Final Report

Application for Police Force

(CW) – Community Watch

Mustafa Azaden

School of Computer Science and Informatics

Supervised by Alun D Preece

Moderated by Xianfang Sun

Student Number: 0915083

Abstract

This report presents the design, implementation and reflections of an android application used by communities to report incidents to police services. Leading from the interim report research, a great deal of information was gathered on how the police service takes statements and stores provided information and existing applications that allow the general public to report or view crimes in their neighbourhood. A list of core requirements was developed from the study, which led to implementing a prototype application. Following from the knowledge gathered, a brief outline of the changes made from the interim report are given along with the new features implemented. The conclusion critically analyses the product created and how it would fair if it were deployed to the police service.

Acknowledgements

I would like to extend my gratitude to the following persons who have made the completion of this project possible: my supervisor, Professor Alun D Preece, for his vital encouragement and support throughout the project, my moderator, Dr Xianfang Sun, for his productive feedback from the interim report, all the Computer Science faculty members and staff who aided me with various problems I encountered, and especially to my family and friends who provided the help and support needed to complete this project.

Contents

1 Introduction 5
2 Specification and Methodology 6
2.1 Derivation of requirements from interviews 6
2.2 Incremental Methodology 7
3 Design 9
3.1 Interaction between components within the system 9
3.2 Class diagrams 12
4 Architecture 14
4.1 Application architecture 14
4.2 Website architecture 15
4.3 Database architecture 16
5 User interface 19
5.1 Application interface 19
5.1.1 Main menu activity 19
5.1.2 Registration activity 19
5.1.3 Login activity 20
5.1.4 Tutorials activity 20
5.1.5 Incident notification activity 21
5.1.6 Reporting activity 21
5.1.7 Incident Map activity 21
5.1.8 Comment activity 22
5.1.9 User profile activity 22
5.2 Website 22
5.2.1 Login page activity 23
5.2.2 Live incident feed activity 23
5.2.3 Map activity 24
5.2.4 Add new user activity 24
6 Implementation 25
6.1 Application implementation 25
6.2 Website implementation 32
6.3 Database implementation 35
6.3.1 Website implementation with database connection 38
6.3.2 Application implementation with database connection 40
6.4 Important features 42
6.5 Main problems 44
6.5.1 Application 44
6.5.2 Website 45
7 Testing 46
7.1 Software verification 46
7.2 Usability testing 53
7.2.1 Application testing 53
7.2.2 Website testing 55
7.3 External usability testing 55
8 Evaluation and Future Work 56
8.1 Application testing feedback 56
8.2 Critical evaluation 59
8.3. Future work 60
9 Result 61
9.1 Goals achieved 61
9.2 Goals not achieved 61
10 Conclusion 62
11 Reflections 63
12 References 64
13 Appendix 66

1. Introduction

The purpose of this report is to discuss the requirements, specifications and designs from the interim report in more detail, and then implement and test the project.

The requirements will be discussed in more detail where the interim failed to do so, as I did not have contact with any of the sectors relating to the application developments. This will consist of completing the interviews with police officers and community groups to fully understand the needs of both sectors. We will then look into the implementation of the application and website that has been created. Both have been tested and built upon considerably during the course of the project.

The testing phase will demonstrate the software verification and the usability of the application in a real world environment. Driven from the test results the application will be modified to fully satisfy the requirements.

Evaluation will look into the goals achieved by developing the application, and also the limitations faced during the project, improvements, and future possibilities after submission of the final report. The evaluation section will also consist of a critical evaluation of the application to determine whether it could be deployed by police services across the country to offer easy interaction between community groups and police services.

Finally, a conclusion will reflect on the overall experience and learning gathered during the project.

2. Specification and Methodology

2.1 Derivation of requirements from interviews

The requirements that were produced in the interim report were gathered from two main sources. The first source of requirements were obtained from background research on existing applications, which guided me to create a list of functions which I thought might be useful to the police service and community group. From the list of functions I developed a prototype version using the basic layout and features which Android offers.

The second source was done using the prototype as a proposal report, to conduct interviews to show the police officers and community groups in order to help them understand the full concept of the application, this included a number of questions to help establish an understanding of the features that could be added or removed to make the application useable for client and user side.

Many questions were raised during the interviews with community groups and police officers:

- How would the information be presented to the police on the website? After further discussion it was determined that the most appropriate outlet for the project was a webpage that offers police officers a map that pin points incident locations, and when a police officer clicks a particular pin point they access information about the incident
- Using a trusted network can reduce the number of users, which means that users will not have access to this app. Therefore, if this app was adopted by the police there will be a range of legal and ethical issues to contend with i.e. allowing freedom of speech whilst protecting against bogus reports. After further discussion it was understood that a trusted network could be implemented in the future to stop bogus incident reporting and inappropriate comments
- Crowd sourcing was mentioned a number of times as this app could be used to alert the community. This would allow users to receive updated information about incidents in their area and have the ability to comment on incidents that have been reported by other users
- If the app user has no access to the internet, then they should be offered an alternative option to report an incident. One of the most appropriate alternatives is a text line service via which users can send messages from their mobile phone. Due to the scope of the project, implementing a text line service could not be achieved
- User information should be protected and be stored in a different location to the reported incident details. Police services feel strongly about securing user information. During this project I will be looking into encryption systems and structuring my database in a different manner
- The User Location Finder (GPS) should work indoors or in built up areas, as app users should not be limited in where they can report incident
- Integration of information from the Police API to give users news on what police officers are doing to improve the community
- Allow a user to choose a category for the type of incidents they are witnessing, e.g. Theft, Anti-Social Behavior etc. This feature was mentioned a number of times during the interviews because, if incidents have been reported, the police service can then send the necessary informants to handle the incident. For example, a health and safety offence would require an employee from the Health and Safety Executive to be sent to the location to investigate further

The rest of the requirements that were produced were made to fit around the core points of the initial report.

2.2 Methodology

The software development methodology that was initially chosen for this project was the evolutionary model. This was because the project would start by producing a working prototype of the application in short amount of time followed by testing and refining during the implementation, this would produce an efficient and less erroneous product, but unfortunately the amount of time taken to create the working prototype took much longer than anticipated because of time constraints and difficulty implementing certain features on the app. Therefore the methodology was changed to a new approach known as 'incremental development' [1].

Incremental development enabled sections of work to be broken down further into subsections, which allowed me to develop each subsection in the model independently of each other. This meant that the system architecture was designed initially and all the features would be implemented in subsections known as increments. Once an increment was complete it was merged into one final system. One of the most beneficial aspects of this methodology was that the program would not be a single piece of software, but a number of smaller modules which form a bigger picture of the system. This methodology allows the code that I implement to be developed in future without the need to format the structuring of the code. Below, an incremental methodology diagram demonstrates the plan followed in the process [Figure 1].

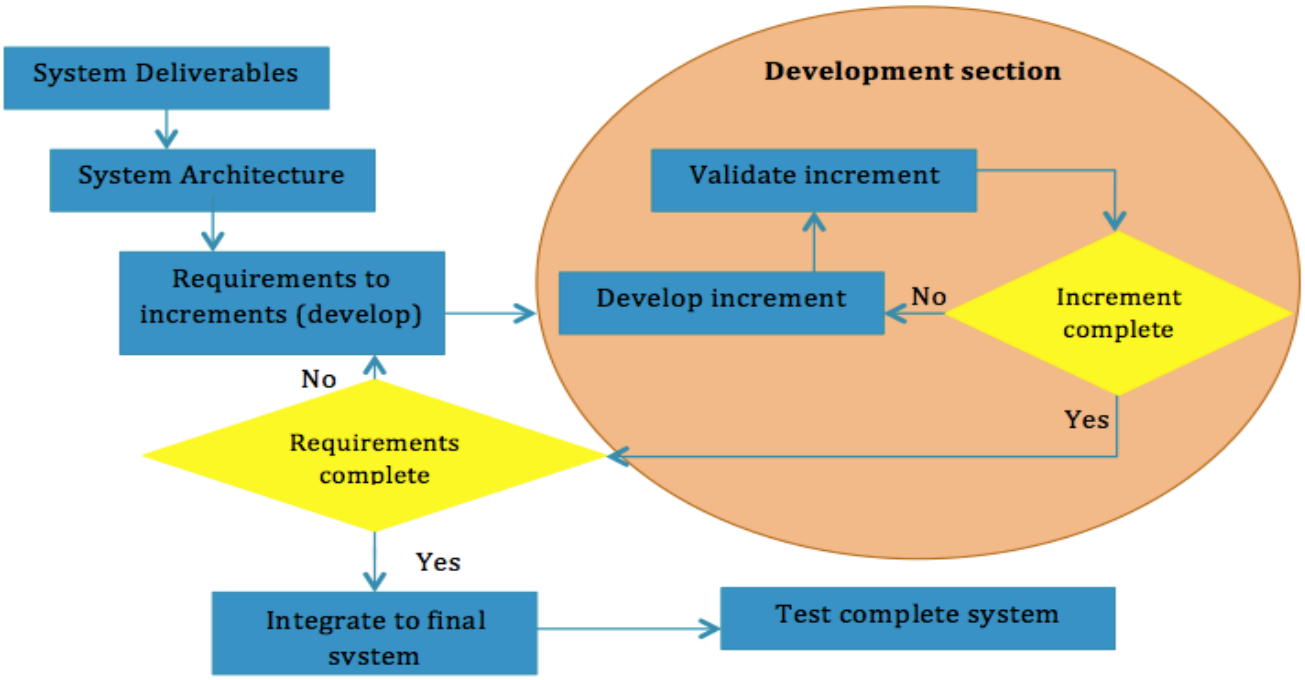


Figure 1: incremental methodology diagram

2.2 Methodology

Project Increments

The increments were chosen are as follows:

Task 1 – Complete Android application.
Increment 1 – Produce the Tutorial fragment activity.
Increment 2 – Produce the Registration activity.
Increment 3 – Produce the Login activity.
Increment 4 – Produce the Incident Notification activity.
Increment 5 – Produce the Reporting activity.
Increment 6 – Produce the Incident Map activity.
Increment 7 – Produce the Comment activity.
Increment 8 – Produce the User Profile activity.
Increment 9 – Produce a Main Menu activity to bind all activities.

Task 2 – Complete database integration.
Increment 1 – Create and integrate the Activation table.
Increment 2 – Create and integrate the Registration table.
Increment 3 – Create and integrate the Report table.
Increment 4 – Create and integrate the Comment table.
Increment 5 – Create and integrate the Police login table.

Task 3 – Complete website.
Increment 1 – Produce the Login page.
Increment 2 – Integrate database with website.
Increment 3 – Produce a live incident feed webpage.
Increment 4 – Produce Map for viewing posted incidents.
Increment 5 – Privileges to add new users to the Community Watch app.

3. Design

In this section we will look at the graphical interface of the community watch application and website, and discover exactly how each component will work using a series of UML diagrams. These diagrams will show the flow of data, functions and methods used within the system. As with any project, issues will be foreseen. Therefore we will discuss the plans devised to overcome them.

3.1 Interaction between components within the system

To illuminate the interaction needed between the components within the system in order to report and view incidents via a mobile application to help build trust between community groups and the police service. A Trusted Community user will be given community watch app to view and report incidents via an Internet connection with the database server which one.com provide. On the other side of the system, a website will be given to police services to gather reported incidents from the online database server with aid of PHP scripts. This will help the police officer to view reported incidents and conduct a standard investigation process [Figure 2].

