University of Portsmouth

School of Creative Technologies

Master of Science project undertaken in partial fulfillment of the requirements for the MSc Degree in Mobile Media Applications

**Conan Doyle’s Portsmouth Tour Guide**

By

**Martha Katsiki, UP789041**

Supervisor: Nipan Maniar

Project Unit: CT7MACTE

September 2016

Project type: **Artefact**

**Abstract**

The aim of this project is to develop a tour guide application on behalf of the Portsmouth City Council in order to create an interactive map of Portsmouth tracing Doyle’s movements in the city. A briefly introductions discusses the main aims of the project, its constraints as well as the time schedule of the development. The literature review will look at the different operating systems for mobile devices, their advantages and disadvantages before choosing the best of them, as well as the tools and technologies which will be used in the implementation of the application. Moreover, the different Software Development Life Cycle (SDLC) methodologies were analyzed and compared in order to choose the one which is the appropriate for this project. The planning of how the design process should proceed is discussed while diagrams and ER models were used as well. Goals and requirements were created to ensure that the result would be successful. The implementation of the User Interface is analyzed including the difficulties which occurred and the cooperation of the Portsmouth City Council both in the User Interface implementation as well as in the requirements’ phase. The application is going to be developed by using Android Studio and, thus, programming in Java. Each phase had its own problems which arose during the process but these problems were examined thoroughly and solved. The development was tested in order to ensure that all the application’s functionalities work as expected and to discover any possible errors or bugs. There were two types of testing; the application was tested with the use of Android Studio’s emulator but also it was tested in a real device to ensure that the implementation is successful. The report concludes with recommendations for future changes as well as the author’s reflections to the general outcome of the project and its process.

Table of Contents

[1. Introduction 4](#_Toc461130281)

[1.1. Aim and Objectives 4](#_Toc461130282)

[1.2. Constraints 4](#_Toc461130283)

[1.3. Planning 4](#_Toc461130284)

[2. Literature Review 5](#_Toc461130285)

[2.1. Mobile Platforms 5](#_Toc461130286)

[2.1.1. Android 5](#_Toc461130287)

[2.1.2. iOS 5](#_Toc461130288)

[2.1.3. Windows Phone 6](#_Toc461130289)

[2.1.4. BlackBerry OS 6](#_Toc461130290)

[2.2. Comparison of Mobile Platforms 6](#_Toc461130291)

[2.2.1. Conclusion 7](#_Toc461130292)

[2.3. SQLite 7](#_Toc461130293)

[2.3.1. SQLite Manager 7](#_Toc461130294)

[3. Methodology 8](#_Toc461130295)

[3.1. Waterfall Model 8](#_Toc461130296)

[3.2. Spiral Model 8](#_Toc461130297)

[3.3. Iterative/Incremental Model 8](#_Toc461130298)

[3.4. Comparative Study of SDCL models 9](#_Toc461130299)

[3.5. Conclusion 10](#_Toc461130300)

[4. Design 11](#_Toc461130301)

[4.1. Requirements 11](#_Toc461130302)

[4.2. Design Process 12](#_Toc461130303)

[4.3. User Interface 15](#_Toc461130304)

[4.4. Conclusion 21](#_Toc461130305)

[5. Implementation & Testing 21](#_Toc461130306)

[5.1. Development Tools 21](#_Toc461130307)

[5.2. Development Method 21](#_Toc461130308)

[5.3. Resources 21](#_Toc461130309)

[5.4. Data Storage 22](#_Toc461130310)

[5.5. Photos Copyrights 22](#_Toc461130311)

[5.6. Splash Screen - Main Activity Class 22](#_Toc461130312)

[5.7. Maps Activity Class 23](#_Toc461130313)

[5.8. Main Menu Class 23](#_Toc461130314)

[5.9. Info Details Class 24](#_Toc461130315)

[5.10. Data Base Helper Class 24](#_Toc461130316)

[6. Recommendations 25](#_Toc461130317)

[7. Conclusion 25](#_Toc461130318)

[8. References 26](#_Toc461130319)

[9. Appendices 27](#_Toc461130320)

[Appendix A 31](#_Toc461130321)

[Appendix B 34](#_Toc461130322)

[Appendix C 36](#_Toc461130323)

[Appendix D 46](#_Toc461130324)

[Appendix E 49](#_Toc461130325)

[Appendix F 54](#_Toc461130326)

[Appendix G 58](#_Toc461130327)

**Table of Figures**

Figure 1: Time schedule of the process 4

Figure 2: User Interface Diagram 13

Figure 3: Class Diagram 14

Figure 4: The flow of the application 15

Figure 5: Splash Screen 16

Figure 6: Map Layout 17

Figure 7: Slider Menu 18

Figure 8: Result of a menu category 19

Figure 9: Information layout without including photos 20

Figure 10: Information layout including photos 20

Figure 11: Worldwide Smartphone OS Market Share. (IDC, n.d.) 27

Figure 12: Mobile devices preferred by app developers worldwide. (Statista, 2012) 28

Figure 13: Top 8 Mobile Operating Systems. (Asokan, 2013) 28

Figure 14: Waterfall Model (Sparsha, n.d.) 29

Figure 15: Spiral Model (Sparsha, n.d.) 29

Figure 16: Iterative Model (Sparsha, n.d.) 30

Figure 17: Comparison of SDLC models (Fulton, Aziz, 2015, p. 106-111) 30

# Introduction

## Aim and Objectives

This paper will analyze the development of an Android application. It will explore each stage of the process, research, design, implementation and testing. Portsmouth City Council is the client of this artefact. It is home to Richard Lancelyn Green’s largest public collection on all things relating to Conan Doyle and Sherlock Holmes. The client requires a mobile application with simple interface with areas of interest, navigation and mechanism for the public to locate them. The proposed application aims to solve this problem.

It will be an interactive map of Portsmouth tracing Doyle's movements in the city. Sir Arthur Conan Doyle lived and worked in Portsmouth for eight years. This project is to create a mobile application that presents a guided tour to sites of interest around Portsmouth to visit locations related to Conan Doyle and other notable events. Moreover, it will include photos of the locations from that eight-year period, as well as the map of Portsmouth and will also include information regarding each place. This mobile application could guide ‘players’ around the city but the application could use a combination of media. (see Appendix F,G)

## Constraints

The logo, photos and icons will be provided by the Portsmouth City Council, therefore their permission is important in order to use them or edit them. In addition, the client asked that the photos which will be shown must be protected so users cannot download them in their phones since the City Council owns the copyrights. As a result, they should be protected either with the use of watermarks or by disabling the ability to be downloaded in the user’s phone.

## Planning

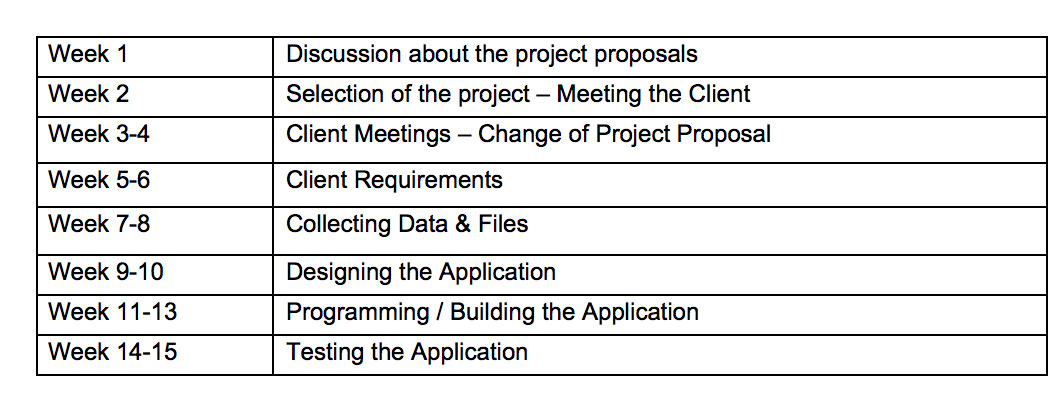


Figure : Time schedule of the process

At the beginning, an initial search will be conducted to identify existing examples for Android applications that work as tour guides. Secondly, a research will follow concerning the technologies that have been used in the app’s development such as database. The tools which will be used are Android Studio, SQLite Manager and Adobe tools. In order to develop this application, the skills needed are programming in Java and use of MySQL. The design of the app will include the map layout with a slider menu, a splash screen and a page/layout with the information for each location. Then the database will be built which will include the data and locations and will be used in the map and info page.

# Literature Review

## Mobile Platforms

In this article (IDC Research, n.d.), it states that the international smartphone market has been expanded to 13.0 per cent over the last three years, according to the research from the International Data Corporation (IDC). Android has the highest share in the market in 2015 (82.8 %), followed by iOS with a share of 22.3 per cent. Windows Phone has a share of 2.6 per cent and the operating system ranked last is the Blackberry OS, which hasn’t increased and has remained at (0.3%) (see Figure 11).

### Android

Android is an operating system designed for mobile devices such as smartphones and tablets. Developed by Google and the Open Handset Alliance, it is based on the Linux kernel and grants an SKD for developing applications using Java. Android is an open source platform, contrary to iOS (Patil, Ramteke, Chaudhari, 2014, pp.3906-3910).

It multitasks; using different applications running simultaneously. It offers a big external storage. Some of services provide are Bluetooth, EDGE, 3G, WiFi support, camera, GPS and GSM Telephony support. In addition, it supports media such as audio, video and images (Asokan, 2013, pp. 377-382).

It also provides development tools making faster to implement applications. Developers can also use all services and data indiscriminatel (Jones, n.d.). It supports the use of SQLite for data storage as well (Asokan, 2013, pp. 377-382).

### iOS

iOS is a strong operating system designed by Apple for the iPhone but also used in iPad, iPhone and iPod. It offers encrypted network communication which is used to secure personal data during transmission, when data are stored on the device or when backed up to iTunes (Jyothy, Shinto, 2013, pp.10-19).

Its interface stands out in the market, the design fits well with the Apple devices. Its security is considered high standard and the bugs are limited in the development phase or in updates. It offers easy entry to the Apple store and highly supports the brand-new web standards and cloud storage (Asokan, 2013, pp. 377-382).

### Windows Phone

Windows Phone is a mobile operating system developed by Windows for smartphones. It supports many of Windows’ features such as multi core processor, great storage support and screen resolution. It can nearly resemble the computer version of Windows. (Jyothy, Shinto, 2013, pp.10-19)

Recent version Windows 10 mobile released in June, 2015, aimed to decrease fragmentation in the platform. Moreover, it will be able to keep receiving updates to its features and functionalities. This version has been praised but also criticized for the privacy issues; concerns have been expressed by critics and advocates. Critics indicated the primal release as hasty, doubting the stability of the operating system. (Lozada, n.d.)

### BlackBerry OS

The BlackBerry OS is developed by Research in Motion (RIM) and used entirely in BlackBerry devices. Its network architecture is not similar to the other operating systems’ architecture and it provides two encryption options. (Jyothy, Shinto, 2013, pp.10-19)

The reason of its development was for business, therefore it provides them with highly secure devices and sufficient communication technologies. Thus, it addresses the enterprise market as well as the Security market. As a result, its growth really depends on government and enterprise contracts. The fact that they are only a few applications available, compared with the other operating systems, and the substandard existence in the tablet market makes it difficult to advertise the brand, considering the competition in the constantly growing smartphone market. (Em, n.d.)

## Comparison of Mobile Platforms

Comparing the loading time among the mobile platforms, it is shown that iOS is the fastest; it is also the most popular on the internet in contrast with android phones. Android notifies the user of any incidents might appear without interrupting his tasks. On the other hand, iOS informs the user but his activities are interrupted. (Sharma, Mahender, Sharma, 2013, pp. 1-5)

Blackberry phone offers a quick and easy navigation thanks to the keyboard shortcuts. The GPS in blackberry phones uses google maps like Android and supports an easy way of communication, while the phone’s quality is considered to be better than the iPhone’s quality (Sharma, Mahender, Sharma, 2013, pp. 1-5).

Windows phone offers the possibility of sharing data among applications and/or communicating with each other with the provision of contracts. It can perform background tasks which are affected by the internet, network or location services. Users can monitor the state of the battery and deactivate the background tasks. The Voice Command has been upgraded and it allows users to perform an advanced voice search using commands which are not obligated to follow a specific vocabulary (Muller, n.d.).

Developing applications and maintain them it is easier in iOS, since Apple is the only manufacturer who owns iOS devices. On the contrary, there is a variety of devices from different companies that use Android as an operating system, therefore developers have to take into consideration that the applications they develop must be compatible with all these devices (Sharma, Mahender, Sharma, 2013, pp. 1-5). This might be the reason why the majority of developers prefers to develop applications in the iOS environment (see Figure 12) (Statista, 2012).

### Conclusion

iOS is chosen by developers, while Android is preferred by users (see Figure 13). However, it is the client who will make that decision. After presenting this information to Portsmouth City Council, they made their decision and chose the Android Platform, taking into account the popularity of this mobile operating system. Consequently, this application will be developed in Android, thus the next section will mention the Android’s features that will be used in the development.

## SQLite

SQLite is a library that implements a SQL database engine. It is for use for any purpose either private or commercial (Jan, Shafi, 2015, pp.9-15). Depending on the target platform and the included features, the size can be less than 500 KB (Vogel, 2015).

