class InformationBackCommentsFragment extends Fragment {

    // declare recycler view.
    private RecyclerView commentList;

    //declare boolean.
    private boolean mValueLike = false;

    //declare database references.
    private DatabaseReference mDatabase;
    private DatabaseReference mDatabaseLike;

    //declare firebase authentication.
    private FirebaseAuth mAuth;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
        container,
```

```
                                Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_card_back,
            container, false);
        setHasOptionsMenu(true);

        // get instance of the firebase databases.
        mDatabase = FirebaseDatabase.getInstance().getReference().
            child("comments").child(getMonumentName());
        mDatabaseLike = FirebaseDatabase.getInstance().getReference
            ().child("likes");

        //get the firebase authentication instance.
        mAuth = FirebaseAuth.getInstance();

        //keep both database references synced.
        mDatabase.keepSynced(true);
        mDatabaseLike.keepSynced(true);

        //declare the RecyclerView with reference to the layout.
        commentList = (RecyclerView) view.findViewById(R.id.
            comment_list);
        commentList.setHasFixedSize(true);
        commentList.setLayoutManager(new LinearLayoutManager(
            getActivity()));
        return view;
    }

    /**
     * when the activity is called.
     */
    @Override
    public void onStart() {
        super.onStart();

        FirebaseRecyclerAdapter<Comments, CommentHolder>
            firebaseRecyclerAdapter = new FirebaseRecyclerAdapter<
            Comments, CommentHolder>(
                    Comments.class,
                    R.layout.comments_row,
                    CommentHolder.class,
                    mDatabase

        ) {

            /**
```

```
     * used to populate each post on the specific monument.
     * @param viewHolder
     * holder of information
     * @param model
     * used as a reference to comment model.
     * @param position
     * where on the view each attribute is located.
     */
    protected void populateViewHolder(final CommentHolder
        viewHolder, final Comments model, int position) {

        final String post_key = getRef(position).getKey();

        viewHolder.setTitle(model.getTitle());
        viewHolder.setDescription(model.getDescription());
        viewHolder.setImage(getActivity(),model.getImage())
            ;
        viewHolder.setProfileImage(getActivity(), model.
            getProfile());
        viewHolder.setUsername(model.getUsername());
        viewHolder.setTime(model.getTimestamp());
        viewHolder.setLikes(post_key);

        viewHolder.mView.setOnClickListener(new View.
            OnClickListener() {
            @Override
            public void onClick(View v) {

                Intent viewComments = new Intent(
                    getActivity(), CommentsActivity.class);
                viewComments.putExtra("post_id", post_key);
                viewComments.putExtra("location_id",
                    getMonumentName());
                startActivity(viewComments);
            }
        });

        // used to set like button
        viewHolder.likeButton.setOnClickListener(new View.
            OnClickListener() {
            @Override
            public void onClick(View v) {

                mValueLike = true;
```

```
                        mDatabaseLike.addValueEventListener(new
                            ValueEventListener() {
                             @Override
                             public void onDataChange(DataSnapshot
                                dataSnapshot) {
                                 if(mValueLike) {

                                     if (dataSnapshot.child(post_key
                                        ).hasChild(mAuth.
                                        getCurrentUser().getUid()))
                                        {

                                         viewHolder.likeButton.
                                            setLiked(false);
                                         mDatabaseLike.child(
                                            post_key).child(mAuth.
                                            getCurrentUser().getUid
                                            ()).removeValue();
                                         mValueLike = false;
                                     } else {
                                         viewHolder.likeButton.
                                            setLiked(true);
                                         mDatabaseLike.child(
                                            post_key).child(mAuth.
                                            getCurrentUser().getUid
                                            ()).setValue("random");
                                         mValueLike = false;
                                     }
                                 }
                             }

                             @Override
                             public void onCancelled(DatabaseError
                                databaseError) {

                             }
                        });
                    }
                });

            }
        };

        commentList.setAdapter(firebaseRecyclerAdapter);
    }
```

```
/**
 * comment holder used to populate the layouts
 * attributes.
 */
public static class CommentHolder extends RecyclerView.
    ViewHolder {

    View mView;
    TextView post_NumLikes;
    DatabaseReference mDatabaseLikes;
    FirebaseAuth mAuth;
    LikeButton likeButton;

    public CommentHolder(View itemView) {
        super(itemView);

        mView = itemView;

        post_NumLikes = (TextView) mView.findViewById(R.id.
            commentThumbsUp);
        likeButton = (LikeButton) mView.findViewById(R.id.
            commentLike);

        mDatabaseLikes = FirebaseDatabase.getInstance().
            getReference().child("likes");
        mAuth = FirebaseAuth.getInstance();
        mDatabaseLikes.keepSynced(true);

    }

    /**
     * setting the number of likes of the posted monument
     * @param post_key
     * get the post key to find number of likes.
     */
    public void setLikes(final String post_key) {

        mDatabaseLikes.addValueEventListener(new
            ValueEventListener() {
             @Override
             public void onDataChange(DataSnapshot dataSnapshot)
                 {
```

```
                int numOfLikes = (int) dataSnapshot.child(
                    post_key).getChildrenCount();

                post_NumLikes.setText("Likes - "+numOfLikes);

                if(dataSnapshot.child(post_key).hasChild(mAuth.
                    getCurrentUser().getUid())) {
                     likeButton.setLiked(true);
                } else {
                     likeButton.setLiked(false);
                }

            }

            @Override
            public void onCancelled(DatabaseError databaseError
                ) {

            }
        });

    }

    /**
     * setting the title
     * @param title
     * setting the title to textView.
     */
    public void setTitle(String title) {

        TextView post_title = (TextView) mView.findViewById(R.
            id.commentTitle);
        post_title.setText(title);

    }

    /**
     * setting the description
     * @param description
     * setting the description to textView
     */
    public void setDescription(String description) {

        TextView post_description = (TextView) mView.
            findViewById(R.id.commentDescription);
```

```
        post_description.setText(description);