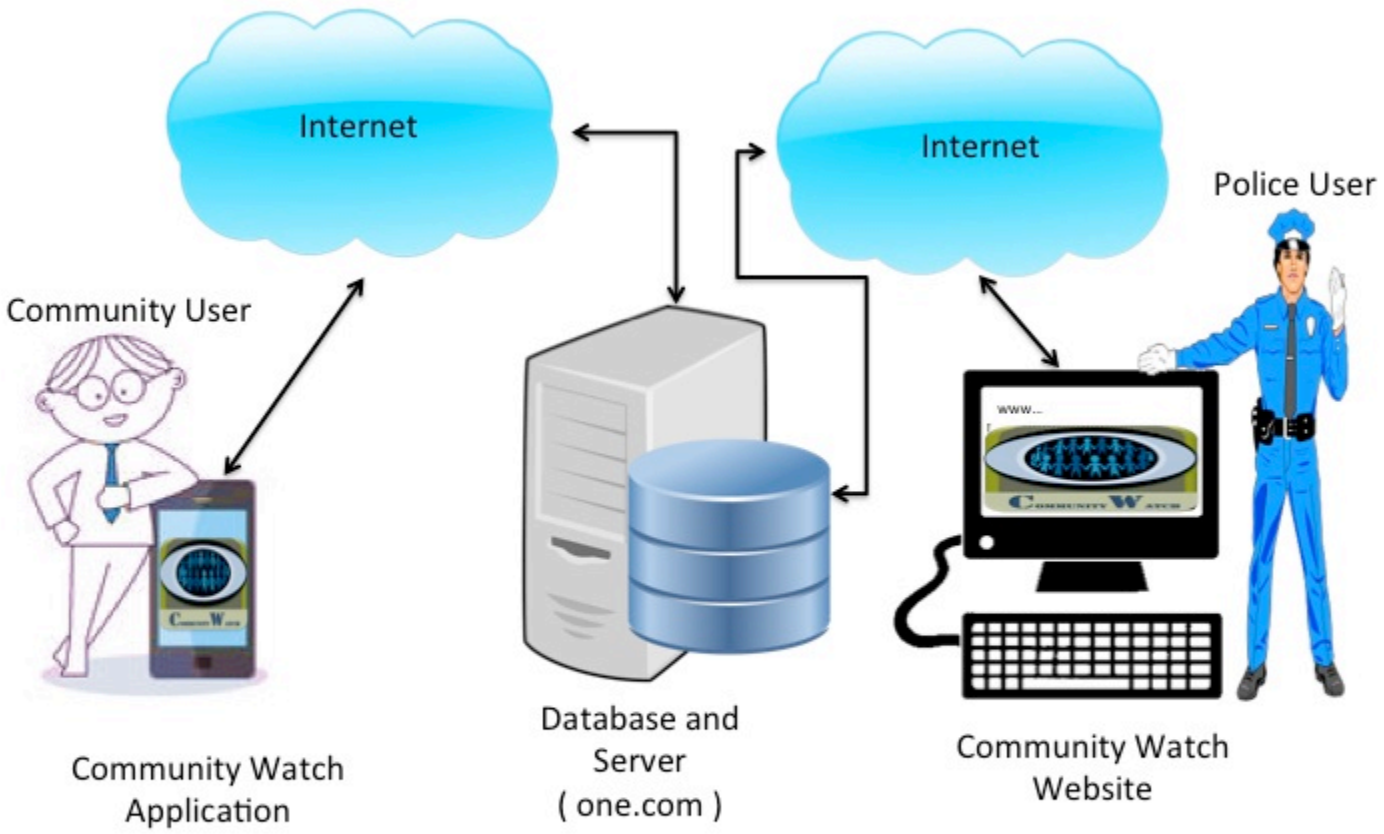
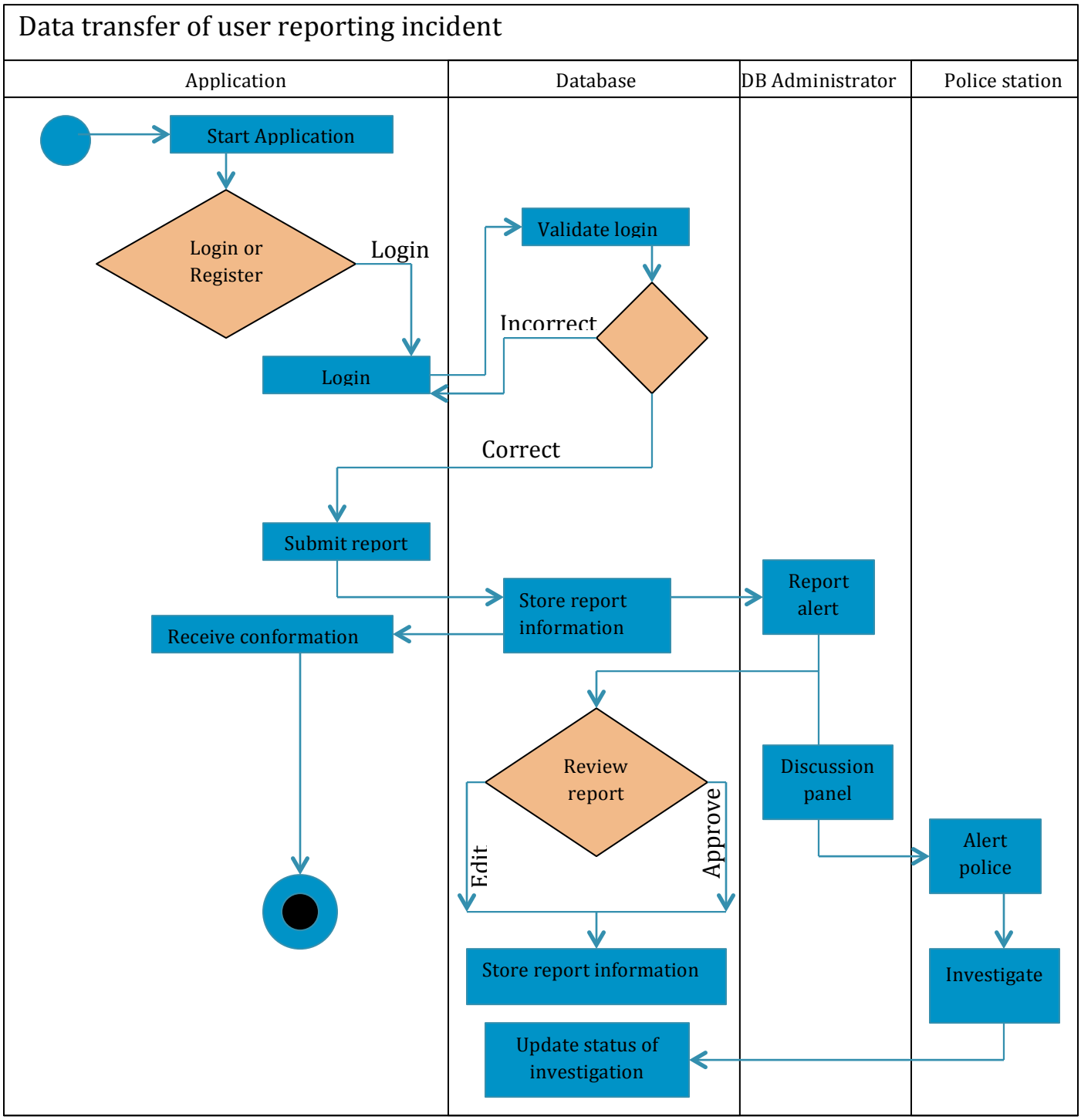
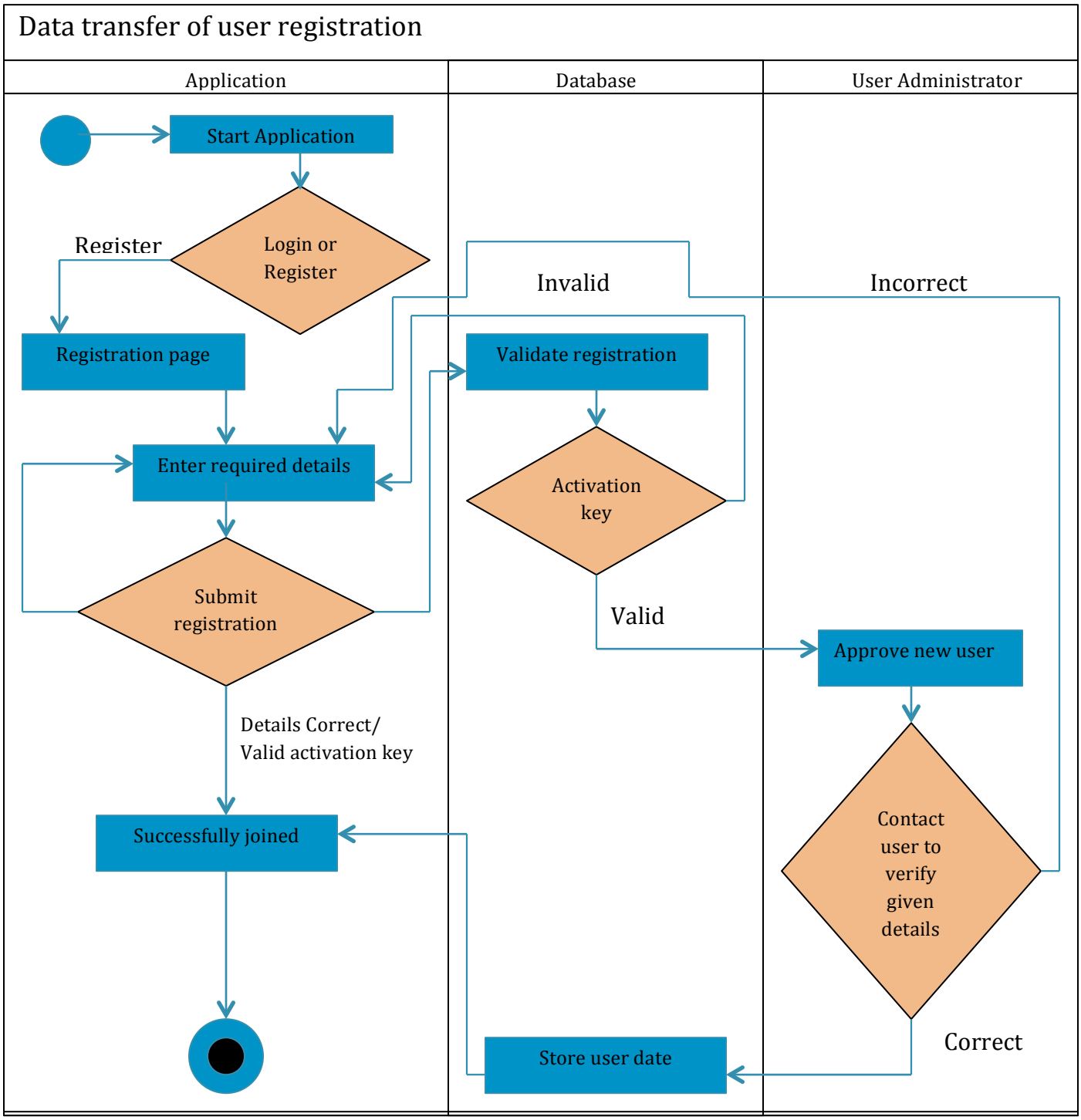


Figure 2: interaction between components within the system

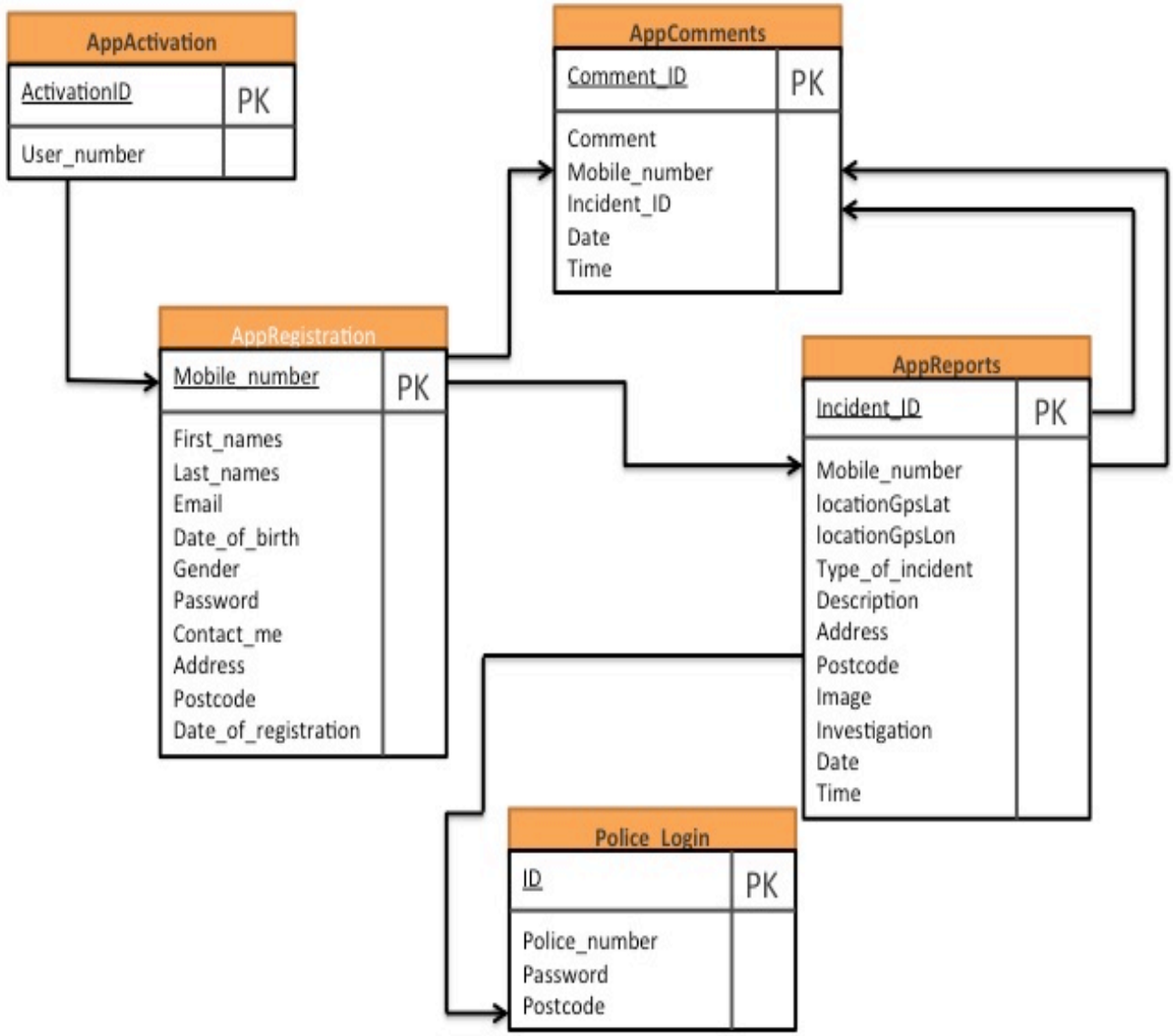
3.1.1 UML Modelling - This model shows how the application interacts with the database and police when user uses the Login and Report Incident functions.



3.1.2 UML Modelling – This model shows the interaction between the application and database when a new user registers with the application.

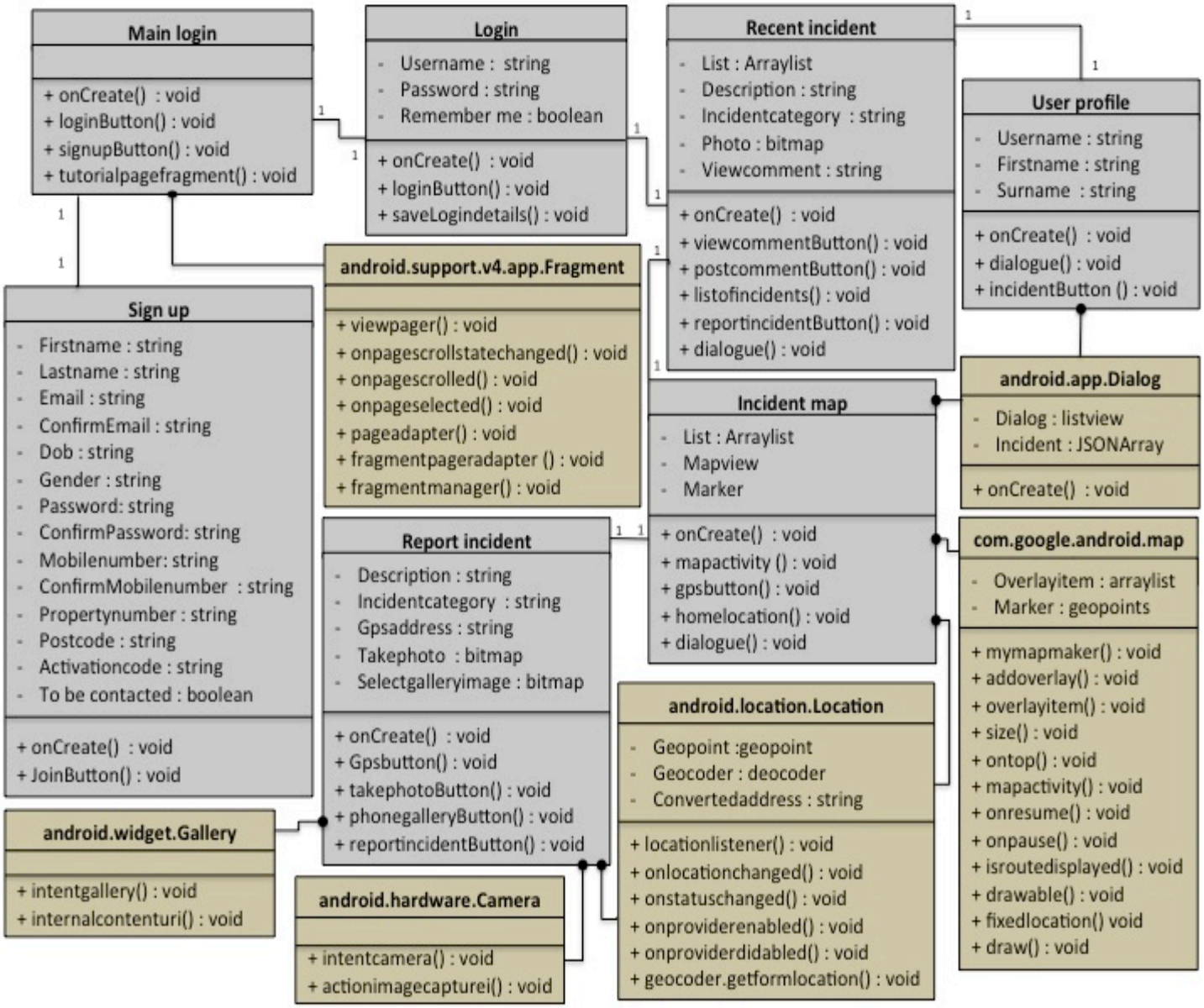


3.2.1 Class diagram – This diagram shows the relationships between entities in the application.



3.2.2 Class diagram

– This diagram shows the relationship between different classes in the app and Android classes where needed to produce each of the tasks. The classes with a brown background and "Android" title are the default classes located in the Android API. For example "android.app.Dialog" is used to load a default instance of a dialog to the user's phone. However, some classes need extensions to work with any version of Android. For instance, "android.support.v4.app.Fragment" imports a default instance of a fragment from the Google compatibility folder; this makes fragments function even if the user's phone is running an older version of Android.