Developed by Richard Hipp in 2000, it is used in numerous browsers, operating systems and embedded systems. SQLite is the most popular open source embedded database thanks to its simple use compare to other databases developed by IBM, Microsoft, Sybase or Oracle Corporation (Jan, Shafi, 2015, pp.9-15). When used in Android platform, it does not require any kind of installation or setup. Although, the process of creating and updating the database is essential. (Vogel, 2015)

SQLite is autonomous, it does not need backing from Android or from outer libraries hence it is applicable for various devices with different formation. It is a server-less database meaning that is does not require administration; developers can directly read and write data from the database. All the contents are store in an individual cross-platform file (Jan, Shafi, 2015, pp.9-15).

### SQLite Manager

The data needed for the application have been provided by the client. It is information about all the locations which are relevant to Sir Arthur Conan Doyle while he lived in Portsmouth, including coordinates, date, address. For this reason, there is no need for SQLite to use attributes in order to write in the database. However, the database must be created and then insert the data. This will be achieved with the use of a tool called SQLite Manager, a free Firefox Extension to manage any SQLite database on the computer. The implemented database will later be added to the Android application. Using the right SQLite elements in Android, the data will be read and used for the implementation of this application.

SQLite Manager permits the use of sqlite2 and sqlite3 and the ability to convert a database from sqlite2 to sqlite3 format. Developers can create, edit and update tables or data, but are also able to export and import databases. In addition, SQLite Manager offers several functions which can be found by navigating through the menus. When registered functionalities are provided without restrictions, however the free version provides the basic services (Rachman, n.d.).

# Methodology

The step, before starting designing and programming the application, will be to find the right methodology for the development. While researching, the Software Development Life Cycle (SDLC) was found. SDLC is a methodology used in software products. There is a number of successful SDLC models which seem to have the same structure: a sequel of tasks which must be completed. This chapter will describe the three most common SDLC models and present the one chosen for the implementation (Fulton, Aziz, 2015, pp. 106-111).

## Waterfall Model

This model has a sequel of phases; these are analysis, design, implementation, testing and maintenance (see Figure 14). Each step must be completed before moving on to the next one. In the first step, the developer team gathers requirements, examines them and presents them to the client in order to make an agreement about the final project and its content. Information from the previous step have been evaluated and the team proceeds to the next step: finding the right strategy to solve the problem using different features such as diagrams or algorithms. When completing the plan, the coding part starts in order to implement the requested software product. Then, the product will be tested in case any bugs may be found or confirm the final solution fulfil the client’s requirements (Fulton, Aziz, 2015, pp. 106-111).

## Spiral Model

The Spiral Model concentrates on risk assessment and the reduction of risks; risks can be analyzed and a decision made as to whether the project should be continued during the process. After collecting the requirements, developers examine each step for the requirements and gather information about potential risks. If new requirements appear, they will repeat the process until it is time for installation and maintenance. These repetitions are called prototypes; Spiral model mix both elements from design and from these prototypes (Fulton, Aziz, 2015, pp. 106-111) (see Figure 15).

## Iterative/Incremental Model

This model uses the waterfall phases in repetition creating a “multi-waterfall” cycle. With the end of a development cycle, a new functionality has been implemented (Ramoshaba, n.d.). Therefore, the work has been divided in smaller pieces. This method will be applied until all the functionalities will be developed. (Fulton, Aziz, 2015, p. 106-111) (see Figure 16)

The iterative/incremental model is best suited for a project whose requirements must be collected and examined from the beginning (before the iterations start) and when the project must be delivered in a short time (Ramoshaba, n.d.).

## Comparative Study of SDCL models

Waterfall Model:

Advantages:

* Easily understandable, manageable and operative.
* Extensively used.
* Sequential model, easily developed.
* Helps amateur developers by offering design.
* Each step presents its results which will be examined and documented before moving to the next step.

Disadvantages:

* Requirements must be defined in advance.
* Strict structure.
* Each phase must be processed in detail before proceeding in the next one since it is really difficult to return to the previous stage and fix errors.
* The final product is ready at the last phase and cannot be examined earlier.
* The risks are high in case the client cannot explain with clarity his requirements.
* It is not suitable for complicated and object-oriented projects.

The Waterfall Model is better used in cases when the best quality is need regardless of the time or expenses, when the requirements are indubitable, processed and know from the start or when a product already exists and either a new version needs to be implemented or it has to be ported in another environment (Fulton, Aziz, 2015, p. 106-111) (Ramoshaba, n.d.) (see Figure 14).

Spiral Model:

Advantages:

* Very good evaluation of risks.
* The team can start developing the program from the beginning so the client can examine it from the early phases.
* New functionalities can be submitted in every loop.
* Development is broken down into smaller pieces so the parts that include risk can be fixed early in the coding phase, accommodating the risk management.

Disadvantages:

* Complicated management of risks, while experience in it is essential.
* It has high expenses especially for small projects.
* Time cannot be defined accurately when dealing with risks.
* Each loop of the process might be immoderate.
* The process of analyzing risks defines the success of the final result.

The Spiral model is used for projects with a high rate of risks and await important changes when the risks’ estimation is significant. It can also be used when requirements cannot be clear from the start (Fulton, Aziz, 2015, p. 106-111) (Ramoshaba, n.d.) (Sparsha, n.d.) (see Figure 15).

Iteration Model:

Advantages:

* The developing parts which include great risks or important functionalities can be implemented early. In this case the client can review any new functionality added to the project and give feedback.
* Since the development has been divided into smaller pieces, the risks are also separated.
* At the end of every iteration, a product is given and the result will be documented and reviewed so if the team finds errors they can fix them.
* Its method minimizes the probability of lack of success and the alteration of the requirements.
* Decreases the primal expenses.

Disadvantages:

* Demands good design and preparations.
* Demands early definition of requirements in order to proceed to the development; it is not suitable for changes.
* Demands a lot of resources.
* Exact schedule may not be defined clearly.
* Risk analysis is a very important part of the project.

Iteration Model is suitable for projects whose risks are not high or if there is need to deliver a functional product early. Moreover, it can be used when the development phase seems to be extensive or there is the risk to implement the project all at once instead of diving it. When new technologies are used and the development must be divided into smaller parts in order to help the user adapt to the system (Ramoshaba, n.d.) (Sparsha, n.d.) (Fulton, Aziz, 2015, pp. 106-111) (see Figure 16).

## Conclusion

After reviewing the pros and cons of these SDLC models, the next step is to decide which is more suitable for the development of the application. First of all, it is essential to analyze what features apply in this application:

* The requirements are defined by the client with regular meetings so there is a possibility of change.
* The application must be delivered in a short period of time (3 months).
* The client must review the project and be regularly informed about the progress of the implementation.
* The data needed for the development will not be given from the start, therefore when the application must be tested after a new entry.
* Considering that requirements and data are not known from the beginning or there is a possibility of change, flexibility is important.

After taking into consideration the needs of the application and the features of each model (see Figure 17), it was decided that the most appropriate model is the Incremental/Iterative model. It is flexible, client can be highly involved, it eases the implementation and the testing and it is appropriate for requirements whose definition might be altered during the process.

# Design

## Requirements

This section explains the process of the application’s design. As mentioned above, this application will be developed for a client, therefore the client must express his opinion before making a decision about the design. First of all, client required an easy to use interface, compatible with all smart phones’ screen sizes which use Android as an operating system.

In the previous chapter, the SDLC model were explained and compared leading to the final choice the Iteration/Incremental model (see Figure 16). Later, the design, the development, the testing and the implementation follow. The process is repeated until the product is ready. According to the Iteration model, the requirements must be gathered from the beginning. In addition, client suggested the same because he knew he will be absent in the last part of the development phase. During the meetings the client explained what he really requires concerning the layouts, the functionality and the data of the application. He also clarified that the data, the icons and the logo will be provided by the Portsmouth City Council. The final requirements will be displayed bellow:

* Simple to use. Few activities.
* The application will contain a map with all the locations included.
* The map layout will include a slider menu as well. This menu will categorize the locations according to the tags provided by the client. These are: Sherlock Holmes, Conan Doyle, Doctor, Writer, Family, Spiritualism, Sport, Literary and Scientific Society, Portsmouth, Southsea, Friends and Politics.
* When a category is chosen, the map will display only the locations of the specific tag.
* When a marker is clicked, a popup window will show up asking the user if he wishes to see more info for the specific location. If yes, a new activity will open.
* The new layout contains information for that place such as address, date, information, additional information, photos (if there are available) and informs the user if the building still exists.
* It must be an Android application and programmed in Java.
* It must support the use of database.
* It should be free from as many bugs as possible.
* Data will be retrieved from the database.
* Database and photos will be stored on a server.

Since the analysis of the requirements of the application has been completed, the User Interface will be designed to confirm that the needed functionalities will be implemented in a simple layout. It was decided that each functionality will have its own layout.

These layouts are:

* The Slash screen layout: The client requires a Slash screen with the logo and the name of the application. It will be the first thing the user will see when he opens the application.
* Map layout with a slide menu: The map will show all the locations in Portsmouth while the slide menu will separate the locations according to their tags, which have been provided by the client, and will show these locations on the map.
* Info layout: When a marker is clicked on the map, a pop-up window will ask the user if he wishes to see further info for the specific location. If yes, the Info layout will open providing all the information available for this location.

## Design Process

Representing the requirements graphically benefits with a better understanding of the functionality (Kappel, Proll, Reich, 2006, p.43). Kappel et al. (2006, p.41) claim that Unified Modeling Language (UML) can be used for specification of the functional requirements; it uses diagrams and use cases for modeling.

Using the online program (Gliffy, Inc.), the following diagrams were created in order to have a better understanding of the design process. The next diagram describes functional requirements by identifying user’s “activities” in the application. The user can see on the map all the locations related to Sir Arthur Conan Doyle. He can also choose a category from the menu in order to display only the locations of this category. If he wants to get more information about a specific location, he can click on the marker and details will be displayed in a new layout.

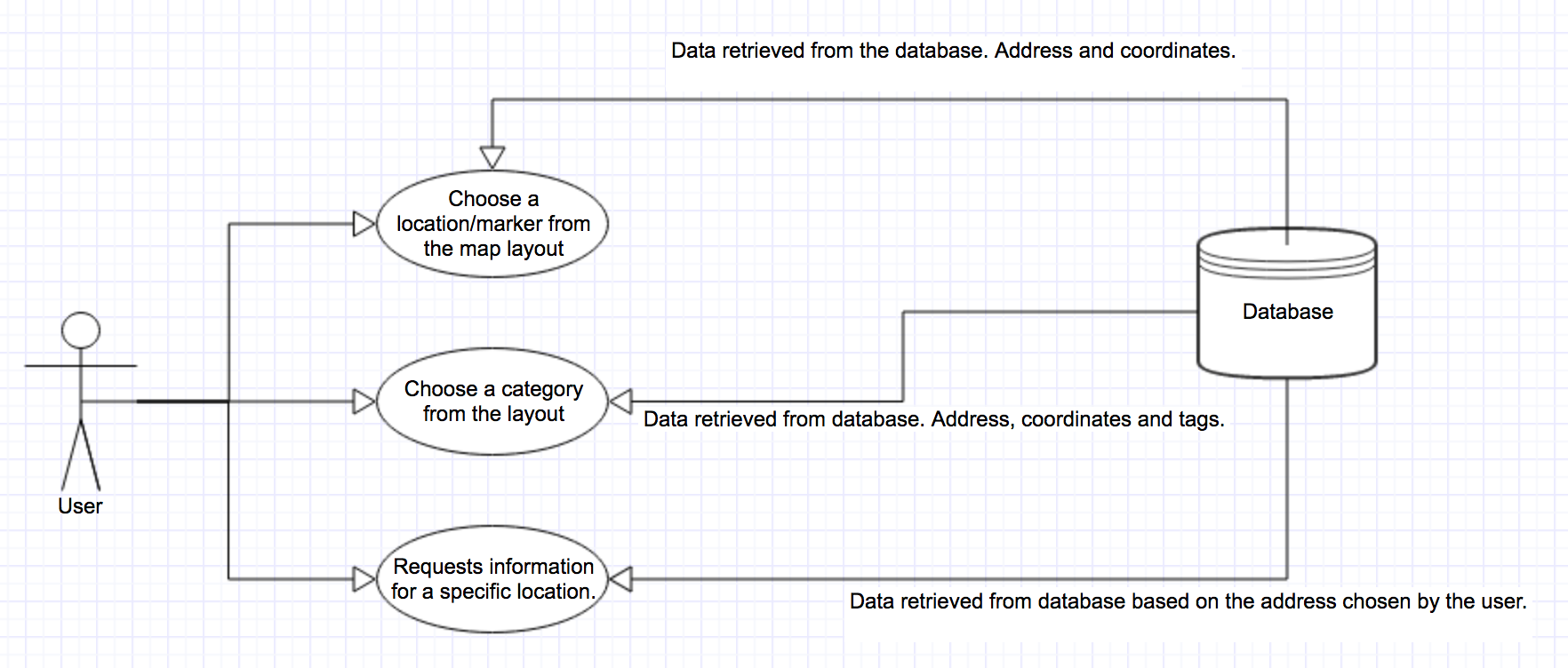


Figure : User Interface Diagram

The previous diagram was focused on the level of User Interface; it involves services which are displayed on the layout. The following diagram describes the process of searching for information about a location, chosen by the user, by using a UML class diagram (Kapper et al., 2006, p.42-44). The design can be split into two main parts. The presentation, which is basically the user interface, and the development which includes all the work happening on the background. (Hammer, 2004, p.2). Therefore, diagrams can help to understand both sides in order to achieve a functional application.