    }

    /**
     * setting the image of the post
     * @param mContext
     * the context of the activity
     * @param image
     * the url string to load with picasso
     */
    public void setImage(final Context mContext,final String
        image) {

        final ImageView post_image = (ImageView) mView.
            findViewById(R.id.commentImage);
        Picasso.with(mContext).load(image).resize(1200,750).
            into(post_image);

    }

    /**
     * setting the username into textView
     * @param username
     * username to set to textView.
     */
    public void setUsername(String username) {

        TextView post_username = (TextView) mView.findViewById(
            R.id.commentUsername);
        post_username.setText("Posted by : "+username);
    }

    /**
     * setting the time of the post description.
     * @param timestamp
     * long timestamp to change to data and time.
     */
    public void setTime(Long timestamp) {

        TextView post_username = (TextView) mView.findViewById(
            R.id.commentTime);
        try {
```

```
                DateFormat sdf = new SimpleDateFormat().
                    getDateTimeInstance();
                Date netDate = (new Date(timestamp));
                post_username.setText(sdf.format(netDate));
            }catch(Exception ex) {

            }

        }

        /**
         * setting of the profile image of the user who posted.
         * @param mContext
         * activity which is shown.
         * @param profile
         * string reference for the url to load.
         */
        public void setProfileImage(final Context mContext, final
            String profile) {

            ImageView post_image = (ImageView) mView.findViewById(R
                .id.commentProfile);

            Picasso.with(mContext).load(profile)
                    .placeholder(R.drawable.defaulticon).error(R.
                        drawable.defaulticon)
                    .into(post_image);

        }
    }

    /**
     * creation of the menu options.
     * @param menu
     * menu reference.
     * @param inflater
     * inflater reference.
     */
    @Override
    public void onCreateOptionsMenu(Menu menu, MenuInflater
        inflater) {
        // TODO Add your menu entries here
        inflater.inflate(R.menu.menu_custom, menu);
        super.onCreateOptionsMenu(menu, inflater);
    }
```

```
    /**
     * post information on historic monument.
     * @param item
     * selected item
     * @return
     * the options
     */
    public boolean onOptionsItemSelected(MenuItem item) {

        if(item.getItemId() == R.id.action_add) {
            Intent postActivity = new Intent(getContext(),
                PostActivity.class);
            startActivity(postActivity);
        }
        return super.onOptionsItemSelected(item);
    }

    /**
     * resume method.
     */
    @Override
    public void onResume() {
        super.onResume();  // Always call the superclass method
            first

    }

    /**
     * pause method.
     */
    @Override
    public void onPause() {
        super.onPause();  // Always call the superclass method
            first

    }

}
```

Listing 12.15: InformationBackCommentsFragment

12.4.6 PostActivity

```
package adamoconnor.informe.PostingInformationAndComments;
```

```
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.Toast;
import adamoconnor.informe.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ServerValue;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.squareup.picasso.Picasso;
import java.util.Random;
import static adamoconnor.informe.MapsAndGeofencing.Place.
    getMonumentName;

public class PostActivity extends AppCompatActivity {

    //declare context.
    private Context mContext;

    //declare image button
    private ImageButton selectImage;

    //declare editTexts.
    private EditText postTitle;
```

```
    private EditText postDescription;

    //declare button.
    private Button submitPost;

    //declare Uri
    private Uri imageUri = null;

    //declare gallery request.
    private static final int GALLERY_REQUEST = 1;

    //declare progress dialog.
    private ProgressDialog mProgress;

    //declare database references
    private DatabaseReference mDatabase;
    private DatabaseReference mDatabaseUsers;
    private DatabaseReference mDatabaseUsersProfilePicture;

    //declare firebase authentication.
    private FirebaseAuth mAuth;

    //declare firebase user.
    private FirebaseUser mCurrentUser;

    //declare firebase storage.
    private StorageReference mStorage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_post);

        //setting screen orientation to stop fragments view showing
            on eachother.
        this.setRequestedOrientation(ActivityInfo.
           SCREEN_ORIENTATION_PORTRAIT);

        // declare context reference
        mContext = this;

        //declare authentication instance
        mAuth = FirebaseAuth.getInstance();

        // get the users authentcation
```

```
mCurrentUser = mAuth.getCurrentUser();

// get the storage and database instances with each
   reference
mStorage = FirebaseStorage.getInstance().getReference();
mDatabase = FirebaseDatabase.getInstance().getReference().
   child("comments").child(getMonumentName());
mDatabaseUsers = FirebaseDatabase.getInstance().
   getReference().child("users").child(mCurrentUser.getUid
   ());
mDatabaseUsersProfilePicture = FirebaseDatabase.getInstance
   ().getReference().child("users").child(mCurrentUser.
   getUid()).child("image");

//reference to selecting image.
selectImage = (ImageButton) findViewById(R.id.imageSelect);

//reference to edit text fields
postTitle = (EditText) findViewById(R.id.title);
postDescription = (EditText) findViewById(R.id.description)
   ;

//reference to the button on the layout
submitPost = (Button) findViewById(R.id.submitButton);

//setting reference to progress dialog.
mProgress = new ProgressDialog(this);

//selecting of the image listener.
selectImage.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        new Thread() {
            @Override
            public void run() {

                Intent galleryIntent = new Intent(Intent.
                   ACTION_GET_CONTENT);
                //only look at images.
                galleryIntent.setType("image/*");
                startActivityForResult(galleryIntent,
                   GALLERY_REQUEST);
            }
        }.start();
```

```
            }
        });

        //submit the post.
        submitPost.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                new LongOperation().execute();
            }
        });

    }

    /**
     * random string generator for the uploaded image
     * @return
     * random string to return
     */
    public static String random() {
        Random generator = new Random();
        StringBuilder randomStringBuilder = new StringBuilder();
        int randomLength = generator.nextInt(10);
        char tempChar;
        for (int i = 0; i < randomLength; i++){
            tempChar = (char) (generator.nextInt(96) + 32);
            randomStringBuilder.append(tempChar);
        }
        return randomStringBuilder.toString();
    }