4. Architecture - Application

4.1. Application Architecture

As the application was to be developed using Java, it was obvious that it would conform to the object-orientated paradigms upon which the language is based. In each class there were a number of objects that had to be used which made each class extremely large. For instance, Android contains reference classes, which binned classes in the project, the reference class contains a list of unique IDs which refer to drawables, identifiers, attributes, layouts and strings used throughout the program [13,16].

With Android, the reference class is defined in the "R.java" class which contains IDs of each resources as follows:

Layout class contains IDs which refer to the xml layouts which were created in the program:

```
public static final class layout {
    public static final int border=0x7f030000;
    public static final int listlayout=0x7f030001;
    public static final int main=0x7f030002;
        }
```

String class contains IDs which refer to the strings variables which were declared in the program:

```
public static final class string {
    public static final int app_name=0x7f040001;
    public static final int hello=0x7f040000;
   }
```

Drawable class contains IDs which refer to all the graphical resources which were created in the program:

```
public static final class drawable {
    public static final int incidentlogo=0x7f020000;
    public static final int bluebutton=0x7f020001;
    public static final int boxgreen=0x7f020002;
        }
```

If we look at the layout class, we can see that "main" is declared as layout in the program, which can be found in the res/ layout/ folder. When layouts are created in Android they will automatically be given identifiers to prevent confusion.

For the project I will be using many different classes to prevent confusion, and I will be importing prebuild functions that Google API offers to developers. The class diagram in [Design: 3.2.2] was designed to display relevant classes which need to be created to meet the requirements of the project.

4.2. Website Architecture

The website was designed to be developed using the HTML5, Bootstrap CSS, PHP and JavaScript web languages. HTML5 and Bootstrap were mainly used for the graphical aspect of the website. PHP scripts were used to integrate the website to the database that stored all the incidents and user details. JavaScript was used to manipulate information retrieved from the database [17,18].

Bootstrap is an open-source framework developed by Twitter which comes with a number of well tested style sheets that can be used to create a generic, professional looking user interface by using the CSS provided by the library. The purpose of using Twitter Bootstrap in this instance is to make a professional looking web-application which can be developed rapidly. This allows for greater emphasis on developing other elements of the application. Another advantage of using Twitter Bootstrap is that due to the generic CSS class names; we can easily swap out the default Bootstrap style sheets at a later date and replace them with a style sheet developed by another web developer [4].

The data flow diagram (Figure 3) below shows the passage of information interacting between the website and the database.

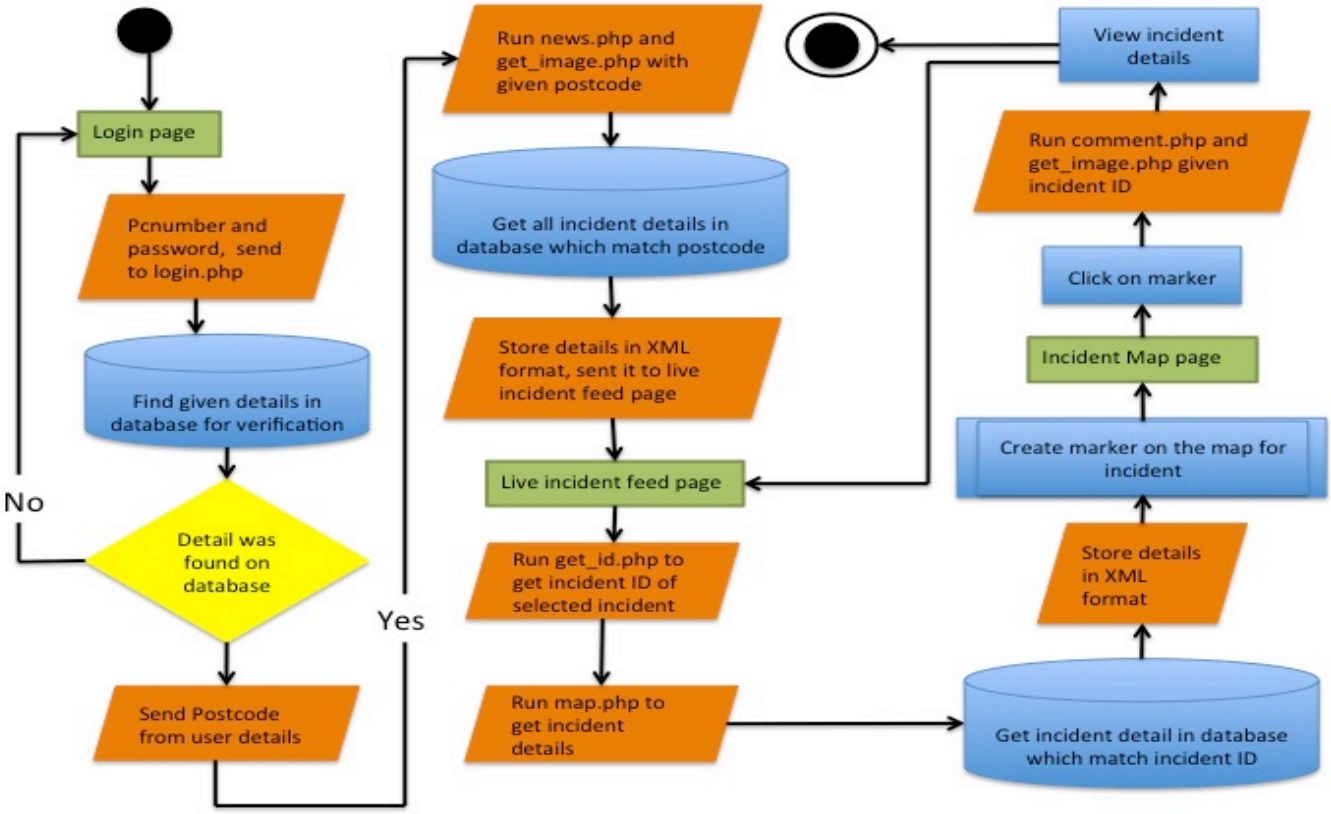


Figure 3: Database and website interaction

A PHP script was required to retrieve information from the database and display it in XML format, as Google maps only parses XML data. In order to read the XML data the Google API offers a function which reads XML called GDownloadURL, this runs the PHP script and retrieves given data and formats it according to the xml tags.

A standard XML style sheet is used to display information when the PHP script runs, for example, when retrieving information about an incident each row in the database gets given a tag according to the column name.

```
<marker>
    <INCIDENT_ID= "023321" >
        <MOBILE_NUMBER> 07790800000 </MOBILE_NUMBER>
        <locationGpsLat> -3.223322444 </locationGpsLat>
        <locationGpsLon>  6.22322442  </locationGpsLon>
        <TYPE_OF_INCIDENT> Vandalism  </TYPE_OF_INCIDENT>
        <DESCRIPTION> Someone has been painting on my garage wall ….</DESCRIPTION>
        <ADDRESS> 8, Pen-Y-Cae, Cardiff </ADDRESS>
        <POSTCODE> CF14  </POSTCODE>
        <IMAGE> BLOB (BASE64 ENCODED STRING), converted to .jpg image </IMAGE>
        <INVESTIGATION>  1  <I/NVESTIGATION>
        <DATE> DD-MM-YYYY </DATE>
        <TIME> HH:MM:SS  </TIME>
    </INCIDENT_ID>
</marker>
```

When the XML style sheet displays the format of the information, Google maps can then display the information as a pin point on the map, along with information related to the incident i.e. the description gets displayed in clear HTML format when a user clicks on a particular incident icon.

4.3. Database Architecture

The database will gather and source all the information sent and retrieved by the community watch app. The following tables were created in the database to deal with information gathered using the app and website:

AppComments			
Column	**Type**	**Length**	**Attributes**
COMMENT_ID	INT	100	Primary key, Auto increment
INCIDENT_ID	INT	100	
MOBILE_NUMBER	VARCHAR	11	
COMMENT	VARCHAR	1000	
DATE	DATE	DD/MM/YYYY	
TIME	TIME	HH:MM:SS	

AppReports			
Column	**Type**	**Length**	**Attributes**
INCIDENT_ID	INT	100	Primary key, Auto increment
MOBILE_NUMBER	VARCHAR	11	
locationGpsLat	VARCHAR	17	
locationGpsLon	VARCHAR	17	
TYPE_OF_INCIDENT	VARCHAR	100	
DESCRIPTION	VARCHAR	1000	
ADDRESS	VARCHAR	50	
POSTCODE	VARCHAR	8	
IMAGE	BLOB		
INVESTIGATION	INT	1	
DATE	DATE	DD/MM/YYYY	
TIME	TIME	HH:MM:SS	

AppRegistration			
Column	**Type**	**Length**	**Attributes**
MOBILE_NUMBER	VARCHAR	11	Primary key
FIRST_NAMES	VARCHAR	100	
LAST_NAMES	VARCHAR	100	
EMAIL	VARCHAR	50	
DATE_OF_BIRTH	DATE	DD/MM/YYYY	
GENDER	VARCHAR	10	
PASSWORD	VARCHAR	100	
CONTACT_ME	VARCHAR	3	
ADDRESS	VARCHAR	50	
POSTCODE	VARCHAR	8	
DATE_OF_REGISTRATION	DATE	DD/MM/YYYY	

AppActivation			
Column	**Type**	**Length**	**Attributes**
ActivationID	INT	20	Primary key, Auto increment
User_number	VARCHAR	11	

Police_Login			
Column	**Type**	**Length**	**Attributes**
ID	INT	20	Primary key, Auto increment
Police_Number	VARCHAR	20	
Password	VARCHAR	20	
Postcode	VARCHAR	20	

In order to integrate the tables to the app, I used the class diagram in [Design: 3.2.1] to show the relationships between each table and how they interact with each other.

For example, when a user wants to register with the app they need to fill in the registration page which is connected to the AppRegistration table and AppActivation. When the user submits the information the following process takes place:

Compare the user MOBILE_NUMBER and activationID with the list of mobile numbers that are in the AppActivation table, if the mobile number and activationID is authorised, then all given user information gets sent to the AppRegistration table and is stored for future access.

When the user has successfully registered they then have the access to Report Incident. When reporting an incident, the user's MOBILE_NUMBER will be sent along with incident information to the AppReports table. When retrieving the report information from the database, the MOBILE_NUMBER is found in the AppRegistration table to identify which user has reported the incident.

A similar requirement is needed for comments on the incident. To comment on an incident, the INCIDENT_ID is required so that the comment is referenced to the right incident. The MOBILE_NUMBER is also required so that the police can know who commented on the particular incident.

5. User Interface - Application

As the aim of this project is to design a community watch app, developing an accessible and usable user interface is one of the most important aspects of the application development process. The app will be targeted at all age groups and people who are not familiar with smart phones. The user interface will be the only thing the users will see when using the app, therefore the app needs to cater to all types of users by following the general layouts and structures that all Android applications follow.

5.1. Application Interface

The application will consist of several pages. The fundamental elements to be implemented will be: Registration, Login, Incident Notification, Reporting, an Incident Map, and the ability to leave comments. Along with these features, the basic components of the app will be included: a tutorial, managing a user profile, and a good Main Menu (the heart of the application, which will offer registration and login options to the user).

5.1.1 Main Menu activity

The initial Main Menu activity will consist of a tutorial page, and login and signup buttons. This is the first page users see when they load the app. The app logo will be displayed on the tutorial page to make users aware of what application they have opened.

5.1.2 Registration activity

When a user selects the “Sign-up” button they will be transferred to a registration screen. In order to register, a user will have to declare several items of information in order to verify that they are who they say they are; this gathered information can then be used to build a trusted network. The details requested by the application are needed so that, if they wish, the police can contact the person who reported the incident. This information can help prevent bogus reporting, as the police will be in possession of enough information on a user to be able to carry out a judicial process for wasting police time.

Due to a lengthy registration process, the registration form will be split into three pages:

- Page 1(First name, Last name, Email, Date of birth, Gender)
- Page 2 (Password, Mobile Number, Property Number, Postcode)
- Page 3(Activation code and checkbox - to be contacted or not)

The three pages will stop the user becoming overwhelmed with fields, focusing their attention on just a smaller set of fields at any one time.

Any information gathered from the application about users will be processed in a professional manner that will adhere to all statutory laws and regulations placed by the UK government, e.g. The Data Protection Act 1998 and data encryption. The police can only access user information if they wish to talk to users who have reported an incident or commented on an incident. However, if the user indicates that they do not wish to be contacted when registering, then the police will not have access to the user's contact details until they get a court hearing.