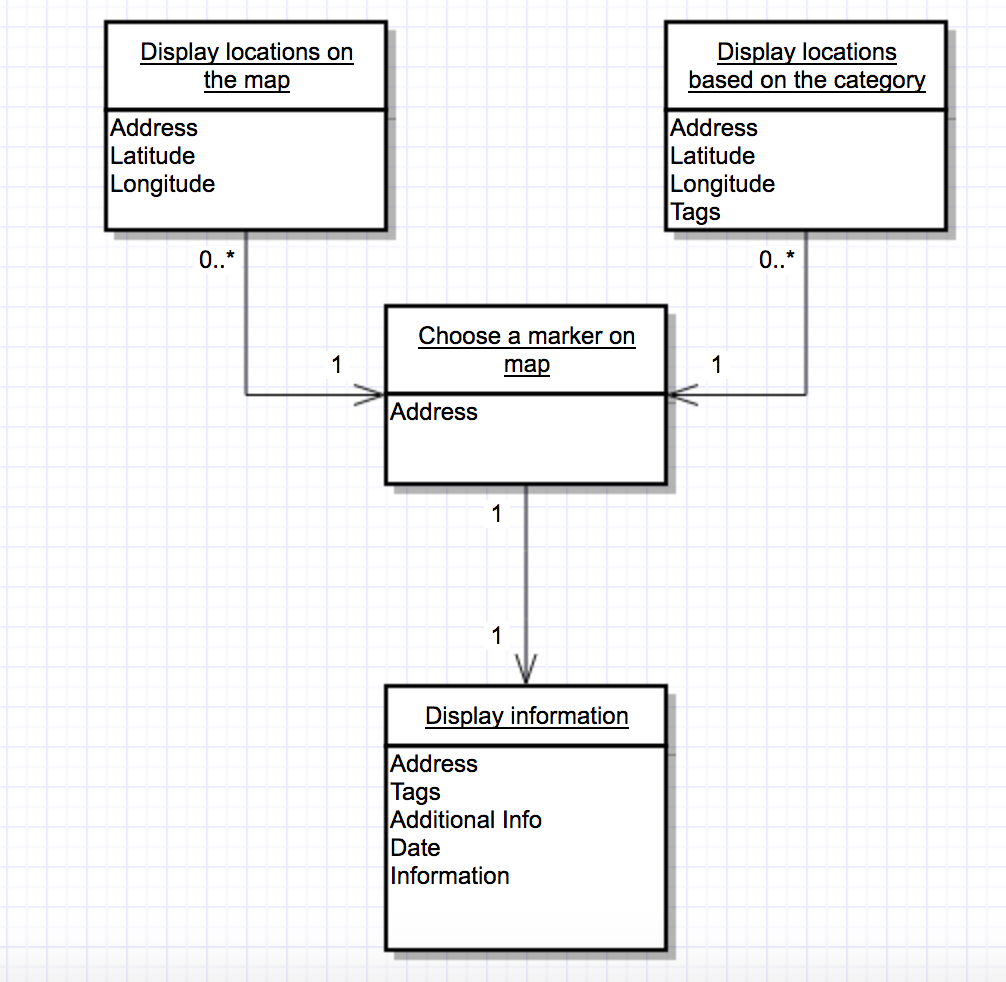


Figure : Class Diagram

The diagram on the next page indicates the flow of the application from a top-level view. Its purpose is to be an overview of the main process of the application. In addition, it could help understand how the layouts and the functionalities are connected together.



Figure : The flow of the application

The next section will focus on the layouts’ design and its process.

## User Interface

This part describes the process of the layouts’ creation. These layouts will be presented to the user while he uses the application’s service. The goal is to create of an easy, simple interface in order to be compatible with all the Android devices. Thus, it is important to create a general user interface which can be designed easily and fast since the development must be completed in a short time.

The user interface is designed for a screen of 1080 pixels wide by 1920 pixels high; these are the dimensions of Nexus 5. However, it must be compatible with devices which have different dimensions so the application was tested in the emulator by using the devices included in the Android Studio. The main screens of the interface are:

* Splash Screen: the first screen displayed on the user. The client designed and provided the image.
* Map layout: Locations are displayed. The layout also includes a slider menu with categories for the locations.
* Information layout: All the information is displayed for a specific location.

During the meetings, the results of the initial research were presented to the client. This research was focused on finding existing applications similar to the concept of this project. The client examined both the presentation and the functionalities of these applications. His decisions were mentioned in the requirements.

Each screen will be displayed below and described how it was built and what features are included:

Splash Screen:



Figure : Splash Screen

The layout of this class (Splash Screen) was very easy to develop. It consists of a picture which covers the whole screen. Splash Screen is the application’s first screen; it is used as quick information screens which disappear after a short time.

Map Activity:

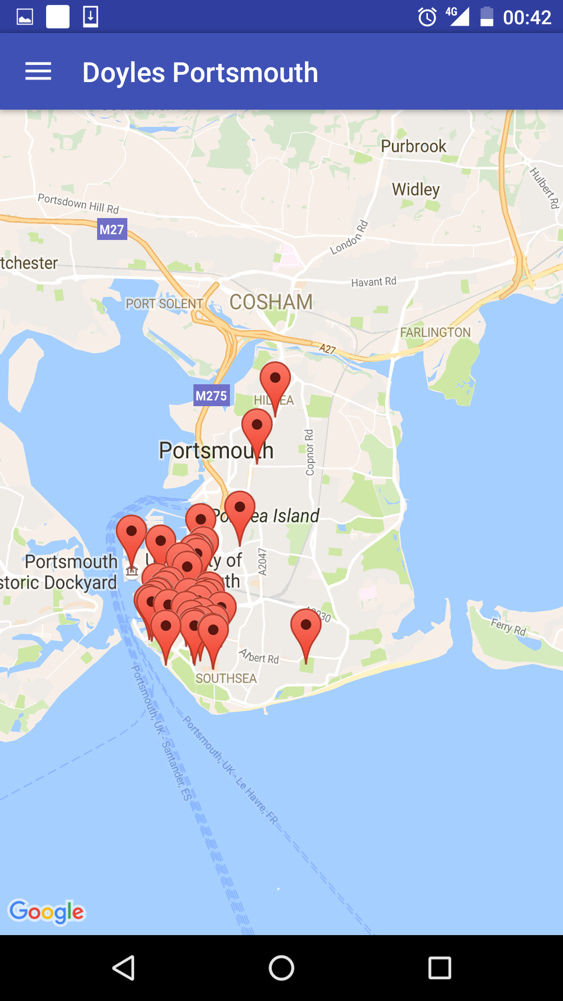


Figure : Map Layout

This activity uses the Maps Activity layout retrieved from the Android Studio. The markers are created by using the coordinates from the database. As requested from the client, a slider menu was added to the layout.

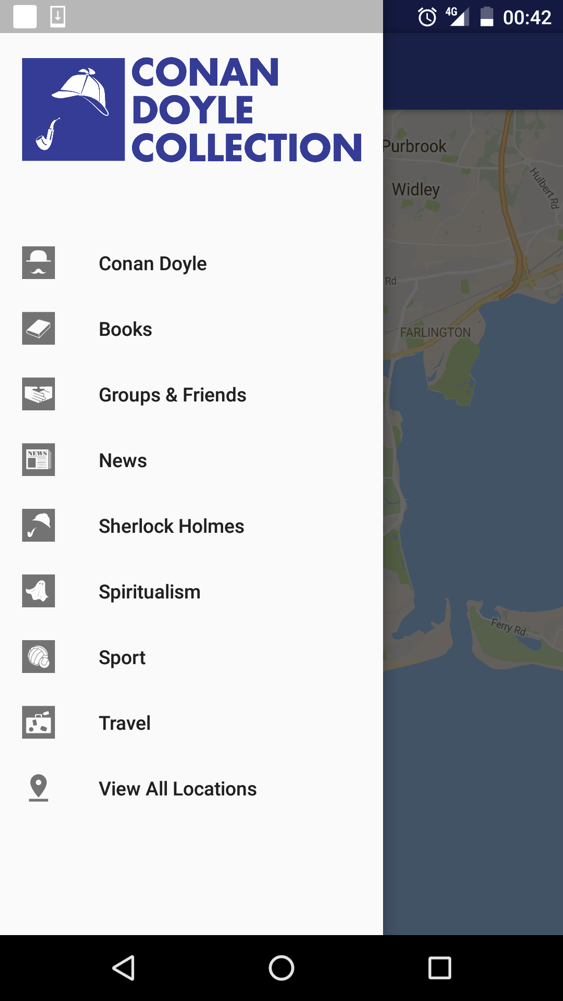


Figure : Slider Menu

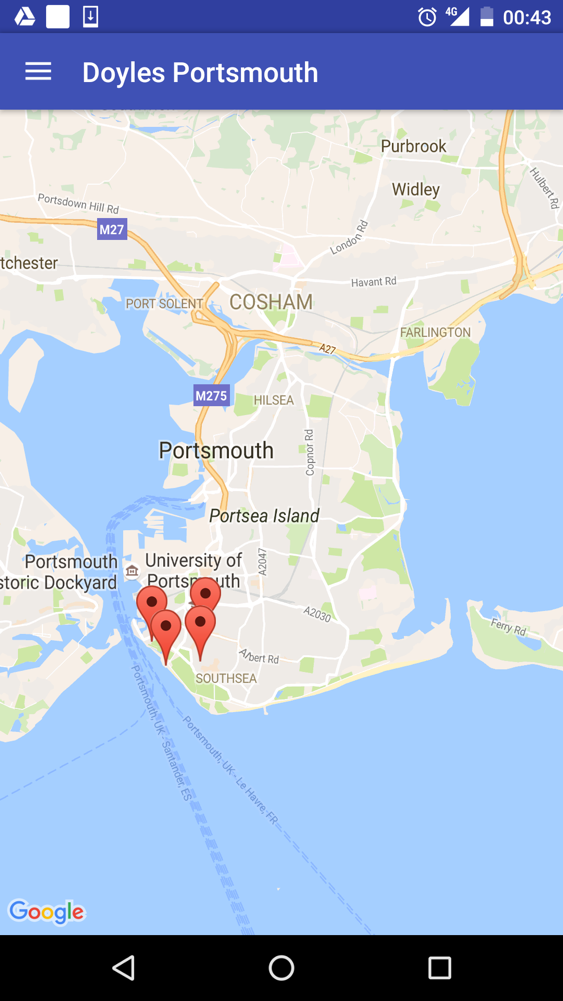


Figure : Result of a menu category

The menu contains categories based on the tags, which have been retrieved from the database, and display the locations which belong to the category/tag chosen by the user. Therefore, the user can accommodate the search.

If a marker is clicked, an alert window will pop up. This window asks the user if he wishes to get more information for the specific location. If he refuses, then the window will close and the user can check the other locations on the map. Otherwise, a new activity will open.

Info Details:

The new activity is called Info Details and includes all the information for the location which have been retrieved from the database. So far, the client has provided a few photos of places which are displayed in this activity’s layout. As a result, the layout is not the same for every location, since there are two possible results depending on the existence of the photos.

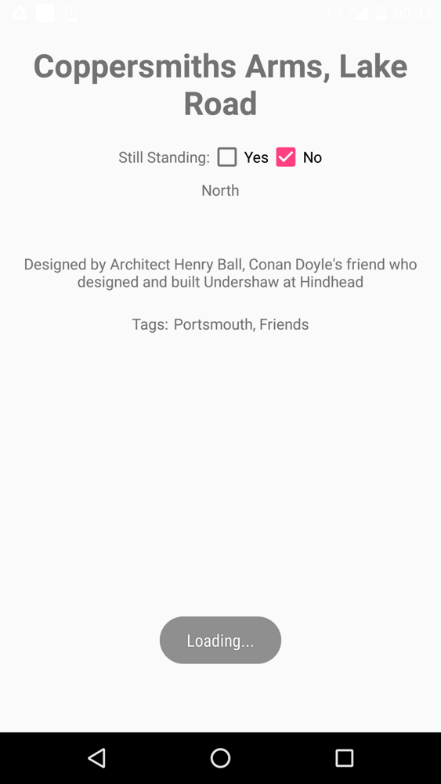


Figure : Information layout without including photos



Figure : Information layout including photos

The above layouts are the result of the initial ideas and the client’s requirements. Further development might include adding new features or improving the design. Appendix A includes a detailed description of the layouts’ implementation.

## Conclusion

This chapter presented an overview of the design process which is considered to be one of the important parts of the project’s implementation. The structure of the application has been discussed as well as the requirements and the design of the User Interface before moving on to the next chapter which explains the implementation process of the project.

# Implementation & Testing

The goal of this chapter is to provide an overview of the implementation of this artefact. This section will focus on the development and describe in detail the process as well as the testing phase, explaining what difficulties occurred while coding and the final result.

## Development Tools

The selection of development tools was based on the software which is available in the market as well as its popularity and functionality. Although, a reference to the tools has been made in the *Introduction*, it is worth mentioning that these tools are open-source projects and most of them were free of charge but there were also few that were installed in the university’s facilities. Those who were provided by the University are the Adobe tools. The rest of them were found online, these are Android Studio and SQLite Manager. These tools have online communities or forums which can provide support.

## Development Method

The aim is to implement precipitately each activity and test during the process. Any bugs or errors were found and fixed. The development will include activities and its components as well as the code. The components and elements must work along with the methods which will be implemented by the developer. The project will be tested efficiently on the Android Studio’s emulator (Nexus 5, Nexus S, Nexus One, Galaxy Nexus, Nexus 7/9/10) and later to an actual device (Nexus 5X).