    /**
     * display the image on the ImageButton
     * @param requestCode
     * get request code
     * @param resultCode
     * get result code
     * @param data
     * retrieve image.
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode
        , final Intent data) {
        super.onActivityResult(requestCode,resultCode,data);
```

```
        if(requestCode == GALLERY_REQUEST && resultCode ==
            RESULT_OK) {

            this.runOnUiThread(new Runnable() {
                public void run() {
                    imageUri = data.getData();
                    Picasso.with(mContext)
                            .load(imageUri)
                            .centerCrop().resize(1080,780)
                            .into(selectImage);
                }
            });
        }
    }

    /**
     * onResume method.
     */
    @Override
    public void onResume() {
        super.onResume();  // Always call the superclass method
            first

    }

    /**
     * onPause method.
     */
    @Override
    public void onPause() {
        super.onPause();  // Always call the superclass method
            first

    }

    /**
     * long operation to post the information about the specific
     * monument.
     */
    private class LongOperation extends AsyncTask<Void,String,
        String> {

        /**
         * pre execution used for the progress
         * dialog.
```

```
     */
    @Override
    protected void onPreExecute()
    {
        mProgress = new ProgressDialog(PostActivity.this);
        mProgress.setMessage("Processing Request...");
        mProgress.setIndeterminate(false);
        mProgress.setCancelable(false);
        mProgress.show();
        super.onPreExecute();
    }

    /**
     * run a method in the background.
     * @param params
     * pass parameters
     * @return
     * return execution.
     */
    @Override
    protected String doInBackground(Void... params) {

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        startPosting();
        return "Executed";
    }

    private void startPosting() {

        final String titleValue = String.valueOf(postTitle.
           getText()).trim();
        final String descriptionValue = String.valueOf(
           postDescription.getText()).trim();

        if (!TextUtils.isEmpty(titleValue) && !TextUtils.
           isEmpty(descriptionValue) && imageUri != null) {

            StorageReference filePath = mStorage.child("
               Comment_Images").child(random());
```

```
filePath.putFile(imageUri).addOnSuccessListener(new
    OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override

    public void onSuccess(UploadTask.TaskSnapshot
        taskSnapshot) {
     @SuppressWarnings("VisibleForTests")
     final Uri downloadUrl = taskSnapshot.
         getDownloadUrl();
     final DatabaseReference newPost = mDatabase
         .push();

     mDatabaseUsers.addValueEventListener(new
         ValueEventListener() {
      @Override
      public void onDataChange(DataSnapshot
          dataSnapshot) {

          newPost.child("title").setValue(
              titleValue);
          newPost.child("description").
              setValue(descriptionValue);
          newPost.child("image").setValue(
              downloadUrl.toString());
          newPost.child("uid").setValue(
              mCurrentUser.getUid());
          newPost.child("username").setValue(
              dataSnapshot.child("name").
              getValue());
          newPost.child("timestamp").setValue
              (ServerValue.TIMESTAMP);
          newPost.child("profile").setValue(
              dataSnapshot.child("image").
              getValue()).
              addOnCompleteListener(new
              OnCompleteListener<Void>() {
               @Override
               public void onComplete(@NonNull
                   Task<Void> task) {

                   new Thread(new Runnable() {
                       @Override
                       public void run() {
                           try {
```

```
                                    Thread.sleep
                                        (5000);
                                } catch (
                                    InterruptedException
                                    e) {
                                    e.
                                        printStackTrace
                                        ();
                                }
                            }
                        }).start();

                        if(task.isSuccessful()) {
                            mProgress.dismiss();
                            Intent
                                startCommentFragment
                                = new Intent(
                                PostActivity.this,
                                InformationFlipActivity
                                .class);
                            startCommentFragment.
                                setFlags(Intent.
                                FLAG_ACTIVITY_CLEAR_TOP
                                );
                            startActivity(
                                startCommentFragment
                                );
                        }
                        else {
                            mProgress.dismiss();
                            Toast.makeText(
                                PostActivity.this, "
                                OOps looks like
                                something went wrong
                                , please check
                                internet connection
                                ...",
                                    Toast.
                                        LENGTH_LONG)
                                    .show();
                        }

                    }
                });
            }
```

```
                                @Override
                                public void onCancelled(DatabaseError
                                    databaseError) {

                                }
                            });

                        }

                    });
                } else {
                    mProgress.dismiss();
                    Toast.makeText(PostActivity.this, "Please fill in
                        the form...",
                            Toast.LENGTH_LONG).show();
                }
            }

            @Override
            protected void onPostExecute(String result) {

                super.onPostExecute(result);

            }


        }

    }
```

Listing 12.16: PostActivity

12.4.7 UpdateActivity

```
package adamoconnor.informe.PostingInformationAndComments;

import android.app.ProgressDialog;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.AsyncTask;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import adamoconnor.informe.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ServerValue;
import com.google.firebase.database.ValueEventListener;
import com.squareup.picasso.Picasso;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

import de.hdodenhof.circleimageview.CircleImageView;

public class UpdateActivity extends AppCompatActivity {

    //declare the specific keys needed for post.
    private String mPost_key = null;
    private String mLocation_key = null;
    private String post_uid = null;

    //declare database reference.
    private DatabaseReference mDatabase;

    //declare image views textviews and buttons.
    private ImageView postImage;
    private EditText postTitle;
    private EditText postDescription;
    private TextView postTimestamp;
    private TextView postUsername;
    private CircleImageView postProfile;
    private Button postUpdate;

    //declare progress dialog
    private ProgressDialog mProgress;

    //declare firebase authentication.
```

```
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update);

        //setting screen orientation to stop fragments view showing
            on eachother.
        this.setRequestedOrientation(ActivityInfo.
            SCREEN_ORIENTATION_PORTRAIT);

        // get the extras sent by an intent.
        mPost_key = getIntent().getExtras().getString("post_id");
        mLocation_key = getIntent().getExtras().getString("
            location_id");