5.1.3 Login activity

Figure 4: login activity

When a user selects the "login" button on the Main Menu they will be transferred to login activity. Here the user can enter information to access the app; two-text boxes will be presented to them - one for their mobile number and another for their password. When the user fills in the required information they can then select "login" to access the application and retrieve and report information about their community [Figure 4].

5.1.4 Tutorial activity

Figure 5: Tutorial activity

The tutorial activity is included in the Main Menu. Here the user can learn the purpose of the app with the aid of visual images and a step-by-step guide on how to use the app. At the end of the tutorial a short paragraph is displayed consisting of terms and conditions and an explanation of the legal obligations of the app in regards to the handling of personal data and ensuring the safety of the its users [Figure 5].

5.1.5 Incident notification activity

When a user logs in successfully, recent incidents in the user's locality will be displayed in list format. Additional information shown on the list will include an investigation level for each incident (red icon = hasn't been viewed by police, yellow icon = under investigation by the police, green icon = the incident has been resolved). Viewing incidents in list format allows the user to access any incident from the list and obtain more information about the particular incident [Figure 6].

Figure 6: Incident notification activity

5.1.6 Reporting activity

When a user exits "Incident notification activity", they will be navigated to a set of tabs which give the user the ability to report incidents, view incidents via the map and view their user profile. The first tab is a report incident tab. This will present a number of steps which the user has to take in order to report an incident. A simple layout will be used to suit users of both advanced Android phones and basic phones.

The first thing the user will be asked to do is take a picture of the incident or upload a previously taken photo saved on their phone. When a user uploads the picture, it will be shown on the page as a form of verification. The second stage is to identify the user's location using the GPS tracker system. The third stage is to choose a category, which is predefined on the app. The final step is to describe the incident in plain text and then select "Report Incident" [Figure 7].

Figure 7: Report activity

5.1.7 Incident Map activity

The second tab is the incident map, which offers users the option to search for incidents near the home location that they registered with, or at the current GPS location. The default setting is GPS location, so when the user selects the "incident map" tab, they will be shown a map with many pins that represent incidents around their location. The user will have the option to view the incidents in greater detail by selecting the corresponding pin on the map [Figure 8].

Figure 8: Incident Map activity

5.1.8 Comment activity

Figure 9: Comment activity

When a user selects an incident from the list of incidents or by choosing a pin on the map, they will be presented with the incident details in a clear manner. Users will be given two options; one of the options is to view comments made relating to a particular incident by selecting "view comments". This will display comments made by other users. The second option is to write a comment if they have additional information to add which may help the police with their investigations [Figure 9].

5.1.9 User profile activity

The third tab offers users a clear view of their personal information. They can also view the number of incidents they have reported and how many more incidents they need to report to become a trusted user. Users can access a list of incidents they have reported and see the investigation level of the each. Also, users have the option to comment on any of the incidents they have reported to provide more information to the police to help them with their investigations [Figure 10].

Figure 10: User profile activity

5.2. Website Interface

The website interface was designed to allow police officers to determine retrieved information from the app. The main features of the website are a login page, a live incident feed to alert police about incidents reported around the area that they are responsible for, a map for viewing posted incidents, and an add new users function.

5.2.1 Login page activity

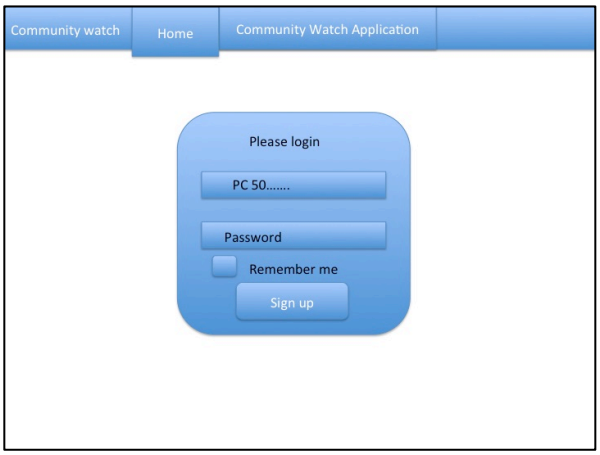


Figure 11: Login page activity

The first page on the website is the login page. The login page will consist of basic requirements i.e. the Pc number and password. The Pc number is the number given to the police officer when they first joined the police force, and the password is set by the administrator directly in the database. This will prevent any unwanted user access. The login page also offers a link to download the community watch app. This will allow users download the application in the quickest manner possible [Figure 11].

5.2.2 Live incident feed activity

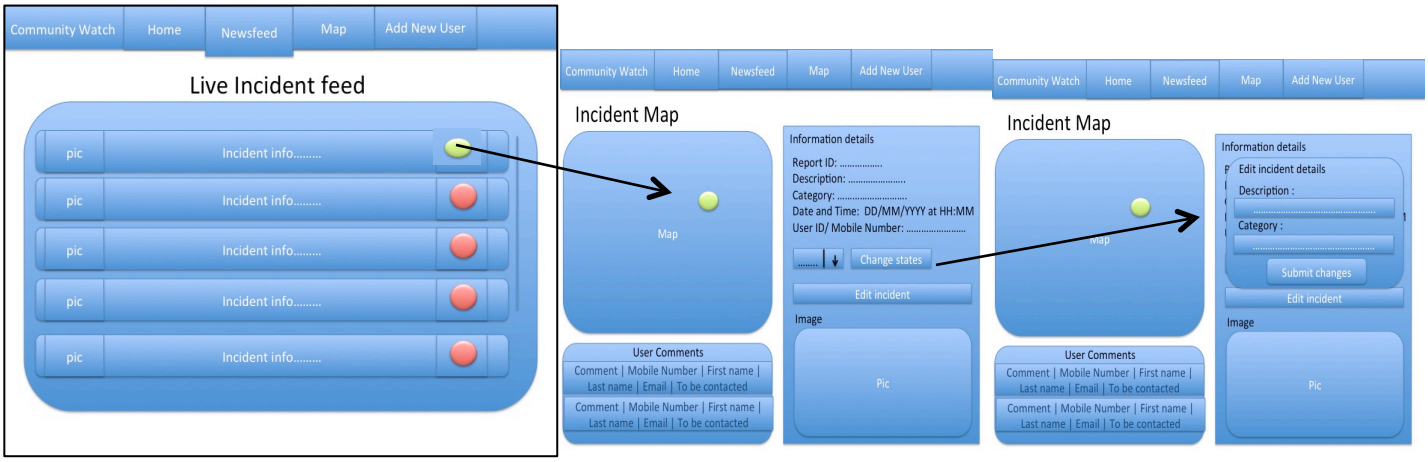


Figure 12: Live incident feed activity

When a user has logged in successfully, they will then be directed to a “live incident feed”. This will display all incidents in the area that a particular officer is responsible for. This will help police officers to deal with reported incidents in a quicker manner, as the system will update with new incidents. Police officers can view incidents in the list in detail by clicking on the incident they wish to view, as shown in [Figure 12]. Police officers are also able to change the investigation level to inform the community that the incident has been looked at. The most important feature of all is the ability to edit incident details to prevent offensive language or misleading categorisation, as seen in [Figure 12].

5.2.3 Map activity

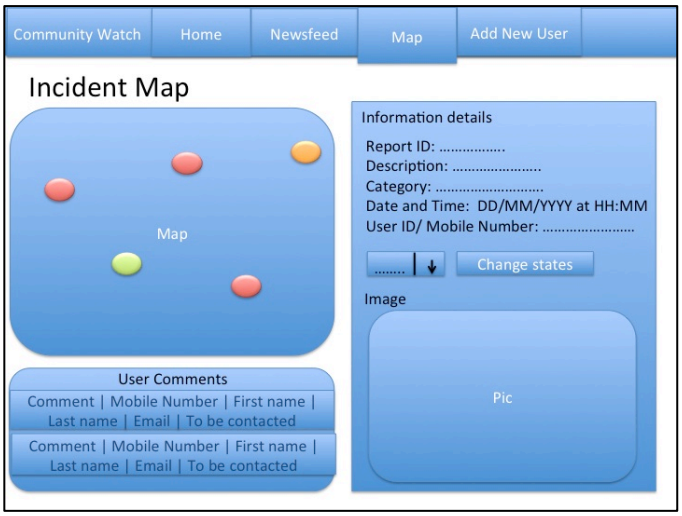


Figure 13: Map activity

The incident map will consist of a Google map, incident information and comments made by app users. The map will display all the incidents that have been posted to the app. A pin will be shown for every incident reported. This will identify the location and address of the incident. When a police officer clicks on one of the pin markers, the details of the incident and comments by app users will be displayed. The officer can change the investigation level to update the app users on the progress of the investigation. The incident map page will be clearly formatted and the contents will be easily readable, as seen in [Figure 13].

5.2.4 Add new users activity

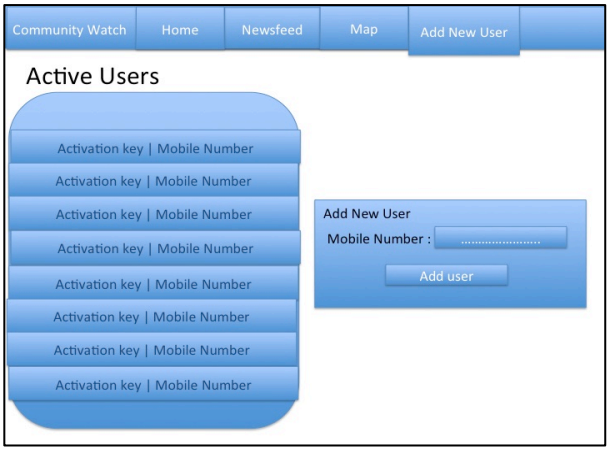


Figure 14: Add new users activity

The page for adding new users allows police officers to view all the active users who use the community watch app. If they wish to add a new user to the system they can enter a user's mobile number then submit it. A new activation key will then be generated, which they can use to register. This will help limit the number of users accessing the app and, most importantly, will build a trusted network, meaning that information provided relating to incidents can be trusted, therefore preventing abuse of the system [Figure 14].

6. Implementation - Application

6.1. Application implementation

The first step to take in developing the app was to research and understand the basics of an Android program. In order to understand the way an Android program functions, a few exercise programs were developed. Further to this I decided to create a prototype that showed the visual side of the app. This was used during initial interviews with police and community groups to help explain the features required for such an app. A tutorial on how to create basic Android programs with aid of Google Map API and a screenshot of the prototype can be found in [Appendix 1].

The next step was to create the user interface for the app. This involved using XML format code using Eclipse. Before creating the interface for the app, I researched the layouts which most Android programs follow [2,13-16]. The most important requirements are that the app has to have a clear layout, consistent colour theme, and user notification and verification when the user carries out a task on the app.

Upon understanding the Android platform requirements, I began to develop the layout.

Main Menu and Tutorial

Figure 15: Main Menu

To provide the user with the full app experience, when I first developed a prototype the tutorial page was placed on a different task page. However, after researching Android platform requirements, I became aware that it was more appropriate to place the tutorial page on the main menu, so that when the application loads the user can view the functions of the app.

The tutorial page is using a new feature which Android released in 2012. This feature is called 'fragments', and it gives developers access to multiple fragments in a single activity [3]. This will offer swift navigation to users when reading the tutorial page. This feature was very important to the app as it allows the user to swipe the "community watch" logo to explore the tutorial page and, also, when the user exits the main menu page the fragment will be distorted automatically to free up memory for other tasks in the app [Figure 15].

When I initially researched the fragments layout, I considered using a fragment display for the main menu page but, unfortunately, when I decided to implement a simple button within the fragment to call another class, i.e. 'login class', I found out that fragments can't have their own independent action listener . This was a problem as button require action listeners to perform a given task. To overcome this problem I used fragments for the tutorial page, but I resized the "viewpage" to reduce the size of the fragments and a standard layout for the two buttons below (login, signup). This gave me the flexibility to split the page into two tasks, as seen in [Figure 16, 17].

```
<android.support.v4.view.ViewPager
        android:id="@+id/tabviewpagers"
        android:layout_width="fill_parent"
        android:layout_height="390dp"
        android:layout_weight="1" />
  </LinearLayout>
   <Button
     android:id="@+id/login"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:layout_alignParentLeft="true"
     android:layout_below="@+id/linearLayout2"
     android:padding="20dip"
     android:background="@drawable/buttonst"
     android:text="Log in" />
   <Button
     android:id="@+id/Button01"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:layout_alignParentLeft="true"
     android:layout_below="@+id/sendtofragmentview"
     android:padding="15dip"
     android:background="@drawable/buttons"
     android:text="Sign up" />
```