There were a few difficulties that were encountered throughout the development. These difficulties will be explained in the following sections.

## Resources

This section presents the resources which have been used and had an important part in the development of this project.

* Human resources: Project supervisor, lecturers, Portsmouth City Council – staff.
* Personal skills: programming in Java, use of Adobe tools, programming skills, information gather from internet and books, knowledge learnt at university, experience in implementing Android applications, use of Android Studio and its services or/and components.
* Hardware: a development laptop which contains tools and programs for development and testing as well as tools for editing icons and photos. A mobile device, Nexus 5X, used for testing the layout and the functionalities.
* Software: Android Studio for developing provided by Google Developers. Emulator provided by Android Studio for testing. The required software mentioned above is open-source, available online and easy accessible.
* Internet resources: This contains academic articles, e-books and game development websites. Needless to say that Android Developer website is the prime source information.

## Data Storage

The client required the use of server which will include a database and photos used in the layout. However, there was not enough time and the application should have been developed rapidly. In addition, when the project started, the client stated that the server of Portsmouth City Council was under maintenance. As a result, it could not be used for the implementation. For this reason, another database was created by using SQLite Manager. This database is local and it has been added in the Android Studio’s project as an asset. This is a temporary solution in order to be able to implement the artefact and present it to the client in a three-month period.

## Photos Copyrights

As mentioned earlier, the client provided photos from the Conan Doyle’s collection in order to be used in the application. These photos display locations in Portsmouth from the time Sir Arthur Conan Doyle lived in the area. Although, the client gave permission to use these photos in the project but they have to be protected since Portsmouth City Council owns the copyrights. The client asked either to disable the possibility of the “copy-paste” method or to add watermarks in the photos. With the use of Adobe Photoshop, watermarks were added in the photos (Patterson, n.d.). However, the watermarks, that were used, were created by the developer in order to proceed with the implementation of the application but Portsmouth City Council have their own to use which can be added later. The client did not send all the necessary photos but a small part of them, enough to complete this part of the development phase; the rest of them can be added later.

## Splash Screen - Main Activity Class

The Main Activity class was the first one to be implemented; its purpose was to create a Splash Screen visible for 5 seconds before the Main Menu class opens. Its implementation was easy and through the testing phase it was proved that it works properly. The code is displayed below:

package com.example.marthakat.sirarthurconandoyleinportsmouth;  
import android.content.Intent;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import java.util.Timer;  
import java.util.TimerTask;  
  
public class MainActivity extends AppCompatActivity {  
//This is a Splash Screen  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 int timeout = 5000; // makes the activity visible for 5 seconds  
  
 Timer timer = new Timer();  
  
 //when the time is over we move to the MainMenu activity  
 timer.schedule(new TimerTask() {  
 @Override  
 public void run() {  
 finish();  
 Intent homepage = new Intent(MainActivity.this, MainMenu.class);  
 startActivity(homepage);  
 }  
 },timeout);  
 }  
}

## Maps Activity Class

The Maps Activity class was originally created because the google map is an important part of the application. However, there were difficulties during the implementation. It was proved throughout the testing phase that both the activity and the layout of this class could not use the elements needed for the slider menu because of the class that was extended. Consequently, the Maps Activity class was not used in the development, but its layout was used as a part of the layout of the Main Menu class (see Appendix C). Appendix B contains the coding of this class.

## Main Menu Class

The Main Menu class contains the map layout and the slider menu. Data have been retrieved from the database in order to add the locations on the map as markers. The menu includes the categories which divide the locations; the categories are based on the tags from the database. During the testing, there were difficulties designing the menu’s layout (see Appendix A) but the implementation was easy since it was based on the coding of the map development. The important part of the coding in this class was the connection to the MySQL database and the recovery of data. However, there were some difficulties while writing the code. Due to the fact that two columns’ names were changed in the database, the application could not function properly. Null exceptions were caused because the change of the name could not be recognized by the system. At that moment, a rapid solution was found; the column names were removed from the MySQL query, the final code was:

Cursor cursor2 = db.rawQuery("SELECT Date, Information, Tags FROM portsmouth WHERE Location = '" + loc + "'", null);

Since the data from these two columns were essential to the process, their data were retrieved in a different way (see Appendix C). While the application was being tested, it was revealed that the database could not recognize changes and updates in general. It is important to mention that when the application was tested on the emulator, there were no problems with the updated data. However, when it was tested in the Nexus 5X device, the application “crashed”. Although, there were one case caused Exception both in the emulator and the device. That an input in the database, specifically one of the addresses was written in a way that the query could not recognize it (e.g. St Agatha**’s** church). These changes were essential for the display of the data. Then, after researching online about SQLite databases, a method was found in order to update the database. As a result, the functionalities work properly (SQLite, n.d.). The user can click on a marker and request to read more information for the specific location. For this service, the Alert Dialog element was used. The positive button opens the Info Details class and the negative button closes the Alert Dialog and the map is display on the screnn. Appendix C contains the coding of this class.

## Info Details Class

The Info Details class contains all the information for the selected location which was chosen by the user. The data were saved from the Main Menu class using the Intent element and the method putExtra (Main Menu class) and the method getStringExtra (Info Details class) since all the MySQL queries are implemented in the Main Menu class. The Info Details class sets the data as values to the elements. In addition, this class implements the method to display the photos available for a few locations. During the testing, it was proved that the functionalities work smoothly without errors, exceptions or bugs. Appendix D contains the coding of this class.

## Data Base Helper Class

The Data Base Helper class is used in order to create the database and install a connection when its methods are called in the other classes. The class contains the methods that open the database and close it; there is a method that check if the database already exists and if it does not (exist), then give the right commands. Finally, there is a method that copy the database from the “assets” folder into the system. Appendix E contains the coding of this class.

This chapter explained the phase of the application development and the testing phase. The next chapter will discuss what part of application should be modified in order to be implemented in a better way.

# Recommendations

During the one-month period, the application has been tested in order to ensure that all the components of the system work properly. It was proved that individual methods perform well what is requested of them. Generally, the application has been developed to an acceptable level for the purpose of this project. The design artefact (User Interface), analyzed in the Design chapter, has been achieved to a satisfactory level, even though it can still be improved.

Most of what has been examined in the requirements section in the Design chapter has been developed. However, in order to improve the generic use of the project and make it more attractive to user as a well-performed application, the data storage should be upgraded. So far, a local database is used as it has been explained in the previous chapter because of the limited time of the implementation and the fact that the client could not provide a functional server. Therefore, a database hosted in a server is a better choice. After the implementation has been completed and the database will be ready to use, the client will be responsible for uploading photos, adding or editing data. The application has already the required permissions in its Android Manifest file, in order to use the Internet since it was necessary for the use of google maps. As a result, the users will be able to read the new data and see the new photos without being forced to update the application into a new version. In addition, the photos will be protected when they will be displayed by using the appropriate watermarks instead of the current ones.

This recommendations, when developed, will give to the application a more professional image and will improve the user experience of the application.

# Conclusion

The primary aim of the project was to create an application that could be used on any Android device. The most of the requirements, outlined at the Design chapter, have been met. Their progress on whether or not they have been completely achieved, has been described in the previous chapter as well as any further development in the future.

The Design chapter does not only include the gathering of the requirements but also the planning and the process of the design and the implementation of the application. The design was created with the assistance of the client as well as his contribution in the process, since he has provided the icons, the logo and the photos used in the application’s design.

The development phase of the application was the most difficult and lengthy part of the whole application. However, the development is flexible enough to accommodate future changes if required. During the implementation, some methods were not easy to implement but through research and testing the difficulties have been overcome and all the methods have been successfully implemented. The testing phase was a crucial part of the development process since all the errors have been recognized during testing. This way they were able to be fixed and proceed with the implementation. The application was tested in the emulator which provide a number of Android devices as well as in a real device in order to ensure that the application works as expected.

In conclusion, the projected has reached its purpose in such a small time frame and produce a functional mobile tour guide compatible with Android mobile devices. Nevertheless, there is still the need of future modifications in order to improve the design and some of the existing functionalities.

# References

1. Asokan, M. (2013). *Android VS iOS – An Analysis*. International Journal of Computer Engineering and Technology, 4 (1), 377-382.
2. Em, E. (n.d.). *Marketing Research Blackberry*. Retrieved from: <http://www.academia.edu/8183458/Marketing_Research_Blackberry>
3. Fulton, I., A., Aziz, K., A. (2015). *A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model*. International Journal of Computer Issues, 12 (1), p. 106 -111
4. Google Developers. (n.d.). *Maps Android API*. Retrieved from: <https://developers.google.com/maps/documentation/android-api/start>
5. Hamer, C. (2004). *J2ME Games with MIDP2*. Berkeley: Apress.
6. IDC Research (n.d.). *Smartphone OS Market Share, 2015 Q2*. Retrieved from: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>
7. Jan, I., Shafi, J. (2015). *Sqlite – A Better Choice as an Embedded Database for Smart phones*. International Peer Reviewed Refereed Journal, 2 (6), p. 9-15
8. Jones, T. (n.d.). *Android Application Development – Has Completely Captured the Market*. Retrieved from: <http://www.academia.edu/3611493/Android_Application_Development_-Has_Completely_Captured_the_Market>
9. Jyothy, J., Shinto K. K. (2013). *Mobile OS- Comparative Study*. Journal of Engineering, Computers & Applied Sciences, 2 (10), p. 10-19.
10. Kappel, G., Proll, B., Reich, S. (2006). *Web Engineering: The Discipline of Systematic Development of Web Applications*. Heidelberg, Germany: Wiley.
11. Lozada, L.D. (n.d.) Windows 10. Retrieved from: <http://www.academia.edu/21568743/Windows_10>
12. Muller, S. (n.d.). *Windows Phone 8.1 – platform concepts and app development*. Retrieved from: <http://www.academia.edu/7712500/Windows_Phone_8.1_platform_concepts_and_app_development>
13. Patil, D. B., Ramteke, P.L., Chaudhari, D.N. (2014). *Development of Android Mobile Application for Cloud Server*. International Journal on Recent and Innovation Trends in Computing and Communication, 2 (12), 3906-3910.
14. Patterson, S. (n.d.). *Add A Copyright Watermark Pattern to A Photo with Photoshop*. PhotoshopEssentials. Retrieved from: <http://www.photoshopessentials.com/photo-effects/copyright/>
15. Saokosal, O. (2016). *Latest Android Studio Navigation Drawer Tutorial*. Retrieved from: <https://www.youtube.com/watch?v=H4R-hz56oBA>
16. Sharma., T.N., Mahender, Kr. B., Sharma, A. (2013). *Comparative Study of Different Mobile Operating Systems*. International Journal of Advancement in Research & Technology, 2 (3), p. 1-5.
17. Sparsha, A. (n.d.). *Software Development Life Cycle*. Retrieved from: <http://www.academia.edu/10759303/Software_Development_Life_Cycle_SDLC>
18. SQLite, (n.d.). SQL As Understood By SQLite. Retrieved from: <https://www.sqlite.org/lang_reindex.html>
19. Statista. (2012). *Mobile devices preferred by app developers worldwide*. Retrieved from: http://www.statista.com/statistics/256833/mobile-devices-preferred-by-app-developers-worldwide/
20. Rachman, D., A. (n.d.). *SQLite Manager*. Retrieved from: <http://www.academia.edu/5093125/SQlite_Manager>
21. Ramoshaba, T. (n.d.). *SDLC*. Retrieved from: <http://www.academia.edu/22446106/SDLC>
22. Vogel, L. (2013). *Android SQLite Database and Content Provider Tutorial*. Retrieved from: <http://www.academia.edu/4446368/Android_SQLite_Database_and_Content_Provider_Tutorial>

# Appendices

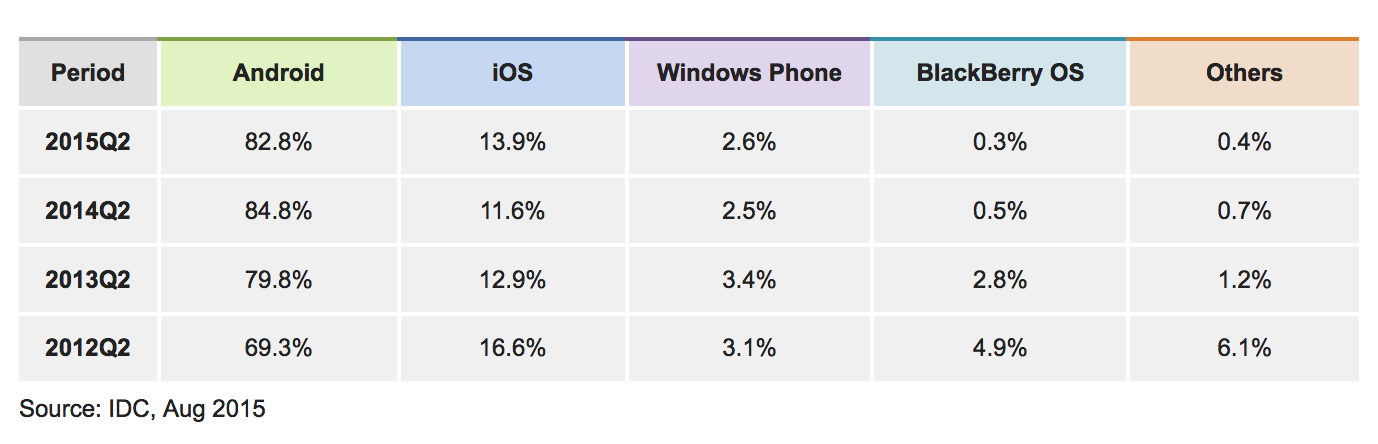


Figure : Worldwide Smartphone OS Market Share. (IDC, n.d.)