        //get the reference to the database
        mDatabase = FirebaseDatabase.getInstance().getReference().
            child("comments").child(mLocation_key);
        //authentication of the user.
        mAuth = FirebaseAuth.getInstance();

        // get the reference's to the buttons text-views and edit-
            texts.
        postProfile = (CircleImageView) findViewById(R.id.
            commentProfile);
        postUsername = (TextView) findViewById(R.id.commentUsername
            );
        postTimestamp = (TextView) findViewById(R.id.commentTime);
        postImage = (ImageView) findViewById(R.id.commentImage);
        postTitle = (EditText) findViewById(R.id.commentTitle);
        postDescription = (EditText) findViewById(R.id.
            commentDescription);
        postUpdate = (Button) findViewById(R.id.updatePost);

        // used for the retrieval of information to the text-views
            etc.
        mDatabase.child(mPost_key).addValueEventListener(new
            ValueEventListener() {
             @Override
             public void onDataChange(DataSnapshot dataSnapshot) {

                 String post_title = (String)dataSnapshot.child("
                     title").getValue();
```

```
                String post_desc = (String)dataSnapshot.child("
                    description").getValue();
                String post_image = (String)dataSnapshot.child("
                    image").getValue();

                String post_profile = (String)dataSnapshot.child("
                    profile").getValue();
                String post_username = (String)dataSnapshot.child("
                    username").getValue();
                Long post_timestamp = (Long)dataSnapshot.child("
                    timestamp").getValue();
                post_uid = (String)dataSnapshot.child("uid").
                    getValue();

                postTitle.setText(post_title);
                postDescription.setText(post_desc);
                postUsername.setText(post_username);

                // setting the date and time of the post.
                try {
                    DateFormat sdf = new SimpleDateFormat().
                        getDateTimeInstance();
                    Date netDate = (new Date(post_timestamp));
                    postTimestamp.setText(sdf.format(netDate));
                } catch(NullPointerException ex) {
                    ex.getStackTrace();
                }

                //load the images with picasso
                Picasso.with(UpdateActivity.this).load(post_image).
                    resize(1200,750).into(postImage);
                Picasso.with(UpdateActivity.this).load(post_profile
                    )
                        .placeholder(R.drawable.defaulticon).error(
                            R.drawable.defaulticon)
                        .into(postProfile);
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {

            }
        });

        // used to update the information from the user.
```

```
        postUpdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                new LongOperation().execute();

            }
        });

    }

    /**
     * long operation to update the information about the specific
     * monument.
     */
    private class LongOperation extends AsyncTask<Void, String,
        String> {

        @Override
        protected void onPreExecute()
        {
            mProgress = new ProgressDialog(UpdateActivity.this);
            mProgress.setMessage("Updating Post...");
            mProgress.setIndeterminate(false);
            mProgress.setCancelable(false);
            mProgress.show();
            super.onPreExecute();
        }

        @Override
        protected String doInBackground(Void... params) {

            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            startUpdating();
            return "Executed";
        }

        private void startUpdating() {

            // if user is authenticated again you can post the
               update.
```

```
if (mAuth.getCurrentUser().getUid().equals(post_uid)) {

    mDatabase.child(mPost_key).
        addListenerForSingleValueEvent(new
        ValueEventListener() {

         @Override
         public void onDataChange(DataSnapshot
            dataSnapshot) {

             dataSnapshot.getRef().child("title").
                setValue(postTitle.getText().toString().
                trim());
             dataSnapshot.getRef().child("description").
                setValue(postDescription.getText().
                toString().trim());
             dataSnapshot.getRef().child("timestamp").
                setValue(ServerValue.TIMESTAMP).
                addOnCompleteListener(new
                OnCompleteListener<Void>() {
                 @Override
                 public void onComplete(@NonNull Task<
                    Void> task) {

                     new Thread(new Runnable() {
                         @Override
                         public void run() {
                             try {
                                 Thread.sleep(5000);
                             } catch (
                                InterruptedException e)
                                {
                                 e.printStackTrace();
                             }
                         }
                     }).start();

                     if (task.isSuccessful()) {
                         mProgress.dismiss();
                         Intent startCommentFragment =
                            new Intent(UpdateActivity.
                            this,
                            InformationFlipActivity.
                            class);
```

```
                                    startCommentFragment.setFlags(
                                        Intent.
                                        FLAG_ACTIVITY_CLEAR_TOP);
                                    startActivity(
                                        startCommentFragment);
                                } else {
                                    mProgress.dismiss();
                                    Toast.makeText(UpdateActivity.
                                        this, "OOps looks like
                                        something went wrong, please
                                         check internet connection
                                        ...",
                                            Toast.LENGTH_LONG).show
                                                ();
                                }

                            }
                        });
                    }

                    @Override
                    public void onCancelled(DatabaseError
                        databaseError) {

                    }
                });
            }
        }

        @Override
        protected void onPostExecute(String result) {

            super.onPostExecute(result);

        }


    }

}
```

Listing 12.17: UpdateActivity

12.5 Settings

12.5.1 AppCompatPreferenceActivity

```
package adamoconnor.informe.Settings;

import android.content.res.Configuration;
import android.os.Bundle;
import android.preference.PreferenceActivity;
import android.support.annotation.LayoutRes;
import android.support.annotation.NonNull;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatDelegate;
import android.view.MenuInflater;
import android.view.View;
import android.view.ViewGroup;

/**
 * A {@link PreferenceActivity} which implements and proxies the
    necessary calls
 * to be used with AppCompat.
 */
public abstract class AppCompatPreferenceActivity extends
   PreferenceActivity {

    // declare the appcompat delegate.
    private AppCompatDelegate mDelegate;

    /**
     * on create called on the start of the activity being called
     * @param savedInstanceState
     * the saved instance of the activity.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        getDelegate().installViewFactory();
        getDelegate().onCreate(savedInstanceState);
        super.onCreate(savedInstanceState);
    }

    /**
     * wafter the on create method has been called.
     * @param savedInstanceState
     * pass the state on which the activity is on.
     */
```