Figure 16: XML fragment with aid of viewpage

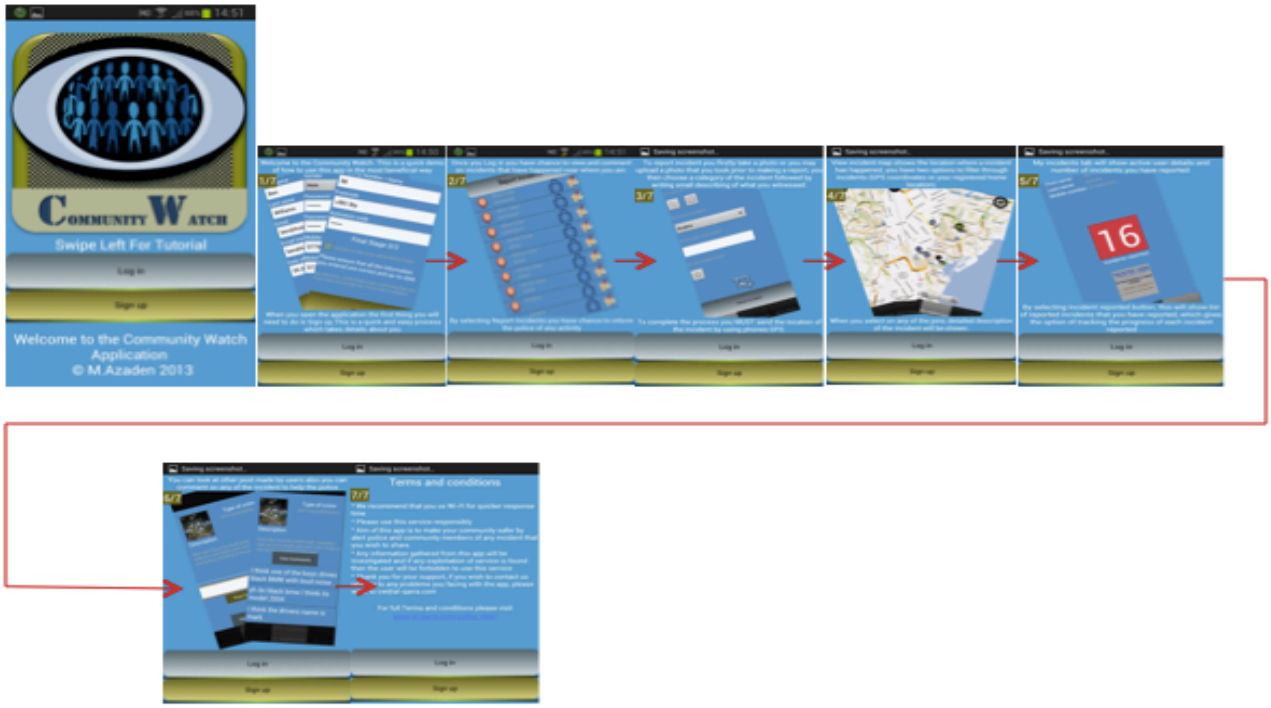


Figure 17: Tutorial page with fragment swipe motion

Registration activity

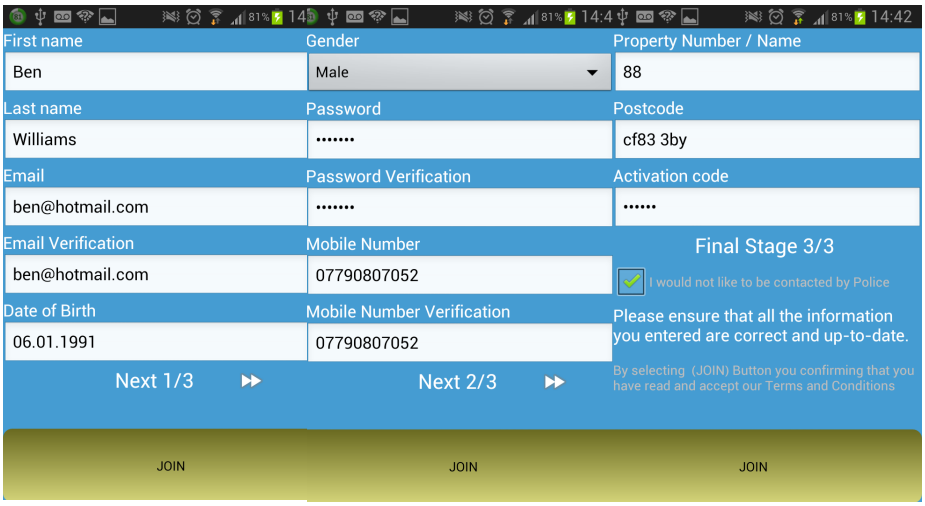


Figure 18: Registration activity

The registration page is the bedrock on this application. It limits the number of users and therefore the gathered information can be trusted. I considered using fragments layout which can be used to display empty text fields in a swift motion, but unfortunately when developing the registration page I discovered that not many mobile phones are compatible with the fragment layout. For this reason I researched different ways that Android offers a similar effect. One of the options was to use different XML layouts and bind them together in a central XML file using “<include/>”. To overcome the swift motion effect, Android offers a “horizontal scroll view” that allows the user to move from one XML layout to another with a swipe. As described above, the Android platform wants all its applications to be user friendly, and therefore tasks on each page should be processed with minimum user interaction [Figure 18].

The handling of personal information has to correspond with the rules and regulations of the Data Protection Act 1998. As this project is a prototype and will not be used in real life, the following acts and precautions will not be taken. However, if I was to deploy this app in a real police system, then storing personal information (addresses, postcodes, mobile numbers, dates of birth), extra safety procedures would have to be taken in regards to the way information is stored and retrieved on the database. The first step would be to notify the Information Commissioner’s Office that the handling of personal data is taking place. Looking into ways of making the data transactions between the app and the database secure I discovered the cryptographic hash function. This is a popular system that Android developers use to encrypt data transactions and prevent data hacking. This will be explained in detail later on in [6.4. Important Features].

Login activity

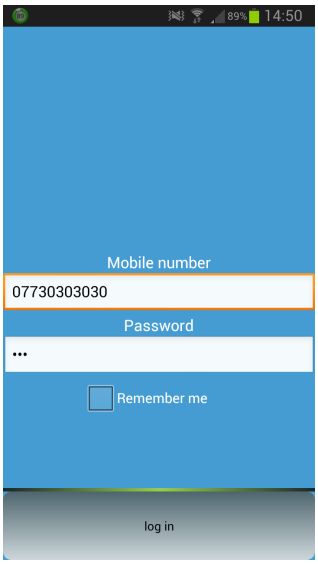


Figure 19: login activity

Here the user can enter user information to access the app. Two-text boxes will be used; one for a mobile number and another for a password. To make the transaction secure I will be looking into password encryption later on in the report. The given information will be transferred and compared with the information on the database. This will also be explained later in the report. When a user fills in the required information they can then select the “login” button to access the application if the given information corresponds with the data on the database [Figure 19].

Incident notification activity / Comment activity

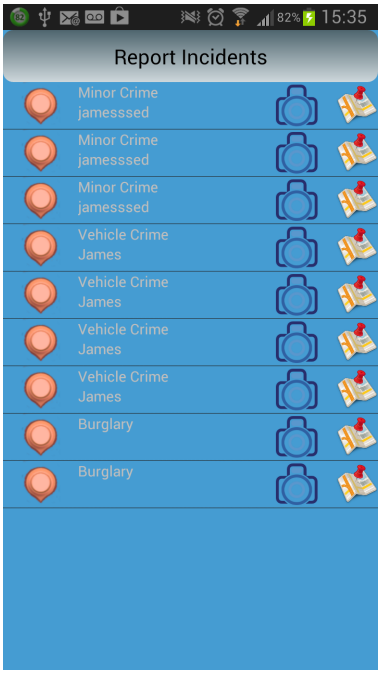


Figure 20: Incident notification activity

When a user logs in successfully, a list of recent incidents around the user's home location will be displayed in list format. To display additional information relating to the incident, i.e. the investigation level or description of the incident on the page, was very difficult as the default standard for list function on Android does not support two different elements being displayed in one list table. Looking at the customised list I discovered a simple way of formatting the list which involved using hashmap to display the retrieved XML data based upon the tags associated with each field.. To bind the customised XML I used "listview", which allows a custom list to override the default setting. Finally, to add the information from hashmap to the list, I used "simpleadapter", which allows dynamic information to be stored on the fly. Using hashmap is beneficial as it resizes whenever it is not used, and therefore freeing up memory [Figure 20].

Using a custom list gave me greater flexibility to position incident information on the list tab, i.e. incident level, incident description and incident category. Finally, to allow users to select any incident on the list and view more information, I used the function: "OnitemClickListener", which gives each tab on the list a separate action listener. To view the incident information I used a conventional method which most Android programmers use called "Dialog". This is created on the fly and displays information on the particular incident in a clear format on an XML layout. This can be seen in [Figure 21].

Figure 21: Android XML layout

Reporting page

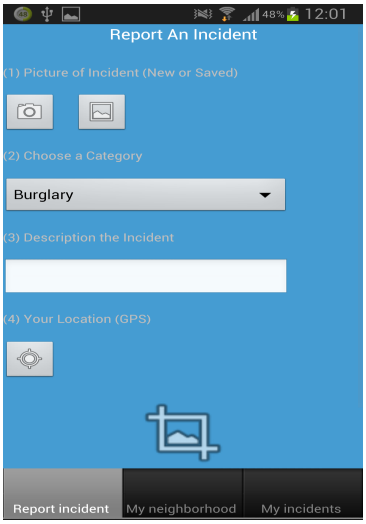


Figure 22: Report incident page

The ethos of reporting page is to provide the user with a quick and easy method of reporting an incident. This gave me a clear vision on how to design the page with aid of a design section and Android platform requirements. The simple XML layout which Android offers was used to place each of the tasks on the page [Figure 22]. A problem arose when interacting with inbuilt phone cameras. The captured photo was not being sent back to the open reporting page of the application, and after many attempts I discovered that the image was being deleted to free memory, as the reporting page was a tab activity. Therefore every time a picture was captured it was going back to the first tab in the list, “my neighbourhood”, which meant that the system would return to the “my neighbourhood” tab resetting the supplied incident information. To overcome this problem I used “activitygroup” to deal with multiple activities on a single tab.

Incident Map / Comment activity

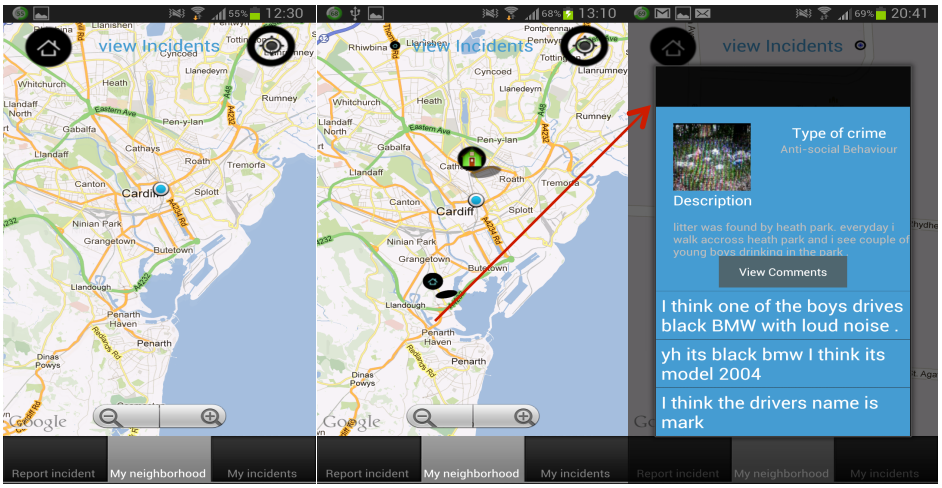


Figure 23: Incident Map and comment activity

The map page was designed to offer users the option to search for incidents in the home location that they registered with or in their current GPS location [Figure 23]. This function required Google Map API imports. However, Android developers have to register their personal information with Google API in order to access Google Maps. This was challenging process as I wasn’t familiar with Google Maps and did not know how to activate the map. Keystore was needed in order to obtain an API key. This process is described in the Android map tutorial [Appendix 1]. To use the API key, Android provide a simple XML layout to display Google Maps. This can be seen on [Figure 24].

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
<com.google.android.maps.MapView
        android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:progress="15"
        android:apiKey="0HaL4YXG-4MQEf9f56xBv_q1vD9LAOsyoqWoD8g"/>
</FrameLayout>
</LinearLayout>
```

Figure 24: Integrating Google map with valid API Key

Once Google Maps was displayed on the page, the next task was to create a marker on the map to indicate user location. This marker required an overlay, which is a transparent layer on top of the map. This overlay would then be the foundation upon which the marker would be created. To determine user location I used the predefined Android class "MyLocationOverlay", which obtains the user location using GPS. In " drawMylocation", a function with the "MyLocationOverlay" class, one feature had to be implemented into the class to obtain the latitude and longitude of the location and convert it into Geopoints, which can be stored in the database. This was carried out using math's formulae [Figure 25].

```
private GeoPoint getPoint(double lat, double lon) {
        return(new GeoPoint((int)(lat*1000000.0), (int)(lon*1000000.0)));
}
```

Figure 25: Converting Geopoints

To convert geo coordinates to get an address, the Google Maps API provides a function which obtains the address providing latitude and longitude. Using "getMaxAddressLineIndex" obtains the full address including postcode. To gather only the required information, Android provides a function for the UK which obtains the postcode from the given location "getPostalCode()". This can be seen in [Figure 26].

```
Geocoder geoCoder = new Geocoder(getBaseContext(), Locale.getDefault());
        try {
        List<Address> addresses = geoCoder.getFromLocation(location.getLatitude(),location.getLongitude(), 1);
                if (addresses.size() > 0) {
                        add = "";
                        for (int i = 0; i < addresses.get(0).getMaxAddressLineIndex(); i++)
                                add += addresses.get(0).getAddressLine(i) + "\n";
                }
                postcode=addresses.get(0).getPostalCode();
```

Figure 26: Get address from GPS location

To create multiple pins on the map to represent incidents , I used a list which creates an overlay every time a new incident gets reported using "List<Overlay> mapOverlays" to create multiple pins. The next challenge was to make each incident stand out on the map. This was done by adding an icon to each overlay, which can be seen in [Figure 27]. To free up memory and unwanted overlays I used "createMap()", which overrides the map and starts the layout afresh.

```
Drawable drawablehome = this.getResources().getDrawable(R.drawable.home2);
itemizedoverlayhome = new MyMapMarker(drawablehome,this);
```

Figure 27: Adding an overlay to Google Maps

The user will have the option to view the incidents in greater detail by selecting the marker/pin on the map. To achieve this interaction between the user and map, "onTouchEvent" was used. The final stage was to present the incident information. This is done using "Dialog", which is created on the fly and displays information on the particular incident in a clear format XML layout. This can be seen in [Figure 28].