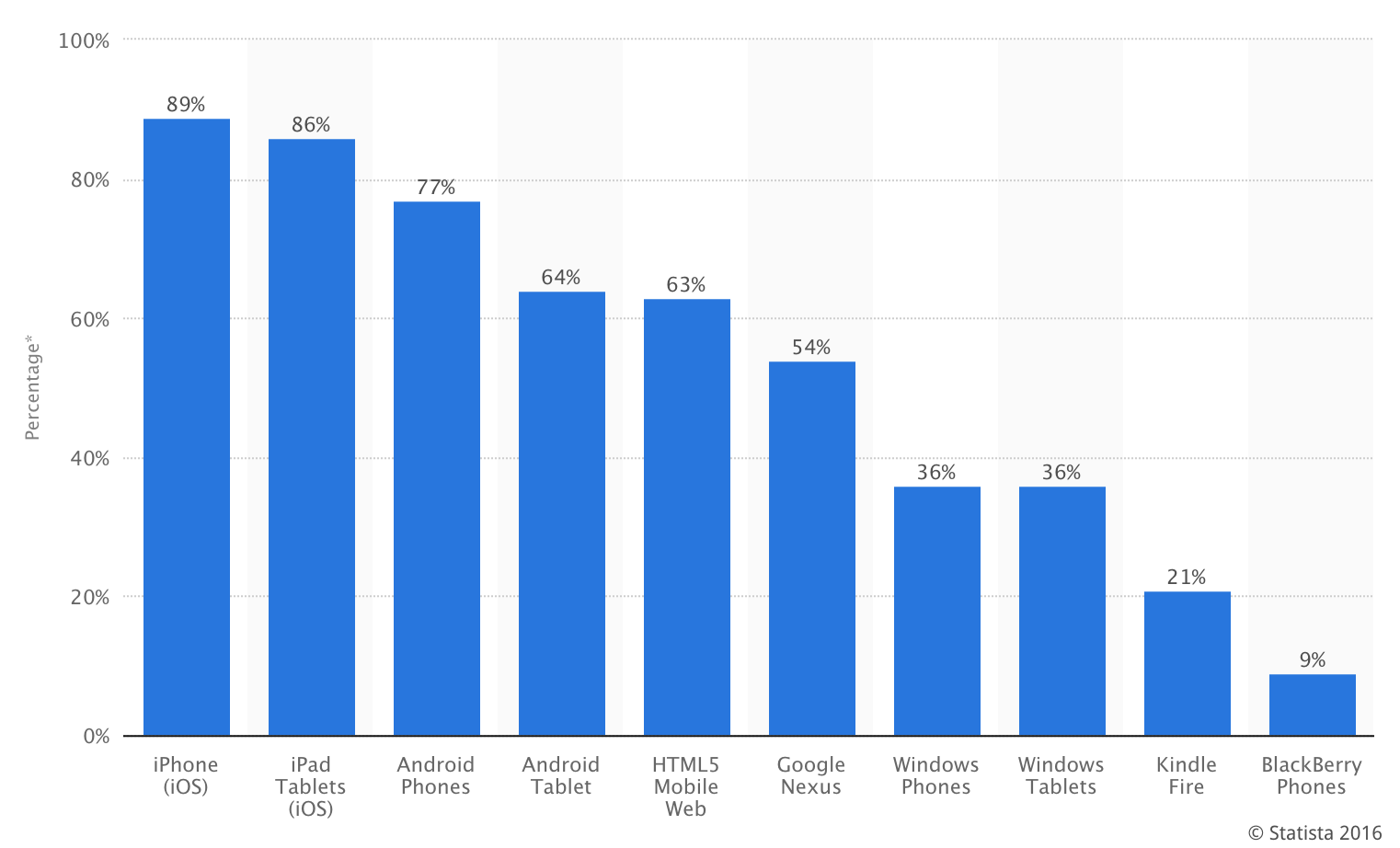


Figure : Mobile devices preferred by app developers worldwide. (Statista, 2012)

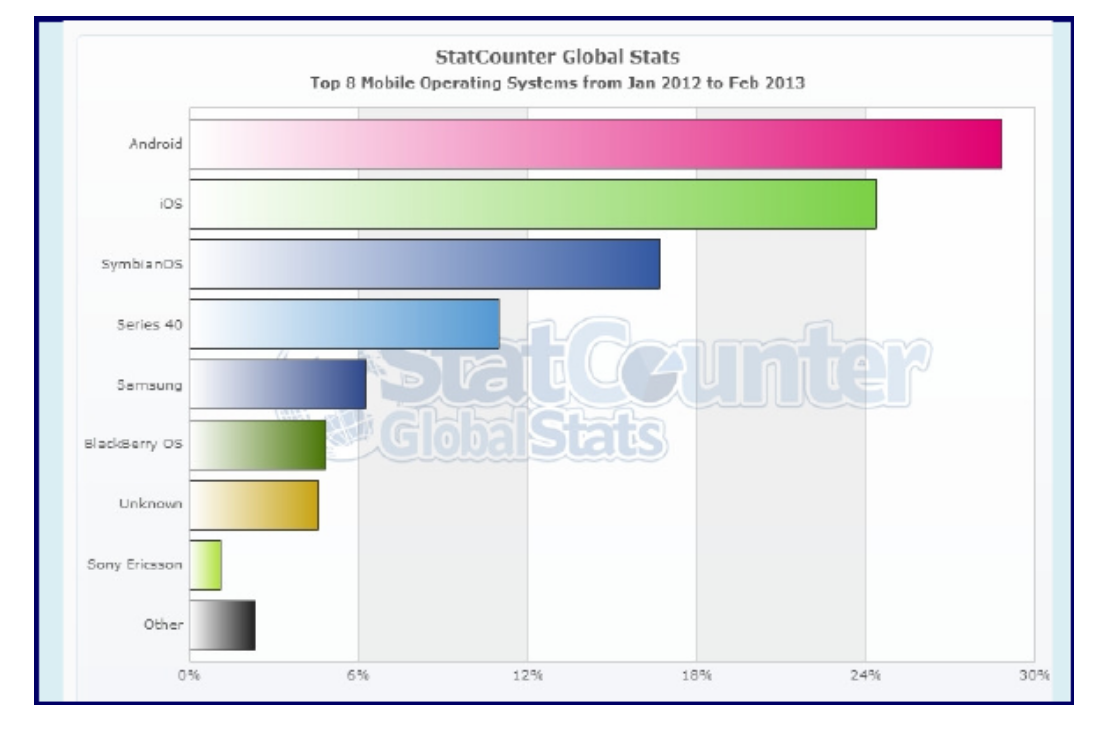


Figure : Top 8 Mobile Operating Systems. (Asokan, 2013)

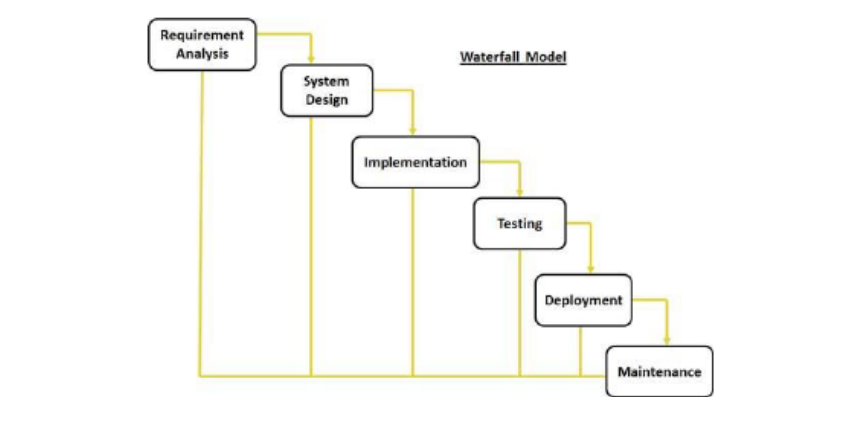


Figure : Waterfall Model (Sparsha, n.d.)

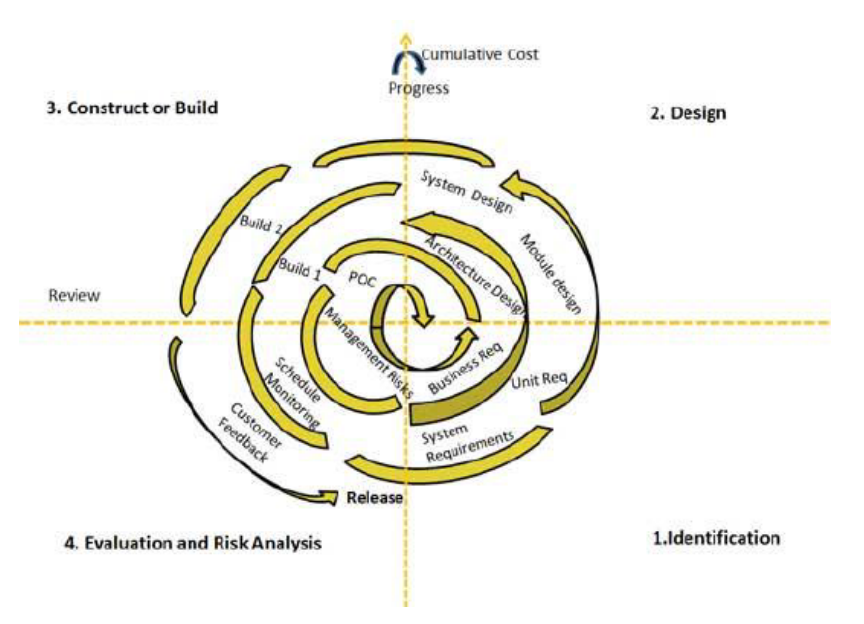


Figure : Spiral Model (Sparsha, n.d.)

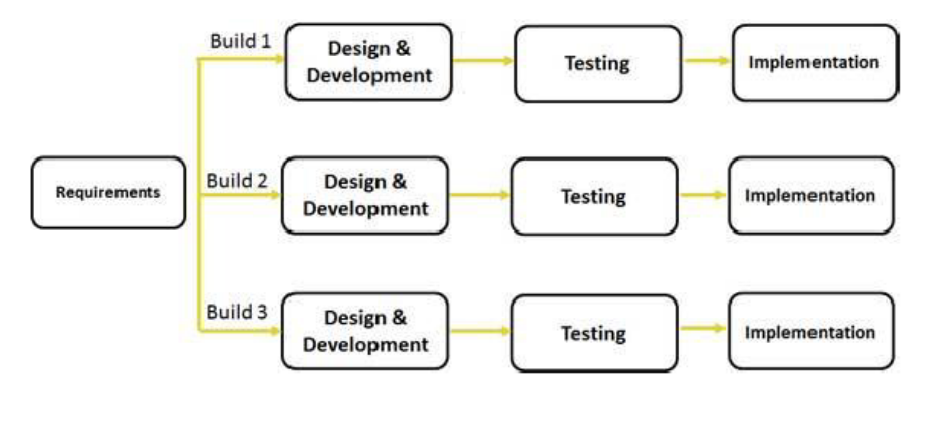


Figure : Iterative Model (Sparsha, n.d.)

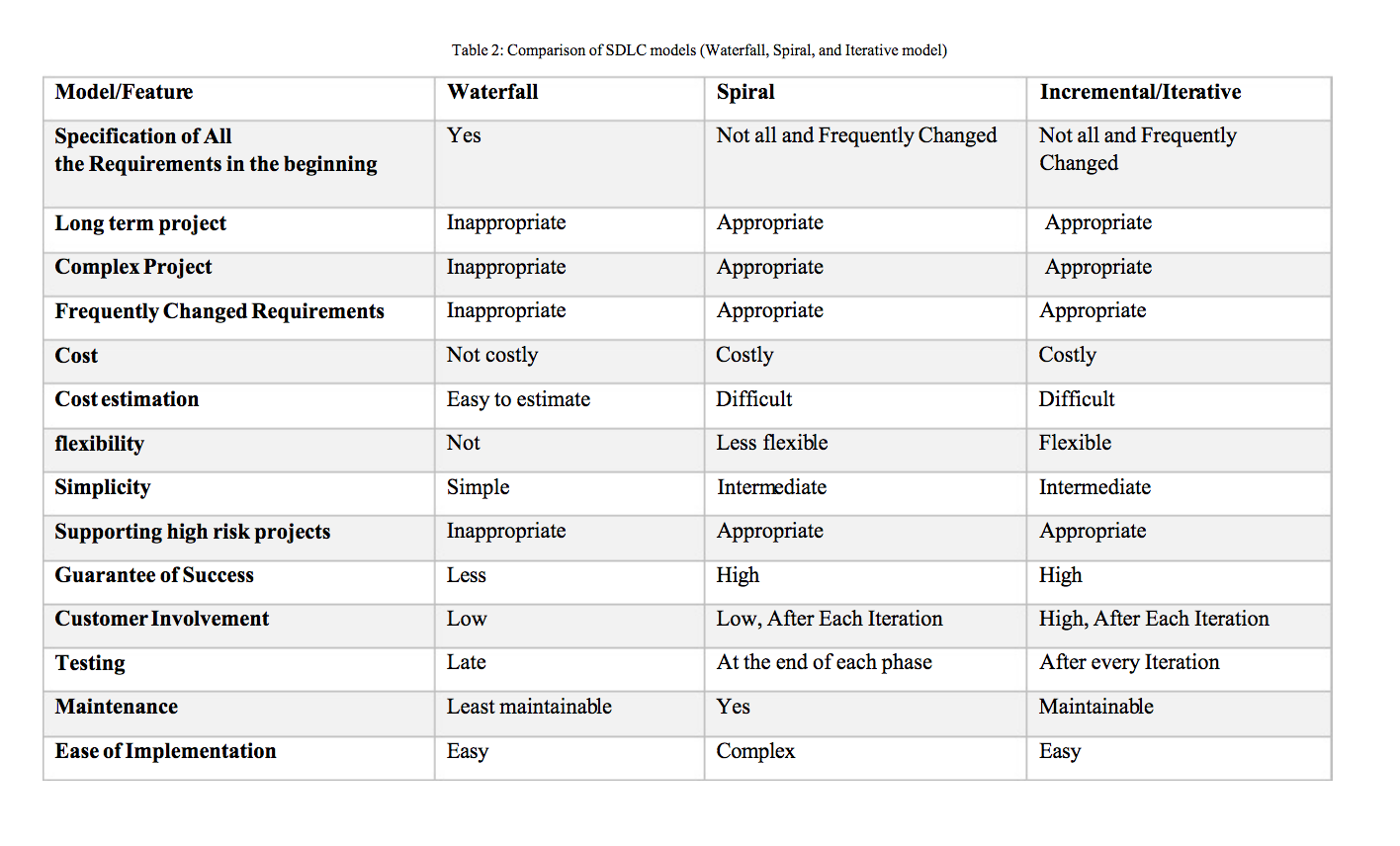


Figure : Comparison of SDLC models (Fulton, Aziz, 2015, p. 106-111)

## Appendix A

Splash Screen

The implementation of the Splash Screen was easy since it consists only from a picture.