```
@Override
protected void onPostCreate(Bundle savedInstanceState) {
    super.onPostCreate(savedInstanceState);
    getDelegate().onPostCreate(savedInstanceState);
}

/**
 * get the action bar which resides at the top of the activity.
 * @return
 * return the actionbar.
 */
public ActionBar getSupportActionBar() {
    return getDelegate().getSupportActionBar();
}

/**
 * find all the items which reside in the menu.
 * @return
 * the menu which is located in the action bar.
 */
@Override
@NonNull
public MenuInflater getMenuInflater() {
    return getDelegate().getMenuInflater();
}

/**
 * set the content layout.
 * @param layoutResID
 * get the layout of the activity.
 */
@Override
public void setContentView(@LayoutRes int layoutResID) {
   getDelegate().setContentView(layoutResID);}

/**
 * set the content view.
 * @param view
 * get the content view of the activity.
 */
@Override
public void setContentView(View view) {
    getDelegate().setContentView(view);
}
```

```
/**
 * set the content view as well as the view group parameters.
 * @param view
 * the view of the activity.
 * @param params
 * the parameters of the activity.
 */
@Override
public void setContentView(View view, ViewGroup.LayoutParams
    params) {
     getDelegate().setContentView(view, params);
}

/**
 * adding of the content view.
 * @param view
 * adding of view
 * @param params
 * adding of parameters.
 */
@Override
public void addContentView(View view, ViewGroup.LayoutParams
    params) {
     getDelegate().addContentView(view, params);
}

/**
 * after the activity has called onResume.
 */
@Override
protected void onPostResume() {
     super.onPostResume();
     getDelegate().onPostResume();
}

/**
 * when the title has changed.
 * @param title
 * title of activity.
 * @param color
 * colour of title.
 */
@Override
protected void onTitleChanged(CharSequence title, int color) {
     super.onTitleChanged(title, color);
```

```
        getDelegate().setTitle(title);
    }

    /**
     * when the configuration has been changed.
     * @param newConfig
     * new configuration of user.
     */
    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        getDelegate().onConfigurationChanged(newConfig);
    }

    /**
     * when the activity has stopped.
     */
    @Override
    protected void onStop() {
        super.onStop();
        getDelegate().onStop();
    }

    /**
     * when the activity has been destroyed.
     */
    @Override
    protected void onDestroy() {
        super.onDestroy();
        getDelegate().onDestroy();
    }

    /**
     * clear the options menu
     */
    public void invalidateOptionsMenu() {
        getDelegate().invalidateOptionsMenu();
    }

    private AppCompatDelegate getDelegate() {
        if (mDelegate == null) {
            mDelegate = AppCompatDelegate.create(this, null);
        }
        return mDelegate;
    }
```

```
}
```

Listing 12.18: AppCompatPreferenceActivity

12.5.2 SettingsActivity

```
package adamoconnor.informe.Settings;

import android.annotation.TargetApi;
import android.content.Context;
import android.content.res.Configuration;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.preference.ListPreference;
import android.preference.Preference;
import android.preference.PreferenceActivity;
import android.preference.PreferenceFragment;
import android.preference.PreferenceManager;
import android.preference.RingtonePreference;
import android.support.v4.app.NavUtils;
import android.support.v7.app.ActionBar;
import android.text.TextUtils;
import android.view.MenuItem;
import adamoconnor.informe.R;

import java.util.List;

/**
 * A {@link PreferenceActivity} that presents a set of application
    settings. On
 * handset devices, settings are presented as a single list. On
    tablets,
 * settings are split by category, with category headers shown to
    the left of
 * the list of settings.
 * <p>
 * See <a href="http://developer.android.com/design/patterns/
    settings.html">
 * Android Design: Settings</a> for design guidelines and the <a
 * href="http://developer.android.com/guide/topics/ui/settings.html
    ">Settings
 * API Guide</a> for more information on developing a Settings UI.
 */
```

```
public class SettingsActivity extends AppCompatPreferenceActivity {

    /**
     * A preference value change listener that updates the
        preference's summary
     * to reflect its new value.
     */
    private static Preference.OnPreferenceChangeListener
       sBindPreferenceSummaryToValueListener = new Preference.
       OnPreferenceChangeListener() {
        @Override
        public boolean onPreferenceChange(Preference preference,
           Object value) {
            String stringValue = value.toString();

            if (preference instanceof ListPreference) {
                // For list preferences, look up the correct
                   display value in
                // the preference's 'entries' list.
                ListPreference listPreference = (ListPreference)
                   preference;
                int index = listPreference.findIndexOfValue(
                   stringValue);

                // Set the summary to reflect the new value.
                preference.setSummary(
                        index >= 0
                                ? listPreference.getEntries()[index
                                    ]
                                : null);

            } else if (preference instanceof RingtonePreference) {
                // For ringtone preferences, look up the correct
                   display value
                // using RingtoneManager.
                if (TextUtils.isEmpty(stringValue)) {
                    // Empty values correspond to 'silent' (no
                       ringtone).
                    preference.setSummary(R.string.
                       pref_ringtone_silent);

                } else {
                    Ringtone ringtone = RingtoneManager.getRingtone
                       (
```

```
                        preference.getContext(), Uri.parse(
                            stringValue));

                if (ringtone == null) {
                    // Clear the summary if there was a lookup
                        error.
                    preference.setSummary(null);
                } else {
                    // Set the summary to reflect the new
                        ringtone display
                    // name.
                    String name = ringtone.getTitle(preference.
                        getContext());
                    preference.setSummary(name);
                }
            }

        } else {
            // For all other preferences, set the summary to
                the value's
            // simple string representation.
            preference.setSummary(stringValue);
        }
        return true;
    }
};

/**
 * Helper method to determine if the device has an extra-large
    screen. For
 * example, 10" tablets are extra-large.
 */
private static boolean isXLargeTablet(Context context) {
    return (context.getResources().getConfiguration().
        screenLayout
            & Configuration.SCREENLAYOUT_SIZE_MASK) >=
                Configuration.SCREENLAYOUT_SIZE_XLARGE;
}