```
final Dialog dialog = new Dialog(mContext);
dialog.setContentView(R.layout.vcrimethree);
TextView text = (TextView) dialog.findViewById(R.id.textView2);
```

Figure 28: Adding dialog to every incident on the map

User profile activity – My incidents page

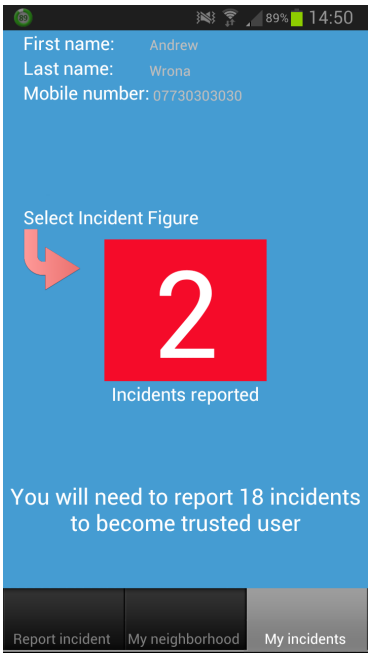


Figure 29: User profile activity

The user profile offers logged in users access to their personal information in a clear manner. They can also view how many incidents they have reported and see how many more incidents they need to report to become trusted user. This was very impotent as many apps on the Android market have a profile page which gives information about their activities on the app. A useful feature implemented into the user profile page was the ability to access incidents the user has reported. This can improve interaction between the community group and the police as the user can add more information about an incident they previously reported [Figure 29].

6.2. Website implementation

The website began with a login page which retrieved information from the database, like the application. The next stage was to create a live incident page to give police officers quick access to incidents that app users reported in the area that the officer is responsible for. Further to the incident page was the map page, which displays all the incidents that have been reported via the app. The map function will offer a clear indication of where the incident was reported.

To make the website function on many browsers, I looked into Bootstrap, which offers simple HTML and CSS layouts for developers. To offer a wider range of browser access, I decided to use HTML5 and preset CSS layouts. Bootstrap can be used on any HTML file by imparting extensions needed before using the methods. A tutorial on using Bootstrap can be seen in [4].

Login page

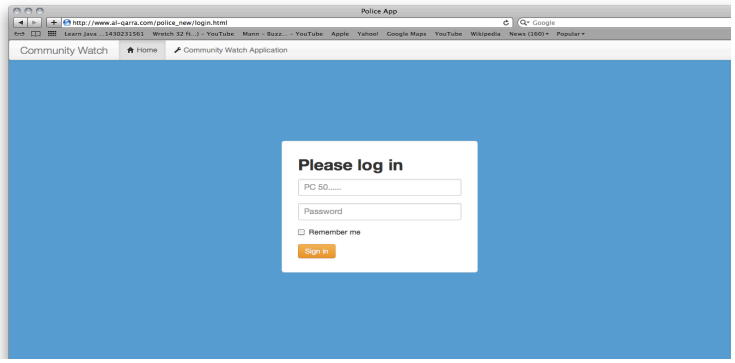


Figure 30: Login page webpage

The login page will request a Pc number and a password. A Pc number is given to a police officer when they first join the police force, and the password is set by the administrator directly into the database. If a police officer successfully logs into the system then they will have access to all of the incident information reported by conventional users of the app. Simple Bootstrap methods were used to create the login layout [Figure 30].

Live incident feed

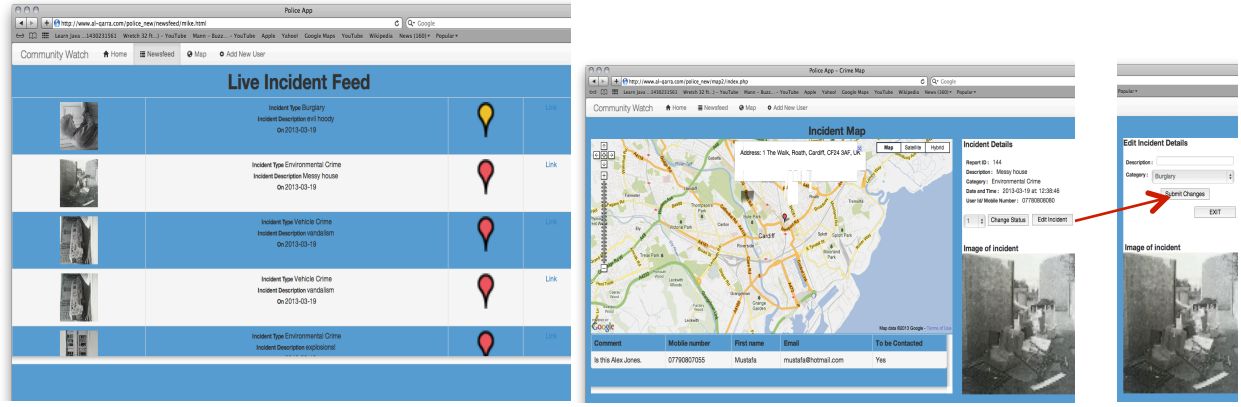


Figure 31: Live incident feed webpage

When a user logs in successfully they will then be directed to the “live incident feed”. T his will show all incidents within the area that the particular officer is responsible for. This will help police officers to deal with reported incidents in quicker manner, as the system will update with new incidents. To present the gathered information relating to the specific area I used the list method which Bootstrap offers. This gave me the flexibility to present an image, description and incident type on the list in a clear manner. To provide more accessibility to police officers I generated a link for every incident on the page. The user can click on this link to view the particular incident in greater detail. This also this gives users access to the change investigation level function, with which they can update the community by changing the status. The user can also change the information provided about the incident if they wish to [Figure 31].

Map

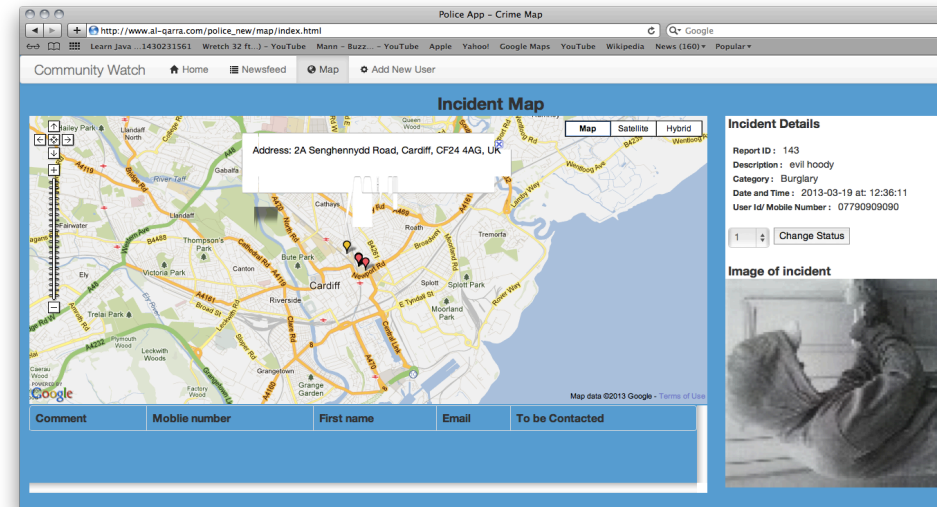


Figure 32: Incident Map webpage

PHP script was used to retrieve each incident from the database outputted in an XML format. Using tools provided by the Google Map API, the XML was parsed and the information shown appropriately on the page. A tool known as "GDownloadURL" runs the PHP script and reads the information retrieved. Tags are given to each row in the database, <Marker />. The Google function reads these tags and saves the information for each incident. Once the map page is selected by the user, multiple markers are created on the map, Each marker on the map is categorised via a colour, where each colour corresponds to a type of incident level, from emergency to low level. When a user clicks on the incidents icon, information about the reported incidents is presented in simple HTML format. Officers also have access to all comments relating to the selected incident [Figure 32].

Add new users

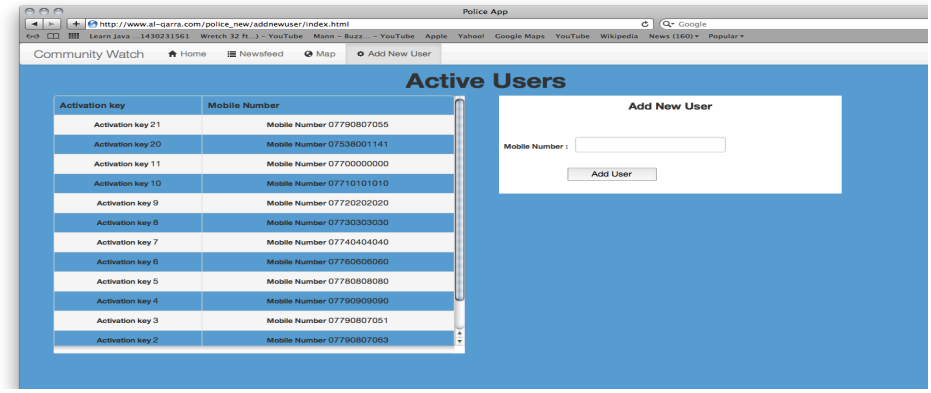


Figure 33: Add new users webpage to trusted network

To minimize the occurrence of bogus reports on the system, a trusted network was implemented. This required any user wishing to register to the system to be granted access by an existing police officer. A simple HTML and PHP script was used to register new users' phone number, which would also be used if the current user forgot or lost their registration key, as they could contact the police station to retrieve it [Figure 33].

6.3. Database Implementation

The database was located remotely using the hosting site, One.com. This site was chosen as it uses a MySQL database, which is a very popular relational database used in Android programming. The Android API`` offers many functions for connecting and retrieving data from a remote MySQL database. The database was set up using phpMyAdmin – a user-friendly frontend used to create MySQL tables. For the application to read and write into the database, many different PHP scripts had to be implemented and placed on the server to call the PHP scripts in order to maintain connection between the database and the phone (client) application.

To connect to a database using PHP, the following command was used. MySQL_Connection has to be placed inside any PHP scripts which need access to the remote database [Figure 34].

```
mysql_connect("hostsite.com.mysql", "Username", "Password") or die(mysql_error()); mysql_select_db("databasename")
or die(mysql_error());
```

Figure 34: Connecting to one.com database server

The username and password were given by the service provide. In this case it was “one.com”. For security, one.com requires a username and password to access the database. This stops intruders running any queries or inserting new information into tables.

The values that were sent from the application are sent to a PHP script; the PHP script will gather the information with a given variable name i.e. $mobile = 07790807055. These were retrieved using a predefined method which PHP offers called “$_POST”. This method is used to collect values from a post method used within the application [Figure 35].

```
$mobile = $_POST['mobile_number'];
$first = $_POST['first_name'];
$last = $_POST['last_name'];
$email = $_POST['email_address'];
$dob = $_POST['date_of_birth'];
$gender = $_POST['gender'];
$pass = $_POST['password'];
$contact = $_POST['contact_me'];
$address = $_POST['address'];
$postcode = $_POST['postcode'];
$activation = $_POST['activation'];
```

Figure 35: Collect information from app using POST method

Gathered information from the user gets inserted into a table with respect to the field names i.e. “INSERT INTO appregistration (MOBILE_NUMBER)”. Given data will be stored in a table named “appregistration”. The field the script will be interested in is “MOBILE_NUMBER”. To insert the value “VALUES('$mobile')” .This takes the value from the given mobile number and saves it in the table under “MOBILE_NUMBER” filed [Figure 36].

```
mysql_query("INSERT INTO appregistration (MOBILE_NUMBER, FIRST_NAMES, LAST_NAMES, EMAIL, DATE_OF_BIRTH,
GENDER, PASSWORD, CONTACT_ME, ADDRESS, POSTCODE,DATE_OF_REGISTRATION)

VALUES('$mobile', '$first', '$last', '$email', '$dob', '$gender', '$pass', '$contact', '$address', '$postcode',  CURDATE())")
```

Figure 36: Insert a row of information into the table "appregistration" with the given values

The next stage was to create a user login script. This involved sending a mobile number and password to verify the information with the login table in the database. If the mobile number and password were identical to the row on the database, this would return true and give the user access to the app. If the mobile number or password was not correct, then it would send a message to the user alerting them they have problem with login.

Many other PHP scripts were created in order to meet the requirements. One of the most important scripts was for reporting an incident. This involved sending plain text, an image and geographical coordinates from GPS. The image was a problem as it was too large to save in the database. To overcome this issue I resized each image sent by the user, which lead me to looking into ways to convert images to string, as plain images cannot be saved to a database directly. With further research I managed to find a free open source package that programmers use to convert images to string. Base64 string converter was used as it can convert image files into string to be stored on the database. The steps that need to be taken when sending an image are as follows:

1. Save image on user phone
2. Get the image and convert each pixel of the image into byte
3. Gather all the bytes in byte array
4. Encode the byte array into a Base64 format.
5. Output of the encoder will be converted into string.

The code that was implemented to carry out this task was Base64.java. This is a class released by Robert Harder to the public with the main function of encoding and decoding image files using a Base64 code. The image would save in the database in blob format as this can store large string files [Figure 37,38].

147	07740404040	51.4846563	-3.1705208	Environmental Crime	explosions!	1 The Walk Roath, Cardiff CF24 3AF	CF24		[BLOB - 4.4 KiB]		1	2013-03-19	13:15:33

Figure 37: Example of blob image file saved in database

```
BitmapDrawable bmp = new BitmapDrawable(is);
Bitmap bitmap = bmp.getBitmap()

// Predefine image size
int width = bitmap.getWidth();
int height = bitmap.getHeight();
float scaleWidth = ((float) 90) / width;
float scaleHeight = ((float) 90) / height;

// Create a matrix for the manipulation
Matrix matrix = new Matrix();
```

```
// Resize the bit map
matrix.postScale(scaleWidth, scaleHeight);

// Recreate the new Bitmap with given size
Bitmap resizedBitmap = Bitmap.createBitmap(bitmap, 0, 0, width, height, matrix, false);

// To get the resized image send to the database first need to create byte array and identify the image type
ByteArrayOutputStream stream = new ByteArrayOutputStream();
resizedBitmap.compress(Bitmap.CompressFormat.JPEG, 90, stream);

byte[] ba = stream.toByteArray();
//sends to class base64
savedpic = Base64.encodeBytes(ba);
one.setVisibility(View.GONE);
image.setImageBitmap(resizedBitmap);
```

Figure 38: Code used to resize and convert given image

To view comments on a particular incident, it was first necessary to find the correct record within the AppReports table. This was achieved by comparing the incident the user clicked on with the incident id on the database. Once the incident was found, information was presented to the user in a clear layout using the Dialog function. This made the process of commenting on incidents easier as a simple textbox was used to gather user comments and uploaded them to the AppComments table. Submitted comments were stored as a new row in the AppComments table and linked to their associated incident using that incidents unique ID. To retrieve the comments that were previously submitted to a selected incident, this was accomplished by using the incident ID to compare with the AppComments table to see which comments are associated with the incident number. To present the information in a clear layout, SlidingDrawer was used, as when the user slides the list it will automatically refresh the comments to give live comment alerts to the user.

In order to retrieve information and present it to the user on the Android platform, JSON (JavaScript Object Notation) was used. When a PHP script was used to get information from the database it was encoded in JSON format. To retrieve the information from JSON, BufferedReader was used to scan all the lines of text in order to convert the information from JSON format into readable text with given tags for each row of information. The tags were used to identify what each piece of information relates to i.e. commentss.add(json_data.getString("COMMENT"));". Each comment from the PHP script gets stored on the comments array on the application [Figure 39].