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@color/color"  
tools:context="com.example.marthakat.sirarthurconandoyleinportsouth.MainActivity">  
  
 <ImageView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:src="@drawable/conan\_doyles\_portsmouth"/>  
</RelativeLayout>

Main Menu (Maps Layout)

The main layout, when created, was ready to use but in order to change in the requested display, the Navigation View element was added. This element is used for the creation of the slider menu. As presented below, the Navigation View element consists of the header layout, which contains the logo of the application, and the menu layout, which includes the categories.

Activity\_main\_menu.xml

<?xml version="1.0" encoding="utf-8"?>  
<android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/drawer\_layout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:openDrawer="start">  
  
 <include  
 layout="@layout/app\_bar\_main\_menu"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 <android.support.design.widget.NavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="match\_parent"  
 android:layout\_gravity="start"  
 android:fitsSystemWindows="true"  
 app:headerLayout="@layout/nav\_header\_main\_menu"  
 app:menu="@menu/activity\_main\_menu\_drawer" />  
  
</android.support.v4.widget.DrawerLayout>

In the activity\_ main\_ menu layout, the element <include>is used in order to reuse a complete layout, in this case the app\_ bar\_ main\_ menu layout.

App\_bar\_main\_menu.xml

<?xml version="1.0" encoding="utf-8"?>  
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:fitsSystemWindows="true"  
 tools:context="com.example.marthakat.sirarthurconandoyleinportsmouth.MainMenu">  
  
 <android.support.design.widget.AppBarLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:theme="@style/AppTheme.AppBarOverlay">  
  
 <android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:background="?attr/colorPrimary"  
 app:popupTheme="@style/AppTheme.PopupOverlay" />  
  
 </android.support.design.widget.AppBarLayout>  
  
 <include layout="@layout/activity\_maps" />  
  
  
  
</android.support.design.widget.CoordinatorLayout>

This layout was created by default so it had to be modified in order to include the google maps layout in the general layout. Therefore, the <include> element defines that the map layout to be included. It is important to mention that an API key is required as well as the right permissions in the Android Manifest file in order to be able to use Google Maps. (Google Developers, n.d.)

Activity\_maps.xml

<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:map="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/map"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.example.marthakat.sirarthurconandoyleinportsmouth.MapsActivity"  
 tools:layout="@layout/activity\_main"  
 />

Constraints

As mentioned above, the main layout of this activity (activity\_main\_menu.xml) uses Navigation View in order to create a slider menu. The menu’s layout is the activity\_main\_menu\_drawer.xml file. This file includes a group of items while each items is assigned to a category. The layout was created by default and it consists not only from the coding part (displayed below) but also from the menu icons. Even though Android studio offers a number of icons ready to use, the client provided icons for the categories, thus they were needed to be replaced by the client’s icons.

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
  
 <group android:checkableBehavior="single">  
 <item  
 android:id="@+id/conan"  
 android:icon="@drawable/conan\_doyle"  
 android:title="Conan Doyle" />  
 <item  
 android:id="@+id/books"  
 android:icon="@drawable/books"  
 android:title="Books" />  
 <item  
 android:id="@+id/groups"  
 android:icon="@drawable/groups\_friends"  
 android:title="Groups &amp; Friends" />  
 <item  
 android:id="@+id/news"  
 android:icon="@drawable/in\_the\_news"  
 android:title="News" />  
 <item  
 android:id="@+id/sherlock\_holmes"  
 android:icon="@drawable/sherlock\_holmes"  
 android:title="Sherlock Holmes" />  
 <item  
 android:id="@+id/spiritualism"  
 android:icon="@drawable/spiritualism"  
 android:title="Spiritualism" />  
 <item  
 android:id="@+id/sport"  
 android:icon="@drawable/sport"  
 android:title="Sport" />  
 <item  
 android:id="@+id/travel"  
 android:icon="@drawable/travel"  
 android:title="Travel" />  
 <item  
 android:id="@+id/everything"  
 android:icon="@drawable/ic\_pin\_drop\_24dp"  
 android:title="View All Locations" />  
 </group>  
</menu>

Nonetheless, the client’s icons were PNG files but the icons that the <item> element “accepts” must be xml files. As a result, the icons had to be modified so they can be used in the layout. Firstly, the files were converted into SVG files with the use of a website, called Convertio, because it was not possible to find a way of converting PNG to XML. The new SVG files can be converted into xml files by using another online service, called “Android SVG to VectorDrawable”. When the conversion is complete, the files are ready to use. (Saokosal, 2016)

## Appendix B

Maps Activity class

package com.example.marthakat.sirarthurconandoyleinportsmouth;  
  
import android.support.v4.app.FragmentActivity;  
import android.os.Bundle;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.MarkerOptions;  
  
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {  
  
 private GoogleMap mMap;  
  
 /\*This activity is not being used anymore  
 It is not part of the development process anymore  
 but its layout is used as part of MainMenu's layout\*/  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_maps*);  
 // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.*map*);  
 mapFragment.getMapAsync(this);  
 }  
  
  
 */\*\*  
 \* Manipulates the map once available.  
 \* This callback is triggered when the map is ready to be used.  
 \* This is where we can add markers or lines, add listeners or move the camera. In this case,  
 \* we just add a marker near Sydney, Australia.  
 \* If Google Play services is not installed on the device, the user will be prompted to install  
 \* it inside the SupportMapFragment. This method will only be triggered once the user has  
 \* installed Google Play services and returned to the app.  
 \*/* @Override  
 public void onMapReady(GoogleMap googleMap) {  
 mMap = googleMap;  
  
 // Add a marker in Portsmouth and move the camera  
 LatLng portsmouth = new LatLng(-1.092499, 50.797193);  
 mMap.addMarker(new MarkerOptions().position(portsmouth).title("Marker in Portsmouth"));  
  
 mMap.moveCamera(CameraUpdateFactory.*newLatLngZoom*(portsmouth, 14));// zoom 14  
 }  
  
}

## Appendix C

Main Menu class

package com.example.marthakat.sirarthurconandoyleinportsmouth;  
  
import android.app.AlertDialog;   
import android.content.DialogInterface;  
import android.content.Intent;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;   
import android.os.Bundle;  
import android.support.design.widget.NavigationView;  
import android.support.v4.view.GravityCompat;  
import android.support.v4.widget.DrawerLayout;  
import android.support.v7.app.ActionBarDrawerToggle;  
import android.support.v7.app.AppCompatActivity;  
import android.support.v7.widget.Toolbar;  
import android.util.Log;  
import android.view.Menu;  
import android.view.MenuItem;   
import android.widget.Toast;  
import com.google.android.gms.maps.CameraUpdate;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
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.io.IOException;  
  
public class MainMenu extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener, OnMapReadyCallback, GoogleMap.OnInfoWindowClickListener, GoogleMap.OnMarkerClickListener {  
  
 private GoogleMap mMap;  
 private CameraUpdate zoom;  
 private SQLiteDatabase db;  
 private Marker place;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main\_menu*);  
 Toolbar toolbar = (Toolbar) findViewById(R.id.*toolbar*);  
 setSupportActionBar(toolbar);  
  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(  
 this, drawer, toolbar, R.string.*navigation\_drawer\_open*, R.string.*navigation\_drawer\_close*);  
 drawer.setDrawerListener(toggle);  
 toggle.syncState();  
  
 //create NavigationView element and value  
 NavigationView navigationView = (NavigationView) findViewById(R.id.*nav\_view*);  
 navigationView.setNavigationItemSelectedListener(this);  
  
 // Obtain the SupportMapFragment and get notified when the map is ready to be used.  
 SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
 .findFragmentById(R.id.*map*);  
 mapFragment.getMapAsync(this);  
  
 }  
  
  
 @Override  
 public void onBackPressed() {  
 //this method is for closing the menu and return to the map  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 if (drawer.isDrawerOpen(GravityCompat.*START*)) {  
 drawer.closeDrawer(GravityCompat.*START*);  
 } else {  
 super.onBackPressed();  
 }  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 // Inflate the menu; this adds items to the action bar if it is present.  
 //getMenuInflater().inflate(R.menu.main\_menu, menu);  
 return true;  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 // Handle action bar item clicks here. The action bar will  
 // automatically handle clicks on the Home/Up button, so long  
 // as you specify a parent activity in AndroidManifest.xml.  
 int id = item.getItemId();  
  
 //noinspection SimplifiableIfStatement  
 if (id == R.id.*action\_settings*) {  
 return true;  
 }  
  
 return super.onOptionsItemSelected(item);  
 }  
  
 @Override  
 public boolean onNavigationItemSelected(MenuItem item) {  
 //method created when the NavigationView.OnNavigationItemSelectedListener is implemented  
 // Handle navigation view item clicks here.  
 int id = item.getItemId();  
  
 //for each category is clicked, the only locations which will be displayed are the ones which have the tags  
 //that belong in the selected category  
 if (id == R.id.*books*) {  
 mMap.clear();  
 Cursor books = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Society%'",null);  
 int sum = books.getCount(); System.*out*.println("!!!!!sum-books: " + sum);  
 if (books.moveToFirst() && books.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 books.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (books.moveToNext());  
 }//end if  
 } else if (id == R.id.*conan*) {  
 mMap.clear();  
 Cursor conan = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Writer%' OR Tags LIKE '%Doctor%' OR Tags LIKE '%Family%'",null);  
 int sum = conan.getCount(); System.*out*.println("!!!!!sum: " + sum);  
 if (conan.moveToFirst() && conan.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 conan.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (conan.moveToNext());  
 }//end if  
 } else if (id == R.id.*groups*) {  
 mMap.clear();  
 Cursor groups = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Friends%'",null);  
 int sum = groups.getCount(); System.*out*.println("!!!!!sum-groups: " + sum);  
 if (groups.moveToFirst() && groups.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 groups.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (groups.moveToNext());  
 }//end if  
 } else if (id == R.id.*news*) {  
 mMap.clear();  
 Cursor news = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Politics%'",null);  
 int sum = news.getCount(); System.*out*.println("!!!!!sum-news: " + sum);  
 if (news.moveToFirst() && news.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 news.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (news.moveToNext());  
 }//end if  
 } else if (id == R.id.*sherlock\_holmes*) {  
 mMap.clear();  
 Cursor holmes = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Sherlock%'",null);  
 int sum = holmes.getCount(); System.*out*.println("!!!!!sum-holmes: " + sum);  
 if (holmes.moveToFirst() && holmes.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 holmes.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (holmes.moveToNext());  
 }//end if  
 } else if (id == R.id.*spiritualism*) {  
 mMap.clear();  
 Cursor spiritualism = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Spiritualism%'",null);  
 int sum = spiritualism.getCount(); System.*out*.println("!!!!!sum-spiritualism: " + sum);  
 if (spiritualism.moveToFirst() && spiritualism.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 spiritualism.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (spiritualism.moveToNext());  
 }//end if  
 } else if (id == R.id.*sport*) {  
 mMap.clear();  
 Cursor sport = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Sport%'",null);  
 int sum = sport.getCount(); System.*out*.println("!!!!!sum-sport: " + sum);  
 if (sport.moveToFirst() && sport.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 sport.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (sport.moveToNext());  
 }//end if  
 } else if (id == R.id.*travel*) {  
 mMap.clear();  
 Cursor travel = db.rawQuery(" SELECT \* FROM portsmouth WHERE Tags LIKE '%Portsmouth%' OR Tags LIKE '%Southsea%'",null);  
 int sum = travel.getCount(); System.*out*.println("!!!!!sum-travel: " + sum);  
 if (travel.moveToFirst() && travel.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 travel.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (travel.moveToNext());  
 }//end if  
 } else if (id == R.id.*everything*) {  
 Cursor cursor = db.rawQuery("SELECT \* FROM portsmouth", null);  
 int sum = cursor.getCount();System.*out*.println("!!!!!sum: " + sum);  
 if (cursor.moveToFirst() && cursor.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 cursor.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
  