/**
 * Binds a preference's summary to its value. More specifically
    , when the
 * preference's value is changed, its summary (line of text
    below the
```

```
     * preference title) is updated to reflect the value. The
        summary is also
     * immediately updated upon calling this method. The exact
        display format is
     * dependent on the type of preference.
     *
     * @see #sBindPreferenceSummaryToValueListener
     */
    private static void bindPreferenceSummaryToValue(Preference
        preference) {
        // Set the listener to watch for value changes.
        preference.setOnPreferenceChangeListener(
            sBindPreferenceSummaryToValueListener);

        // Trigger the listener immediately with the preference's
        // current value.
        sBindPreferenceSummaryToValueListener.onPreferenceChange(
            preference,
                PreferenceManager
                        .getDefaultSharedPreferences(preference.
                            getContext())
                        .getString(preference.getKey(), ""));
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setupActionBar();
    }

    /**
     * Set up the {@link android.app.ActionBar}, if the API is
        available.
     */
    private void setupActionBar() {
        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) {
            // Show the Up button in the action bar.
            actionBar.setDisplayHomeAsUpEnabled(true);

        }
    }

    /**
     * set the options menu
```

```
     * @param item
     * passing in the items in the menu
     * @return
     * return the item selected.
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            NavUtils.navigateUpFromSameTask(this);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
    /**
     * {@inheritDoc}
     */
    @Override
    public boolean onIsMultiPane() {
        return isXLargeTablet(this);
    }

    /**
     * {@inheritDoc}
     */
    @Override
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public void onBuildHeaders(List<Header> target) {
        loadHeadersFromResource(R.xml.pref_headers, target);
    }

    /**
     * This method stops fragment injection in malicious
        applications.
     * Make sure to deny any unknown fragments here.
     */
    protected boolean isValidFragment(String fragmentName) {
        return PreferenceFragment.class.getName().equals(
            fragmentName)
                || DataSyncPreferenceFragment.class.getName().
                    equals(fragmentName)
                || NotificationPreferenceFragment.class.getName().
                    equals(fragmentName);
    }
```

```
    /**
     * This fragment shows notification preferences only. It is
        used when the
     * activity is showing a two-pane settings UI.
     */
    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
    public static class NotificationPreferenceFragment extends
       PreferenceFragment {
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.pref_notification);
            setHasOptionsMenu(true);

            // Bind the summaries of EditText/List/Dialog/Ringtone
               preferences
            // to their values. When their values change, their
               summaries are
            // updated to reflect the new value, per the Android
               Design
            // guidelines.
//              bindPreferenceSummaryToValue(findPreference("
   notifications_new_message"));
            bindPreferenceSummaryToValue(findPreference("
               notifications_new_message_ringtone"));
        }

        /**
         * set the options menu
         * @param item
         * passing in the items in the menu
         * @return
         * return the item selected.
         */
        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            int id = item.getItemId();
            if (id == android.R.id.home) {
                addPreferencesFromResource(R.xml.pref_headers);
                return true;
            }
            return super.onOptionsItemSelected(item);
        }
    }
```

```
/**
 * This fragment shows data and sync preferences only. It is
    used when the
 * activity is showing a two-pane settings UI.
 */
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
public static class DataSyncPreferenceFragment extends
    PreferenceFragment {
    @Override

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.pref_battery);
        setHasOptionsMenu(true);

        // Bind the summaries of EditText/List/Dialog/Ringtone
            preferences
        // to their values. When their values change, their
            summaries are
        // updated to reflect the new value, per the Android
            Design
        // guidelines.
    }

    /**
     * set the options menu
     * @param item
     * passing in the items in the menu
     * @return
     * return the item selected.
     */
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            NavUtils.navigateUpFromSameTask(getActivity());
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

```
}
```

Listing 12.19: SettingsActivity

12.5.3 CheckConnectivity

```
package adamoconnor.informe.Settings;

import android.app.Activity;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Build;
import android.provider.Settings;
import android.support.v7.app.AlertDialog;
import android.text.TextUtils;
import adamoconnor.informe.R;

public class CheckConnectivity {

    // declare booleans.
    private static boolean internetOn;
    private static boolean locationOn;

    /**
     * check if internet is on.
     * @return
     * return the boolean if internet is connected.
     */
    public static boolean isInternetOn() {
        return internetOn;
    }

    /**
     * check if location is on.
     * @return
     * return the boolean if location is connected.
     */
    public static boolean isLocationOn() {
        return locationOn;
    }

    /**
     * used to check if internet is connected if not
```

```
     * user is sent to the settings menu to turn internet
     * on.
     * @param context
     * get the activity to display
     */
    public static void startInternetEnabled(final Context context)
       {

        // check the internets connectivity
        ConnectivityManager cm = (ConnectivityManager) context.
           getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
        boolean network_enabled = activeNetwork != null &&
                activeNetwork.isConnectedOrConnecting();

        if (!network_enabled) {
            // notify user
            AlertDialog.Builder dialog = new AlertDialog.Builder(
               context);
            dialog.setMessage(context.getResources().getString(R.
               string.network_not_enabled));
            dialog.setPositiveButton(context.getResources().
               getString(R.string.open_location_settings), new
               DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface
                   paramDialogInterface, int paramInt) {

                    Intent myIntent = new Intent(Settings.
                       ACTION_WIFI_SETTINGS);
                    context.startActivity(myIntent);
                    //go to internet settings.
                }
            });
            dialog.setNegativeButton(context.getString(R.string.
               Cancel), new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface
                   paramDialogInterface, int paramInt) {

                    ((Activity)context).finish();
                }
            });
```

```
            dialog.show();
        }

        network_enabled = activeNetwork != null &&
                activeNetwork.isConnectedOrConnecting();

        if(network_enabled) {
            internetOn = true;
        }
        else {
            internetOn = false;
        }

    }

    /**
     * used to check if location is connected
     * @param context
     * get the activity to display
     */
    private static boolean isLocationEnabled(Context context) {
        int locationMode = 0;
        String locationProviders;

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT){
            try {
                locationMode = Settings.Secure.getInt(context.
                    getContentResolver(), Settings.Secure.
                    LOCATION_MODE);

            } catch (Settings.SettingNotFoundException e) {
                e.printStackTrace();
                return false;
            }

            return locationMode != Settings.Secure.
                LOCATION_MODE_OFF;

        }else{
            locationProviders = Settings.Secure.getString(context.
                getContentResolver(), Settings.Secure.
                LOCATION_PROVIDERS_ALLOWED);
            return !TextUtils.isEmpty(locationProviders);
        }
```

```
}

/**
 * used to check if location is connected if not
 * user is sent to the settings menu to turn internet
 * on.
 * @param context
 * get the activity to display
 */
public static void startLocationEnabled(final Context context)
   {

    if(!isLocationEnabled(context)) {
        // notify user
        AlertDialog.Builder dialog = new AlertDialog.Builder(
           context);
        dialog.setMessage(context.getResources().getString(R.
           string.gps_network_not_enabled));
        dialog.setPositiveButton(context.getResources().
           getString(R.string.open_location_settings), new
           DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface
               paramDialogInterface, int paramInt) {

                Intent myIntent = new Intent( Settings.
                   ACTION_LOCATION_SOURCE_SETTINGS);
                context.startActivity(myIntent);
                //get gps
            }
        });
        dialog.setNegativeButton(context.getString(R.string.
           Cancel), new DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface
               paramDialogInterface, int paramInt) {

                ((Activity)context).finish();

            }
        });
        dialog.show();
    }
```

```
        if(isLocationEnabled(context)) {
            locationOn = true;
        }
        else {
            locationOn = false;
        }

    }


}
```