```
JSONArray jArray = new JSONArray(result);
JSONObject json_data = null;
ArrayList<String> commentss = new ArrayList<String>();

for (int i = 0; i < jArray.length(); i++) {
        json_data = jArray.getJSONObject(i);
    commentss.add(json_data.getString("COMMENT"));

        }
```

Figure 39: Parse data and filter information needed from the PHP script

From the knowledge gathered I created an incident map page. This required many different PHP scripts to retrieve all the coordinates of incidents that were posted. This was implemented on top of the map library provided by the Google Maps API for developers. Multiple overlayed markers were created to represent incidents on the map, and action listeners were implemented with each marker to allow users to retrieve detailed information about a particular incident.

To satisfy the requirements, I had to look into a way of grouping lists of incidents which are close to the user's location. The first method I looked into was matrix calculations to find the distance between two locations using the Google Maps method "GetDistance". This was intended to be used to find incidents reported close to the user's location. However, the problem with using radio overlays and matrix algorithms is that every time the map loads it looks at every possible location on the map to find the closest incidents, which means all the phone's memory was getting used and crashing the device.

The second method was gathered from the UK postcode system. This led me to study postcode formats using the Royal Mail website [5]. Studying the standard system Royal Mail use to generate postcodes, as seen in [Figure 40], led me to implement the idea into the app. For example, a simple postcode has two characters then three digits, and ends with two characters. To simplify this for the app I will only be using the first two characters and two digits- this gives me a location and street, but this will not pinpoint the house number as this is not needed to compare incidents on the map.

CF83 3BY
CF = which office processes the mail
83 = which office delivers the post
3 = which location area
BY = which posted group to deliver

Figure 40: UK Postcode outline

As Google API allows developers to generate addresses with given coordinates, this gave me the idea of saving the incident address i.e. postcode on the database when the incident is reported, as seen in [Figure 41]. When the user views the map on the app, a small PHP script will retrieve the user's coordination postcode and compare it with a list of incidents on the database. If the incident postcode matches the user postcode, this will create markers on the map. Finally, this will allow the user to select on any of the markers to view a particular incident in detail.

	147	07740404040	51.4846563	-3.1705208	Environmental Crime	explosions!	1 The Walk Roath, Cardiff CF24 3AF	CF24	[BLOB - 4.4 KiB]	1	2013-03-19	13:15:33

Figure 41: Generate address with given coordinates

6.3.1. Website Implementation with database connection

The website interface was designed to allow police officers to retrieve information from the app. The main features of the website are a login page, live incident feed, map and an add new users to the Community Watch app function. To begin the implementation of the website I started a login page with the aid of Bootstrap layouts. To make the page interact with the database I used a PHP script like the application, a simpler PHP script was used to retrieve incidents from the database, but as the information relating to the incidents was being output in XML format, Google API was not accepting it. Using a function which Google API offers called "GDownloadURL" made it possible to parse and present XML information on the map. This will be elaborated on below.

Login page activity

The login page was developed with the same design structure. It was developed with the aid of Bootstrap CSS and a simple PHP script to cross-reference with the information given by the user to the database. When valid user details are given - the police officer would be transferred to “live incident feed” to view the incidents reported around the area that the police officers is responsible for. To limit the number of incidents viewed by each police officer, a postcode system was used. This means that when a police officer accesses the website they will be given a postcode to represent the area they will be in charge of. This postcode gets stored in the police_login table inside the database. This table is used to verify user details and gets the postcode of the area the police officer is responsible for. This postcode then gets used when viewing the list of incidents on the “live incident feeds” page.

Live incident feed activity

Live incident feed was initially was designed to alert police about incidents reported around the area that they are responsible for. This was achieved using a predefined list function which Bootstrap offers, as can be seen in [Figure 42]. To populate the list with incident information, many PHP scripts were used. The first PHP was use to obtain incident details, the second was used to obtain incident images, and finally, another PHP script was used to bind related information about each incident together; this gives the user a link which they can follow to access detailed information about the incident. To present detailed information about a particular incident, the user is transferred to the map page, which gets used to pinpoint the location of the incident; details of the incident are presented on the side of the page, as seen in [Figure 42].

```
<div class="table-section">
    <table class="table table-striped table-bordered">
        <thead >
            <tr>
            <th>Image</th>
            <th>Incident Description</th>
            <th>Status</th>
            <th>Link</th>
            </tr>
        </thead>
      <tbody id="newsfeed_table_body">
</tbody>
</table>
```

Figure 42: Bootstrap table function

Map activity

The map was designed to display posted incidents. This was achieved using the Google Map API and PHP scripts. Google maps offers a tool called GDownloadURL, which parses and displays markers on the map according to the given XML information. To create XML format information, a PHP script was used to retrieve information from the database and output the related information in clear XML format. The location of each incident was stored using two fields in the database: LocationGpsLon, and locationGpsLat. The PHP script creates a marker for each incident, as seen in [Figure 43]. Once the map is populated with incident markers, each marker on the map gets categorised via a colour. Each colour corresponds to a type of incident level, from emergency to low level incident, to alert the police which incidents on the map need the most urgent attention. To provide accessibility to the police, they can click on the marker to retrieve detailed information about the incident and comments from app users in list format.

Additional features were developed to give authority to police officers to change the level of incident and change the content of the description of the incident. This will reduce the risk of offensive language and people misusing the app.

```
$node = $doc->createElement("marker");

$newnode = $parnode->appendChild($node);
$newnode->setAttribute("investigation", $row['INVESTIGATION']);
$newnode->setAttribute("address", $row['ADDRESS']);
$newnode->setAttribute("postcode", $row['POSTCODE']);
$newnode->setAttribute("lat", $row['locationGpsLat']);
$newnode->setAttribute("lng", $row['locationGpsLon']);
$newnode->setAttribute("category", $row['TYPE_OF_INCIDENT']);
$newnode->setAttribute("date", $row['DATE']);
$newnode->setAttribute("time", $row['TIME']);
$newnode->setAttribute("description", $row['DESCRIPTION']);
$newnode->setAttribute("reportno", $row['INCIDENT_ID']);
$newnode->setAttribute("userid", $row['MOBILE_NUMBER']);
$newnode->setAttribute("image", $row['IMAGE']);
```

Figure 43: PHP script to create marker

Add new users activity

The add new users to the Community Watch app function was developed to keep possible bogus reports away from real reports. This was achieved using a simple PHP script which retrieves a list of all the active users on the app from the database. Furthermore, the police have the ability to add new users to the system by inputting a user's mobile number. When a user's number gets submitted to the system they will be given a randomly generated “activation number”, which they can then use to register with the community watch app.

6.3.2. Application implementation with database connection

Android offers a function to developers that allows access to an online database in the most secure and flexible way, the method is called HTTP. This gives Android apps the ability to pass information from the phone to the database. In order to add information the mobile user needs access to the Internet, and in order to gather passed information from the phone a PHP was required [13,15].

The first step was to create a HTTP Client function in order to transmit and receive HTTP messages. HTTPPost was used to define the correct PHP script to run. The PHP scripts were all stored within the host space purchased from One.com. In the example below, the user_login.php file was run in order to verify a user’s details [Figure 44].

```
HttpClient httpclient = new DefaultHttpClient();
HttpPost httppost = new HttpPost("http://hostsite/police_app/ user_login.php");
```

Figure 44: Use HTTPClient to call PHP script

In order to pass values to the PHP script, BasicNameValuePair was used to save the details of the user. In this case it was a mobile number and hash value password. To send more than one instance of nameValuePairs, an array list was used to group many instances together when passing values to the PHP script. The following snip code illustrates how the given values are sent to the database using HTTPPost [Figure 45].

In the following example, the user enters the following information on login to the app and select the login button:

- Mobile number :07790807055
- Password: Mustafa123

```
ArrayList<NameValuePair> nameValuePairs = new
ArrayList<NameValuePair>();
                    nameValuePairs.add(new BasicNameValuePair("mobile",mobilenumber));
                    nameValuePairs.add(new BasicNameValuePair("password",password));

                    httppost.setEntity(new UrlEncodedFormEntity(nameValuePairs));
                    HttpResponse response = httpclient.execute(httppost);

String str = getInfoFromWEBToString(response.getEntity().getContent()).toString();
          if (str.toString().equalsIgnoreCase("1")) {

              startActivity(new Intent(MainLogin.this, infoupdate.class));
          } else {
                    Toast.makeText(getBaseContext(),"Please check your mobile number and password .",
Toast.LENGTH_LONG).show();

          }
```

Figure 45: Use HTTPPost to connect between the app and database

"httppost.setEntity" is a method that takes a URL as an argument and details retrieved data from the given URL. The most interesting feature of HTTPPost is the UrlEncodedFormEntity, which transforms the given data values into an understandable format so that the server can communicate with the app. Finally, HttpResponse is used to send gathered information to the PHP script in order perform a query which check s whether the given information via the user corresponds to the database login information.
This code was used on the app in Mainlogin.java class to open a connection between the app and the database. If the correct login details are entered, the user will be given full access to the app i.e. view incidents, report incidents and view user profile.

All the above codes used to implement the application and the website can be found on [Appendix 8].

6.4. Important Features

6.4.1. Encrypting Passwords

During the research it was clear that saving passwords in a plaintext format compromised the security of the system. There are a couple of reasons for this:

- The first issue was that hackers could break into the database and have clear access to passwords of users. These passwords may be used for criminal activates i.e. the same password could have been used across other websites, such as Hotmail or Facebook, for the same user, therefore exposing other accounts of that user

- The second threat would be that insiders, i.e. police officers or database managers that use the database, could see the passwords of any user. This may lead to the misuse of user accounts and the reporting of bogus incidents on the app. Due to potential risks such as these, no member of the public would approve of their passwords being saved in plaintext.

Many situations have arisen where reputable establishments have not taken the necessary precautions to secure user passwords appropriately and comply with data protection acts.

Twitter was recently 'hacked' using some simple algorithms to access the network [6,7]. It was found that over a 250,000 passwords were stored in plaintext along with other user information. This was ready and waiting for hackers to capture to use for criminal activities. To avoid situations like this, a number of steps were taken to keep information secure. A detailed description follows.

4.4.1.1. Cryptographic hash function on user passwords

Instead of storing passwords as they are typed by the user, a cryptographic hash function is used to store the hash value instead inside the database.

A cryptographic hash function takes in an input, i.e. file or string, and returns a hash value string. It's mathematically infeasible that a cryptographic hash function would return the same output for two different input values and it's impossible to obtain the inputs from the outputted hash value. .

In order to check if a user password is correct, the given password is hashed from the login page on the app, then compared with the password on the AppRegistration table on the database using a simple PHP script. This means that the password given by the user never needs to be stored in its original form, and a hacker will not be able to login unless they know the password or know a string that results in the same hash value. However, it is impossible to generate the same hash value [8,15].

One problem with this approach is that it can be attacked by using a hack called 'rainbow tables'. A rainbow table is a look up table, with which you can look up the value of the hash and find the original string. A rainbow table is created with many entries to generate random hash values. Hash functions would be fast to execute on a rainbow table, even with billions of entries inside the table [9]. This means short passwords are cracked easily with aid of rainbow tables.

To lower the risk of hackers using rainbow tables to encrypt user passwords, a random string will be used to create a more complex hash value. The term for using a random string when generating a hash value is called 'salt'. In the application, the salt method is string set as "policeapp289ma". The snip code below was used in the MainLogin and MainReg class to convert the user's password into a SHA1 format to compare with the database and save in the database respectively [Figure 46].

```
private String name = "policeapp289ma";
        private String encryptPass(String pass) throws NoSuchAlgorithmException {

                MessageDigest pass = MessageDigest.getInstance("SHA1");
                pass.update(pass.getBytes());
                pass.update(name.getBytes());
                byte[] passwordBytes = pass.digest();
                passwordenc = Arrays.toString(passwordBytes);
                return passwordenc;
}
```

Figure 46: Encrypting password using salt

The above code first obtains the user password, combines the password with the set string value which represents salt, then the sha1 algorithm gets used to give an alphanumerical string. To finalise the process, the given string gets saved within a byte[] array as an output value and gets converted to string "passwordenc".

6.5. Main Problems

6.5.1. Application

Storing user information when they exit the app was very difficult as there are many way of performing this type of function i.e. “SavedPreference”. This involved storing information on the app memory, which meant when the user exits the app, their login details would be deleted. To overcome this problem I researched Android memory manger in order to find out more about saving user login details even if the app has been closed. Remember me checkbox was used to gather login details of the user and save the login details in “FileOutputStream”, which is onboard memory storage location which can be used to store small data size, i.e. mobile number and password. This can be seen in [Figure 47].