 }  
 } while (cursor.moveToNext());  
 }//end if  
 }  
  
 DrawerLayout drawer = (DrawerLayout) findViewById(R.id.*drawer\_layout*);  
 drawer.closeDrawer(GravityCompat.*START*);  
 return true;  
 }  
  
  
 @Override  
 public void onMapReady(GoogleMap googleMap) {  
 //this method is called when the OnMapReadyCallback interface is called  
 mMap = googleMap;  
  
 //create an DataBaseHelper (class) attribute in order to create the connection with the database  
 DataBaseHelper myDbHelper;  
 myDbHelper = new DataBaseHelper(this);  
 try {  
 myDbHelper.createDataBase();  
 } catch (IOException ioe) {  
 throw new Error("Unable to create database");  
 }  
  
 myDbHelper.openDataBase();  
  
 db = myDbHelper.getDatabase();  
 //when the connection is installed, we use MYSQL queries to retrieve data  
 Cursor cursor1 = db.rawQuery("SELECT \* FROM sqlite\_master WHERE type='table' AND name='portsmouth'", null);  
 Cursor cursor = db.rawQuery("SELECT \* FROM portsmouth", null);  
 int sum = cursor.getCount();  
 System.*out*.println("sum: " + sum);  
  
 if (cursor.moveToFirst() && cursor.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 //retrieve all the locations' data in order to create the markers on the map  
 Cursor cursor2 = db.rawQuery("SELECT CoordinateA, CoordinateB, Location FROM portsmouth", null);  
 for (int i=0; i < sum; i++) {  
  
 cursor.moveToPosition(i);  
 cursor2.moveToPosition(i);  
 float lat = cursor2.getFloat(0);//position-column 0  
 float lng = cursor2.getFloat(1);//1  
 String loc = cursor2.getString(2);//2  
 LatLng marker = new LatLng(lat,lng);  
 place = mMap.addMarker(new MarkerOptions().position(marker).title(loc));  
 }  
 } while (cursor.moveToNext());  
 }//end if  
  
 // Add a marker in Portsmouth and move the camera  
 LatLng portsmouth = new LatLng(50.814266, -1.071873);  
 zoom = CameraUpdateFactory.*newLatLngZoom*(portsmouth, 12);//zoom 12  
 mMap.moveCamera(CameraUpdateFactory.*newLatLng*(portsmouth));  
 mMap.animateCamera(zoom);  
  
 googleMap.setOnMarkerClickListener(this);// define method  
  
 }//onMapReady  
  
 @Override  
 public void onInfoWindowClick(Marker marker) {  
 //this method is implemented when the GoogleMap.OnInfoWindowClickListener is called  
 //not used in the development phase  
// Toast.makeText(this, "Info window clicked",  
// Toast.LENGTH\_LONG).show();  
 }//end  
  
 @Override  
 public boolean onMarkerClick(Marker marker) {  
  
 //this method is implemented when the GoogleMap.OnMarkerClickListener is called  
 final String loc = marker.getTitle();  
  
 /\*when a marker is clicked an alert window will pop up asking a question which is set below  
 two buttons are created and they are called "Yes" and "No"  
 according to the answer, different actions will occur\*/  
 AlertDialog.Builder builder1 = new AlertDialog.Builder(MainMenu.this);  
 builder1.setMessage("See Info?");//the question  
 builder1.setCancelable(true);  
  
 builder1.setPositiveButton(  
 "Yes",  
 new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 String still = null, add = null,date = null,info = null,tags = null;  
  
 Cursor cursor = db.rawQuery("SELECT \* FROM portsmouth", null);  
 int sum = cursor.getCount();  
 if (cursor.moveToFirst() && cursor.getCount() != 0) {  
 Log.*d*("initView", "inside if -> do-while, for!");  
 do {  
 Cursor cursor2 = db.rawQuery("SELECT Date, Information, Tags FROM portsmouth WHERE Location = '" + loc + "'", null);  
  
 cursor2.moveToPosition(0);  
  
 date = cursor2.getString(0);  
 info = cursor2.getString(1);  
 tags = cursor2.getString(2);  
  
 /\*due to the fact that I changed the name of two columns (Still Standing & Additional info)  
 there was a problem with retrieving the information for these two columns when I used their names  
 in the query above, so I used the code below.  
 I run the for loop for all the locations until it finds the selected one and then retrieve  
 the data for these two columns\*/  
 for (int i=0; i < sum; i++) {  
 cursor.moveToPosition(i);  
 if(cursor.getString(0).equals(loc)){  
 still = cursor.getString(1);  
 add = cursor.getString(2);  
 System.*out*.println("Let's try: " + "\n Still: " + still + "\n Addi: " + add);  
 }  
 }  
 } while (cursor.moveToNext());  
 }//end if  
  
 System.*out*.println("Loc: " + loc + "\n" + "\n Date: " + date  
 + "\n Info: " + info + "\n Tags: " + tags);  
// System.out.println("Loc: " + loc + "\n" + " Still: " + still + "\n AddI: " + add + "\n Date: " + date  
// + "\n Info: " + info + "\n Tags: " + tags);  
  
 /\*we keep the data for the specific location in order to use them when the Info Details class  
 is called in order to be able to use them\*/  
 Intent intent = new Intent(MainMenu.this, InfoDetails.class);  
 intent.putExtra("loc",loc);  
 intent.putExtra("still",still);  
 intent.putExtra("addi",add);  
 intent.putExtra("date",date);  
 intent.putExtra("info",info);  
 intent.putExtra("tags",tags);  
 Toast.*makeText*(MainMenu.this, "Loading...",Toast.*LENGTH\_SHORT*).show();  
 startActivity(intent);  
 dialog.cancel();  
 }  
 }  
  
 );  
  
 builder1.setNegativeButton(  
 "No",  
 new DialogInterface.OnClickListener()  
  
 {  
 public void onClick(DialogInterface dialog, int id) {  
// Toast.makeText(MainMenu.this, "no",Toast.LENGTH\_LONG).show();  
 dialog.cancel();  
 }  
 });  
  
 AlertDialog alert11 = builder1.create();  
 alert11.show();  
  
 return false;  
 }  
  
}

## Appendix D

Info Details class

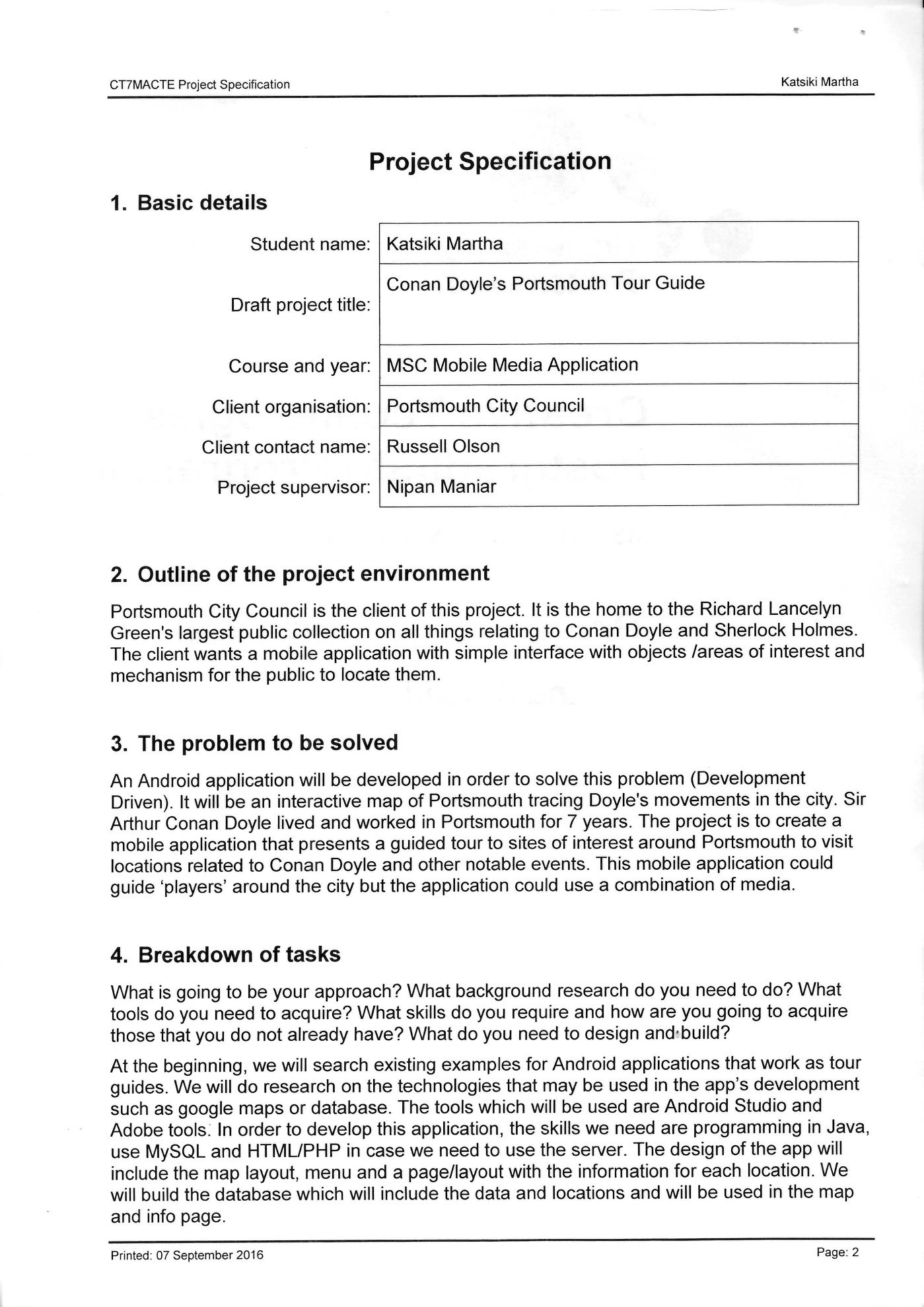
package com.example.marthakat.sirarthurconandoyleinportsmouth;  
  
import android.content.Intent;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.CheckBox;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.ViewFlipper;  
  
public class InfoDetails extends AppCompatActivity implements View.OnClickListener{  
  
 //this class is called when the user wishes to read more information about the place he has chosen  
 private String location = null, still = null, addi = null, date = null, info = null, tags = null;  
  
 private TextView address, addinfo, dt, information, tag;  
 private CheckBox yes, no;  
 private ImageView imv1;  
 private ViewFlipper vfl;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_info\_details*);  
  
 System.*out*.println("---------------InfoDetails--------------------");  
  
 //data kept from the previous class (Main Menu)  
 Intent in = getIntent();  
 location = in.getStringExtra("loc");//address  
 still = in.getStringExtra("still");//still standing Y|N  
 addi = in.getStringExtra("addi");//additional info  
 date = in.getStringExtra("date");//date  
 info = in.getStringExtra("info");//information  
 tags = in.getStringExtra("tags");//tags  
  
 //making sure the data have been retrieved successfully  
 System.*out*.println("Loc: " + location + "\n" + " Still: " + still + "\n AddI: " + addi + "\n Date: " + date  
 + "\n Info: " + info + "\n Tags: " + tags);  
  
 //create elements and set value for every textView in order to add the values and display them  
 address = (TextView) findViewById(R.id.*address*);  
 address.setText(location);  
  
 addinfo = (TextView) findViewById(R.id.*addi*);  
 addinfo.setText(addi);  
  
 dt = (TextView) findViewById(R.id.*date*);  
 dt.setText(date);  
  
 information = (TextView) findViewById(R.id.*info*);  
 information.setText(info);  
  
 tag = (TextView) findViewById(R.id.*tag*);  
 tag.setText(tags);  
  
 //two CheckBoxes were created in order to show if the the building in the specific location is still standing or not  
 yes = (CheckBox) findViewById(R.id.*yes*);  
  
 no = (CheckBox) findViewById(R.id.*no*);  
  
 //first, it is essential check if the attribute is string is null in order to avoid the Null Exception  
 // because it is an information which is not known for every location  
 //then we click the checkbox according to the answer  
 if(still != null)  
 {  
 if(still.equals("Y")) {  
 yes.setChecked(true);  
 no.setChecked(false);  
 } else if (still.equals("N")){  
 yes.setChecked(false);  
 no.setChecked(true);  
 }  
 } else {  
 yes.setChecked(false);  
 no.setChecked(true);  
 }  
  
 //we create an element in order to create a slideshow to display the images for each location  
 vfl = (ViewFlipper) this.findViewById(R.id.*viewflipper*);  
 vfl.setOnClickListener(this);  
  
 imv1 = (ImageView) findViewById(R.id.*imVF*);  
  