Listing 12.20: CheckConnectivity

Chapter 13

Appendix C

13.1 AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android
    "
    xmlns:tools="http://schemas.android.com/tools"
    package="adamoconnor.informe">

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not
            required to use
         Google Maps Android API v2, but you must specify either
            coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.
        ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.
        ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.
        WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.
        READ_EXTERNAL_STORAGE" />

    <application
        android:name="android.support.multidex.MultiDexApplication"
        android:allowBackup="true"
        android:icon="@mipmap/informe_logo_dublin"
        android:label="InforMe@Dublin"
```

```
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:replace="android:label">
        <service android:name="adamoconnor.informe.
            MapsAndGeofencing.GeofenceTransitionsIntentService" />
        <!--
             The API key for Google Maps-based APIs is defined as a
                 string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key
                used to sign the APK.
             You need a different API key for each encryption key,
                including the release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release
                targets in src/debug/ and src/release/.
        -->
        <meta-data
             android:name="com.google.android.geo.API_KEY"
             android:value="AIzaSyB5P0wPErDhqzFRMDwmZkPB-Pq-JShltaI"
                 />
        <activity
             android:name="adamoconnor.informe.LoginAndRegister.
                EmailPasswordAuthentication"
             android:label="InforMe@Dublin"
             android:windowSoftInputMode="stateHidden|adjustResize"
             android:configChanges="keyboardHidden|orientation"
             android:screenOrientation="portrait">
             <intent-filter>
                 <action android:name="android.intent.action.MAIN"
                     />

                 <category android:name="android.intent.category.
                     LAUNCHER" />
             </intent-filter>
        </activity>
        <activity
             android:name="adamoconnor.informe.MapsAndGeofencing.
                MapsActivity"
             android:configChanges="keyboardHidden|orientation"
             android:screenOrientation="portrait"
             android:windowSoftInputMode="stateAlwaysHidden" />
        <!--
    ATTENTION: This was auto-generated to add Google Play services
        to your project for
```

```
App Indexing.  See https://g.co/AppIndexing/AndroidStudio for
    more information.
    -->
    <meta-data
        android:name="com.google.android.gms.version"
        android:value="@integer/google_play_services_version"
            />

    <activity
        android:name="adamoconnor.informe.Settings.
            SettingsActivity"
        android:label="@string/title_activity_settings">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="adamoconnor.informe.LoginAndRegister
                .EmailPasswordAuthentication" />
    </activity>
    <activity
        android:name="adamoconnor.informe.MapsAndGeofencing.
            AddInformation"
        android:label="Submit Proposed Site"
        android:windowSoftInputMode="adjustPan">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="adamoconnor.informe.
                MapsAndGeofencing.MapsActivity" />
    </activity>
    <activity
        android:name=".PostingInformationAndComments.
            InformationFlipActivity"
        android:label="Monument Information"
        android:parentActivityName="adamoconnor.informe.
            MapsAndGeofencing.MapsActivity"
        android:configChanges="keyboardHidden|orientation"
        android:screenOrientation="portrait"
        android:theme="@style/AppTheme">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="adamoconnor.informe.
                MapsAndGeofencing.MapsActivity" />
    </activity>
    <activity
        android:name="adamoconnor.informe.
            PostingInformationAndComments.PostActivity"
        android:windowSoftInputMode="adjustPan"
```

```
            android:configChanges="keyboardHidden|orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name="adamoconnor.informe.LoginAndRegister.
                RegisterAccount"
            android:windowSoftInputMode="adjustPan"
            android:configChanges="keyboardHidden|orientation"
            android:screenOrientation="portrait"/>
        <activity  android:name="adamoconnor.informe.
            LoginAndRegister.SetupActivity"
            android:configChanges="keyboardHidden|orientation"
            android:screenOrientation="portrait"/>
        <activity
            android:name="com.theartofdev.edmodo.cropper.
                CropImageActivity"
            android:theme="@style/Base.Theme.AppCompat" />
        <activity
            android:name="adamoconnor.informe.
                PostingInformationAndComments.CommentsActivity"
            android:windowSoftInputMode="adjustPan" />
        <activity
            android:name="adamoconnor.informe.LoginAndRegister.
                ResetActivity"
            android:label="Reset Password">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="adamoconnor.informe.LoginAndRegister
                    .EmailPasswordAuthentication"
                android:configChanges="keyboardHidden|orientation"
                android:screenOrientation="portrait"/>
        </activity>
        <activity  android:name="adamoconnor.informe.
            PostingInformationAndComments.UpdateActivity"
            android:configChanges="keyboardHidden|orientation"
            android:screenOrientation="portrait"/>
    </application>

</manifest>
```

Listing 13.1: AndroidManifest

Chapter 14

Appendix D

14.1 Google-services.json


```
{
  "project_info": {
    "project_number": "112287408321",
    "firebase_url": "https://authentication-3439c.firebaseio.com",
    "project_id": "authentication-3439c",
    "storage_bucket": "authentication-3439c.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:112287408321:android:
            e6c04824b13f8099",
        "android_client_info": {
          "package_name": "com.example.adamoconnor.authentication"
        }
      },
      "oauth_client": [
        {
          "client_id": "112287408321-
              lu5gj7nmsqrr5ikir8g658stuvorecrs.apps.
              googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyBYx8_8ZyQEILSScPu8aRSn2D9g_00hysk"
        }
```

```
      ],
      "services": {
        "analytics_service": {
          "status": 1
        },
        "appinvite_service": {
          "status": 1,
          "other_platform_oauth_client": []
        },
        "ads_service": {
          "status": 2
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Listing 14.1: google-services

Bibliography

F. Alizadeh-Shabdiz, R.K. JONES, E.J. MORGAN, and M.G. Shean. Location-based services that choose location algorithms based on number of detected access points within range of user device, December 4 2007. URL https://www.google.com/patents/US7305245. US Patent 7,305,245.

GSM Arena. Samsung Galaxy S7 edge. Online, 2016. URL http://www.cellular.co.za/cellphone_inventor.htm. [Accessed 16 October 2016].

Ulrich Bareth and Axel Kupper. Energy-efficient position tracking in proactive location-based services for smartphone environments. In *2011 IEEE 35th Annual Computer Software and Applications Conference*, pages 516–521. IEEE, 2011.

Keith Cheverst, Nigel Davies, Keith Mitchell, Adrian Friday, and Christos Efstratiou. Developing a context-aware electronic tourist guide. *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, 2000. doi: 10.1145/332040.332047.

E. F. Codd. *Further Normalization of the Database Relational Model*, chapter Database Systems, Courant Computer Science Symposia 6, pages 65–98. Prentice-Hall, Englewood Cliffs, N.J, USA, 1972.

Martin Cooper. Inventor Of The Cellphone. Online, 2006. URL http://www.cellular.co.za/cellphone_inventor.htm. [Accessed 16 October 2016].

Android Developers. Android studio. *developer. android. com*, 2015.

Android Developers. Reducing Network Battery Drain. Online, 2016. URL https://developer.android.com/training/performance/battery/network/index.html. [Accessed 12 October 2016].

Jared Dickson. Xamarin mobile development. 2013.

Brady Donnelly. How to Build a Mobile App That Won't Drain Your Battery. Online, 2012. URL http://mashable.com/2017/04/26/disconnect-with-this-phone/#7vAVhLpSeqqH. [Accessed 03 May 2017].

Ben Eligin. Google buys android for its mobile arsenal. *Bloomberg Businessweek*, News Analysis:121, 2005.

Richard Ferraro and Murat Aktihanoglu. *Location-aware applications*. Manning Publications Co., 2011.

Jan Fogl, Jan Dvorak, Ondrej Krejcar, and Kamil Kuca. Intelligent displaying of notes based on current position and time. In *Advanced Computer and Communication Engineering Technology*, pages 693–705. Springer, 2016.

How-To-Geek. Mobile Linux. Online, 2014. URL https://www.verizonwireless.com/support/android-os-faqs/. [Accessed 17 October 2016].

JavatPoint.com. Android Activity Lifecycle. Online, 2013. URL https://www.javatpoint.com/images/androidimages/Android-Activity-Lifecycle.png. [Accessed 03 May 2017].

Steven Martain. Upcoming name change for windows azure, Mar 2014. URL https://azure.microsoft.com/en-us/blog/upcoming-name-change-for-windows-azure/.

Cade Metz. Firebase Does For Applications What Dropbox Did For Doc's. Online, 2012. URL https://www.wired.com/2012/04/firebase/. [Accessed 19 October 2016].

Thomas Myer. *Beginning PhoneGap*. John Wiley & Sons, 2011.

Tutorials Point. SDLC V-Model model description. Online, 2015. URL https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm. [Accessed 15 October 2015].

Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: an overview. In *IEEE International Conference on Cloud Computing*, pages 626–631. Springer, 2009.

P.J. Rankin and J.C. Griffiths. Distributed location based service system, April 12 2005. URL https://www.google.com/patents/US6879838. US Patent 6,879,838.

Qun Ren and Margaret H. Dunham. Using semantic caching to manage location dependent data in mobile computing. *Proceedings of the 6th annual international conference on Mobile computing and networking - MobiCom '00*, 2000. doi: 10.1145/345910.345948.

Ali H Sayed, Alireza Tarighat, and Nima Khajehnouri. Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. *IEEE signal processing magazine*, 22(4):24–40, 2005.

Android Studio. The official ide for android. *Android Studio URL: https://developer. android. com/studio/index.html*, 2016.

Kaiyu Wan and Vasu Alagar. Context-aware mobile tour guide. *2006 International Multi-Conference on Computing in the Global Information Technology - (ICCGI'06)*, 2006. doi: 10.1109/iccgi.2006.20.

Daniel Williams. *Corona SDK Application Design*. Packt Publishing Ltd, 2013.

Mark Wilson. HTC Dream Android Phone. Online, 2008. URL http://gizmodo.com/5053264/t-mobile-g1-full-details-of-the-htc-dream-android-phone.html. [Accessed 17 October 2016].

Verizon Wireless. Android Operating System. Online, 2016. URL https://www.verizonwireless.com/support/android-os-faqs/. [Accessed 17 October 2016].