 //however, right now there are few photos available so we check if the chosen location is one of the locations used below  
 //if yes, then the photo available for this location will be displayed  
 if(location.equals("Bush Villa")) {  
 imv1.setImageResource(R.drawable.*bushvilla\_w*);  
 } else if(location.equals("Orontes, ship docked at Portsmouth Dockyard")){  
 imv1.setImageResource(R.drawable.*dockyard\_w*);  
 } else if(location.equals("George Hotel")){  
 imv1.setImageResource(R.drawable.*george\_hotel\_w*);  
 } else if(location.equals("31 Green Road")){  
 imv1.setImageResource(R.drawable.*green\_road\_w*);  
 } else if(location.equals("Elmwood, Owen House")){  
 imv1.setImageResource(R.drawable.*grove\_road\_for\_elmwoodhouse\_w*);  
 } else if(location.equals("Grosvenor Hotel")){  
 imv1.setImageResource(R.drawable.*grovesnor\_hotel\_w*);  
 } else if(location.equals("Guildhall (Old)")){  
 imv1.setImageResource(R.drawable.*guildhall\_w*);  
 } else if(location.equals("Guildhall (New)")){  
 imv1.setImageResource(R.drawable.*guildhall\_w*);  
 } else if(location.equals("14 High Street")){  
 imv1.setImageResource(R.drawable.*high\_street\_w*);  
 } else if(location.equals("46 Portsmouth High Street")){  
 imv1.setImageResource(R.drawable.*high\_street\_w*);  
 } else if(location.equals("Phoenix Lodge, No 257 110 High Street")){  
 imv1.setImageResource(R.drawable.*high\_street2\_w*);  
 } else if(location.equals("70 Palmerston Road")){  
 imv1.setImageResource(R.drawable.*palmerston\_road\_w*);  
 } else if(location.equals("Queen Street Synagogue")){  
 imv1.setImageResource(R.drawable.*queen\_street\_w*);  
 } else if(location.equals("23 Silver Street")){  
 imv1.setImageResource(R.drawable.*silver\_street\_w*);  
 } else if(location.equals("No 4 Southsea Terrace")){  
 imv1.setImageResource(R.drawable.*southsea\_terrace\_w*);  
 } else if(location.equals("St Agatha Church")){  
 imv1.setImageResource(R.drawable.*st\_agathas\_church\_w*);  
 } else if(location.equals("Yarborough Villa")){  
 imv1.setImageResource(R.drawable.*yarborough\_w*);  
 }  
  
 }  
  
 @Override  
 public void onClick(View v) {  
 //when the ViewFlipper element is clicked the slideshow will start  
 vfl.startFlipping();  
 vfl.setFlipInterval(3000);//3 secs  
 }  
}

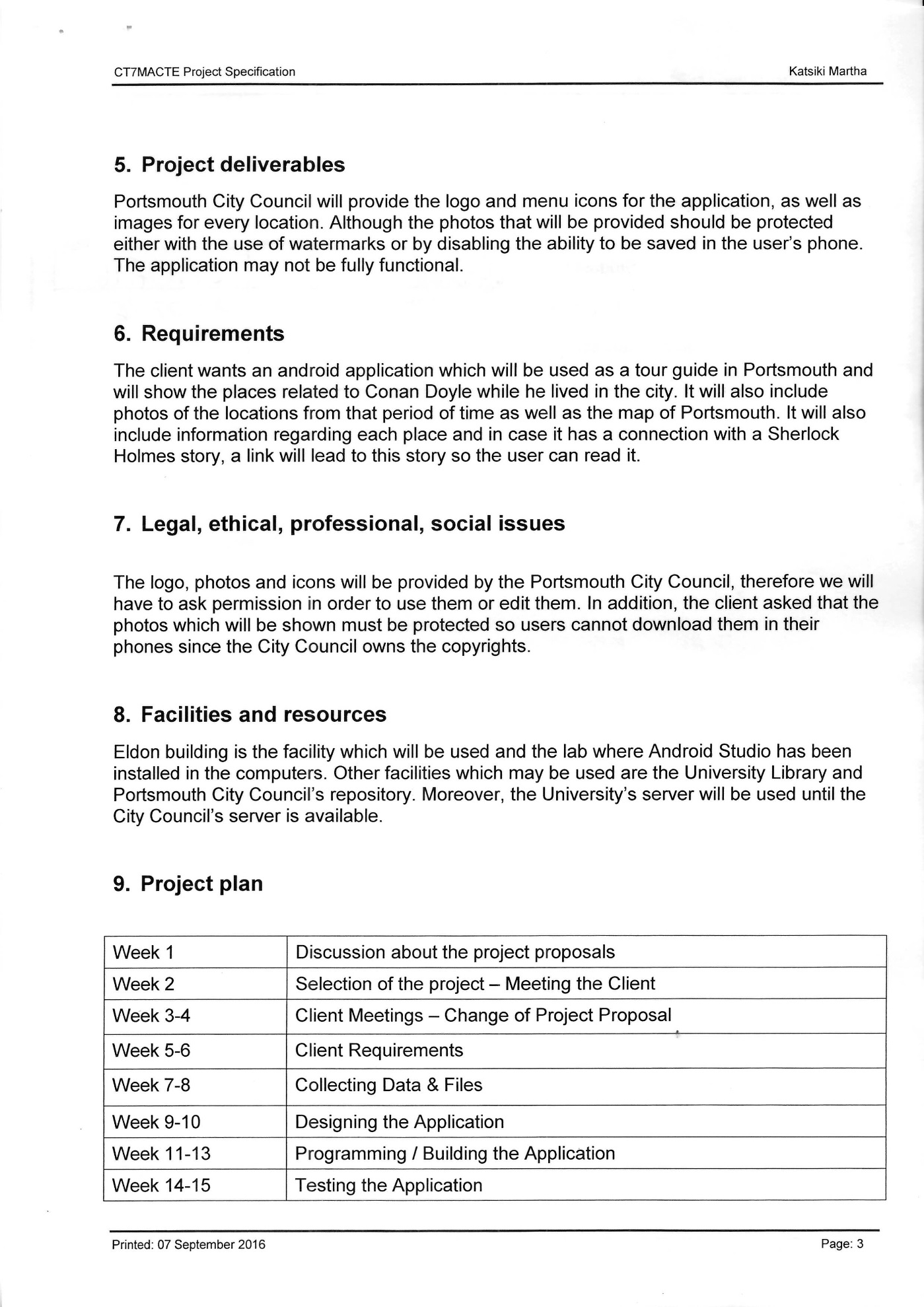
Appendix E  
  
Data Base Helper class

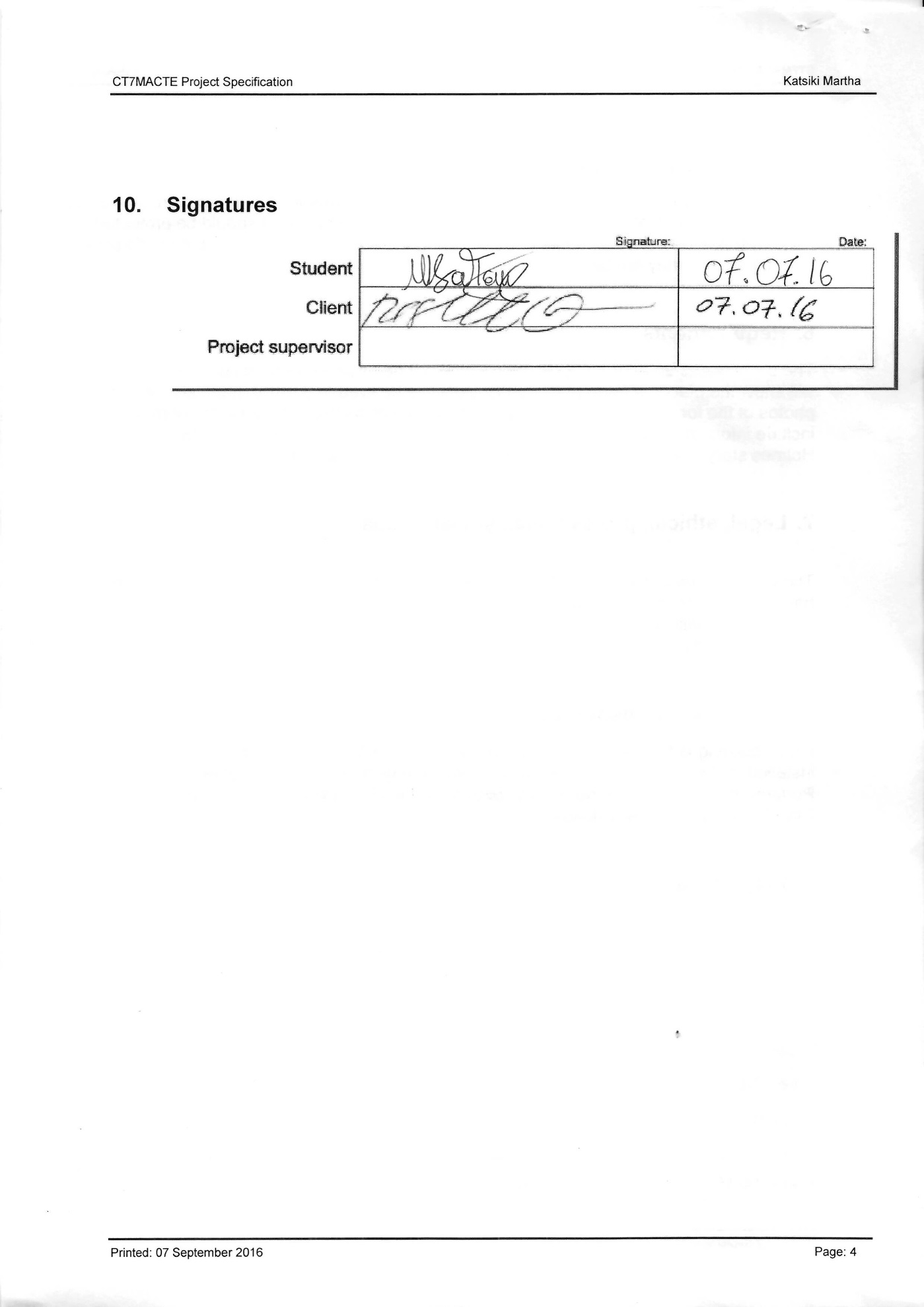
package com.example.marthakat.sirarthurconandoyleinportsmouth;  
  
import android.content.Context;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteException;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
  
*/\*\*  
 \* Created by marthakat on 8/17/16.  
 \*/*public class DataBaseHelper extends SQLiteOpenHelper {  
  
 // The Android's default system path of your application database.  
 private static String *DB\_PATH* = null;  
  
 private static String *DB\_NAME* = "cdoyle.sqlite";//database's name  
  
 private SQLiteDatabase myDataBase;  
  
 private final Context myContext;  
  
 public DataBaseHelper(Context context) {  
 super(context, *DB\_NAME*, null, 1);  
 this.myContext = context;  
 this.*DB\_PATH* = "/data/data/" + context.getPackageName() + "/" + "databases/";//database's path  
 }  
  
 public SQLiteDatabase getDatabase() {  
 return myDataBase;  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 boolean dbExist = checkDataBase();  
  
 if (dbExist) {  
 // do nothing - database already exist  
 } else {  
  
 // By calling this method and empty database will be created into  
 // the default system path  
 // of your application so we are gonna be able to overwrite that  
 // database with our database.  
 this.getReadableDatabase();  
  
 try {  
  
 copyDataBase();  
  
 } catch (IOException e) {  
  
 throw new Error("Error copying database");  
  
 }  
 }  
  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
  
 }  
  
 public void createDataBase() throws IOException {  
  
 boolean dbExist = checkDataBase();  
  
 if (dbExist) {  
 // do nothing - database already exist  
 Log.*d*("createDataBase", "do nothing - database already exists");  
 } else {  
  
 // By calling this method and empty database will be created into  
 // the default system path  
 // of your application so we are gonna be able to overwrite that  
 // database with our database.  
 this.getReadableDatabase();  
  
 try {  
  
 copyDataBase();  
 Log.*d*("createDataBase", "database does not exist");  
  
 } catch (IOException e) {  
  
 throw new Error("Error copying database");  
  
 }  
 }  
  
 }  
  
 private boolean checkDataBase() {  
  
 SQLiteDatabase checkDB = null;  
  
 try {  
 String myPath = *DB\_PATH* + *DB\_NAME*;  
 checkDB = SQLiteDatabase.*openDatabase*(myPath, null,  
 SQLiteDatabase.*OPEN\_READONLY*);  
 Log.*d*("checkDataBase", "openDatabase");  
  
 } catch (SQLiteException e) {  
  
 // database does't exist yet.  
 Log.*d*("checkDataBase", "database does't exist yet");  
  
 }  
  
 if (checkDB != null) {  
  
 checkDB.close();  
  
 }  
  
 return checkDB != null ? true : false;  
 }  
  
 */\*\*  
 \* Copies your database from your local assets-folder to the just created  
 \* empty database in the system folder, from where it can be accessed and  
 \* handled. This is done by transfering bytestream.  
 \* \*/* private void copyDataBase() throws IOException {  
  
 // Open your local db as the input stream  
 InputStream myInput = myContext.getAssets().open(*DB\_NAME*);  
  
 // Path to the just created empty db  
 String outFileName = *DB\_PATH* + *DB\_NAME*;  
  
 // Open the empty db as the output stream  
 OutputStream myOutput = new FileOutputStream(outFileName);  
  
 // transfer bytes from the inputfile to the outputfile  
 byte[] buffer = new byte[1024];  
 int length;  
 while ((length = myInput.read(buffer)) > 0) {  
 myOutput.write(buffer, 0, length);  
 }  
  
 // Close the streams  
 myOutput.flush();  
 myOutput.close();  
 myInput.close();  
  
 }  
  
 public void openDataBase() {  
  
 // Open the database  
 String myPath = *DB\_PATH* + *DB\_NAME*;  
 myDataBase = SQLiteDatabase.*openDatabase*(myPath, null,  
 SQLiteDatabase.*OPEN\_READONLY*);  
  
 }  
  
 @Override  
 public synchronized void close() {  
  
 if (myDataBase != null)  
 myDataBase.close();  
  
 super.close();  
  
 }  
}

## Appendix F









## Appendix G