```
FileOutputStream phoneinfo;
        try {
                String userandpass=username+"£"+ content;
                phoneinfo = openFileOutput(userandpass, Context.MODE_PRIVATE);
                        phoneinfo.write(userandpass.getBytes());
                        phoneinfo.close();
```

Figure 47: Storing user information using SavedPreference

The incident map proved to be one of the trickiest tasks to implement. The map was initially designed to be populated with incidents in order for the user to interact and view them in detail. This required the use of a number of Android features in order to satisfy the requirements. The incident map was simple to create. It required a Google Map API key, shown previously, and to make it easier for users, a zoom function was implemented.

As explained persistently in the implementation section, to create incident markers on the map required overlays to be created for each incident. Overlays are a transparent layer on top of the map which is used to position the markers. Once all the required incident information is gathered from the database, markers are created inside the overlays. The next phase was to make the markers visible and add action listeners to each marker to allow users to fully explore the contents of a particular incident. To display incident information, a function called dialog was used. The implementation of the tasks required for viewing incidents on a map was harder to code than I anticipated. It also took a fair amount of time to experiment and explore Google Map API in order to place multiple markers on a map.

Saving incident images proved to be a difficult task. As previously discussed, there were a number of ways that an image could have been saved, i.e. Java database connectivity – which involves a Java based data connection program to gather the image and save the image directory in independent location to the information of the incident report, this would lead to an extra connection within the database to link the image with the incident via a unique ID. This was an inefficient method to use on this project. This led me to implement Base64 encoding into the application in order to save images. This was a more appropriate choice as it meant all the information the police would need would be found in one place. Another advantage of Base64 is open source, which allows Android developers to update algorithms to speed up the process of converting data to string format, which means app users don’t need to wait for so long to send their reports to the database.

6.5.2. Website

The aim of developing this website was to present and investigate reported incidents via the Community Watch app. A set of requirements was produced with the aid of police and community group interviews. To start the development of the website, I decided to use the latest technology (HTML5 with Bootstrap CSS frameworks). Experimenting with different frameworks and determining which technologies to utilise was difficult due to my inexperience . The time constraints of the project made achieving the requirements difficult. To understand the functions and CSS style sheets with Bootstrap led me to experiment with libraries to create basic functions which could be used on the website. Tutorials can be found on [4].

7. Testing

7.1. Software verification

This section will look at the testing of the mobile phone app, which was developed to meet the community and police service requirements. As the community watch app was developed using the Android platform, the process of testing the application was very difficult, as Android tends to focus on simple, intuitive and easy to follow industry-accepted conventions. This makes the testing process focus on the usability, input methods, screen sizes, network and processing speeds.

Android API guides developers to use a predesigned testing process. One of the most popular usability testing processes was Jakob Nielsen ten heuristic principles, which are widely used in the software industry as logical guidelines to evaluate user interfaces for Android application and websites [10].

The ten heuristic principles stated below are the original writings of Jakob Nielsen. We will use each heuristic to test the application as a whole [11].

Visibility of system status

The system should always keep users informed about what is going on through appropriate feedback within a reasonable time.

The application offers many instances where the user would be informed of what is going on in the background; this was achieved by using a method called Toast, which is an android notification box that alerts users with a message on the screen.

- When a user registers, they will see a message on the screen to confirm they have registered successfully [Figure 48]

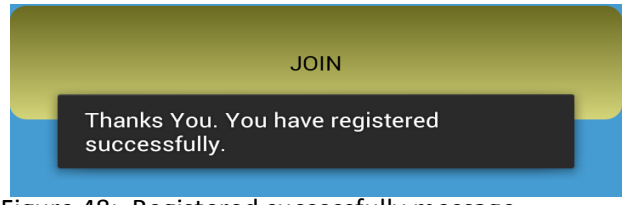


Figure 48: Registered successfully message

- If a user enters incorrect login details, they will be presented with a message stating that the given details are incorrect [Figure 49]

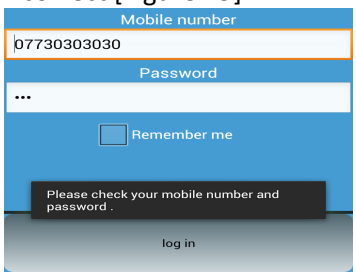


Figure 49: Incorrect login details message

- When a user reports an incident, they will be presented with a message to confirm that their report has been sent successfully [Figure 50]. If the internet connection is slow, a progress-bar will be displayed in order to inform the user that the process of reporting incident is still ongoing [Figure 51]

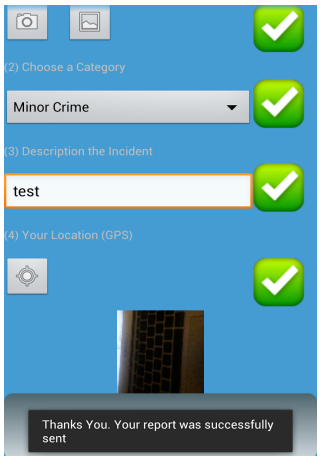


Figure 50: Report send message

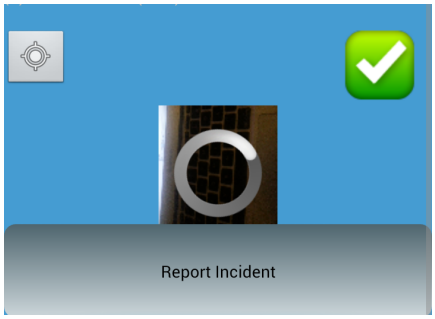


Figure 51: Ongoing progress-bar

- When a user views incidents on the"My neighbourhood" map , they will be presented with a message informing them to wait until the user's GPS has been located[Figure 52]

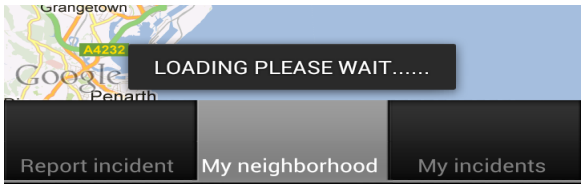


Figure 52: GPS finder message

Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Following real-world conventions and making information appear in a natural and logical order.

The application is aimed at all users with an Android phone. The language used throughout is of a very basic and easy to read standard. In such places as tutorial pages, the user is presented with a snapshot of the tasks the application offers, this gives an extra sense of user friendliness to the content of the page and clear text is used to explain the given task. The tutorial page was designed to give a clear view of the application and to explain what the application offers to users, i.e. community groups [Figure 53].

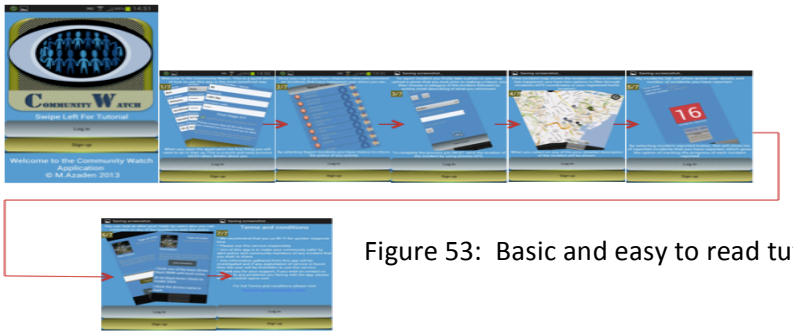


Figure 53: Basic and easy to read tutorial page

User control and freedom

Users often choose system functions by mistake, and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

This principle ensures that the application offers easy navigation to exit a path that they did not intend to proceed on. As the application was developed with strict Android guidelines from the start of implementation, this principle was considered throughout the application. At any point of the application it will be simple for the user to return to where they were previously by simply clicking the Back button, which can be found on every Android mobile phone.

To login takes two paths.

Login button - on main page ->
Login button – on the login page.

To register takes two paths.

Signup button - on main page ->
Join button – on the registration page.

To view a list of incidents around the user's area requires no activity by the user as it gets presented automatically as a form of notification.

To report an incident takes six paths.

Report incident button - on incident notification page ->
Attach picture to incident on report incident page ->
Choose a category for the incident on report incident page ->
Type description to incident on report incident page ->
Attach GPS location to incident on report incident page ->
Report incident button - on incident page.

To view an incident on Google map "My neighborhood", the user needs to select one path to view incidents around their GPS location from the reported incident page. If they wish to view incidents around their registered home location, then they need to select the "home" icon on the map.

To view the user's profile page the user needs to select one path from the "Report incident" or "My neighborhood map".

Finally, to view any incident the user needs to select the incident on the map or on the list. This is done using four paths.

Select on the incident ->
View incident details ->
Select on comment textbox ->
Type comment ->
Select on post comment button (when a user posts a comment they will be presented with list of comments made by active users of the application).

The longest feature on the application is reporting incident, as it takes six paths, which is understandable, as it needs user interaction. With the aid of a simple user interface the process was presented in a clear and professional manner to make the task easy to navigate through.

Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow Android platform conventions.

This principle has been followed within the community watch app, for example each option on the main menu page uses the same words and terminology. Each task on the application has unique identifying nametags to easily explain the task in a clear manner in order to have consistent flow throughout the application. This can be illustrated with the login page, which presents two required fields "mobile number and password". When the user fills in the required information they will then need to click the "login" button in order to access the application contents [Figure 54]. This gives the user clear flow pattern, as the "login" nametag is used on two different pages to prevent confusion or the misleading of users. Another way of satisfying the consistency on the app is presented on the "reporting an incident" page, which gives a step-by-step guide to reporting an incident. This gives the user clearly labeled tasks to carry out. In order to report an incident the user only needs to click on the "report incident" button when logged in to the application [Figure 55].

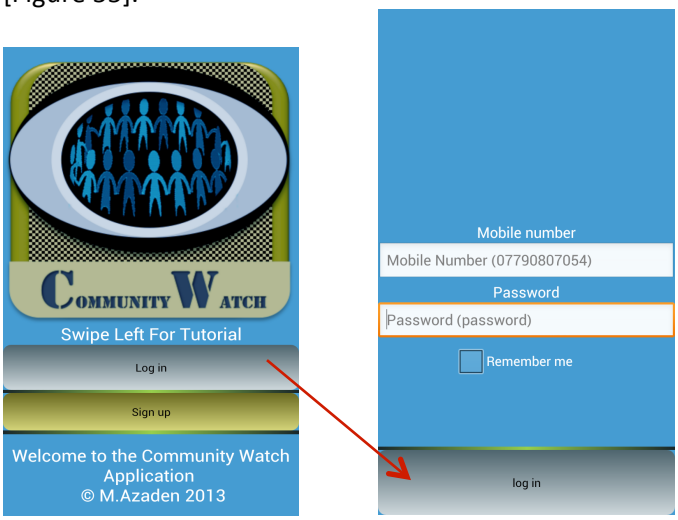


Figure 54: login button consistency

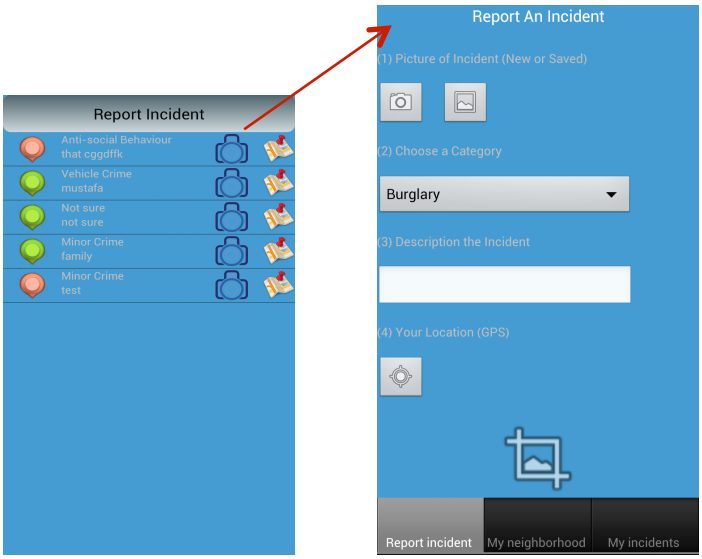


Figure 55: Report incident button standards

Error prevention

Better than a good error message is a careful design which prevents a problem from occurring in the first place. This will either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Error checking was taken into consideration when implementing the application in order to stop users misusing the application and wasting police time with false incident reports. Verification checks were put in place to ensure that users have given the required details.

Error prevention occurs within the registration page. Every user must have a valid activation key and an unregistered mobile number in order to register successfully. If the user does not have an existing mobile number or has a mobile number that does not have activation key, the user will not be registered. The reason for using an activation key is to ensure that a trusted network is in place in order to gather valid and trusted incident reports. When the user enters an incorrect activation key or existing mobile number they will be presented with a general error message [Figure 56].

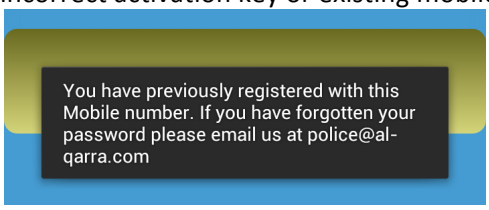


Figure 56: Error prevention message

When reporting an incident, the application requires the user to send a photo, description, category and the location of the incident. When each of the requirements is complete an icon will be displayed on the side of the task in order to alert the user to what tasks are needed to complete the incident report. If a user were to forget to write a description of the incident or forget to attach a GPS location, a Toast would appear to inform the user that they have not entered a description. If a user was to report an incident with no GPS location, the police would have hard time finding out where the incident took place. Using features like location verification will reduce the time needed to investigate incidents reported by application users [Figure 57].

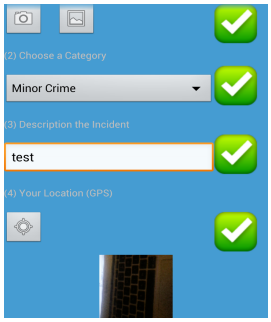


Figure 57: Error prevention check icon

A confirmation dialog box was implemented to reduce errors made by users when reporting an incident. If a user selects "YES", then the information given by the user will be sent to the database as a form of incident report [Figure 58].

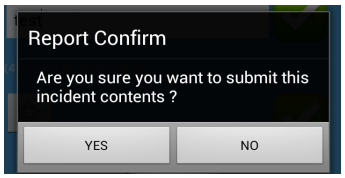


Figure 58: Error prevention confirmation

Recognition rather than recall

Minimise the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

The user does not need to remember each page on the application. For example, the tutorial page consists of a number of pages which the user can navigate using a swipe action, and each page on the screen has a page number to prevent confusion [Figure 60]. Another example would be that the "my neighbourhoods" map has simple instructions, i.e. icons representing the user's home or GPS location. The user may want to view incidents in the current GPS location or the user's home location.[Figure 59].

Figure 59: Two simple icons to represent the tasks available on Map activity

Figure 60: Tutorial page with page numbering

Flexibility and efficiency of use

Accelerators - unseen by the novice user -may often speed up the interaction for the expert user so that the system can cater to both inexperienced and experienced users. It also allows users to tailor frequent actions.

A "Remember Me" checkbox function was implemented to speed up interaction for active users. This allows a user to save their login details so the next time they use the application their login details are saved, which makes the application user friendly as the user does not need to renter their login details in order to use the application. As the application needs a quick response time, this function saves the user from typing unnecessary details [Figure 61].

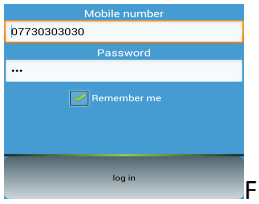


Figure 61: Remember me check box

Aesthetic and minimalist design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

To meet the main ethos of the application, which is to offer a fast and efficient way to report and view incidents for any active users, the design of the application only incorporates the standard layouts which Android API offers to developers. By adhering to Android's strict development structure the application performed with low processor demands and less memory leaks. For example, when attaching a picture to an incident report the picture gets resized and compressed in order to allow quick transaction between the database and the application, as explained in the implementation section. Another example would be when viewing incidents on Google Maps via GPS, the coordinates get converted to address format in order to gather related incidents in the given address using the postcode filed name to speed up the process and reduce irrelevant transactions between the database and the application [Figure 62].

	147	07740404040	51.4846563	-3.1705208	Environmental Crime	explosions!	1 The Walk Roath, Cardiff CF24 3AF	CF24	[BLOB - 4.4 KiB]	1	2013-03-19	13:15:33

Figure 62: Illustrates the contents saved on the database relating to the reported incident with given address and postcode.

Help users recognise, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

The toast function, which is a form of error message on the Android platform, was used throughout the application to alert the user to problems. For example, if the user does not enter their existing mobile number when registering as new user on the application, a Toast message will inform the user that they have provided incorrect information [Figure 63]. Another example would be when the user wishes to view incidents via map and they have weak signal, in this situation they will be prompted with Toast message stating: "Sorry - there was an issue with your Internet signal, try again later". This message will alert the user that the task can't be performed with the given issues [Figure 64].

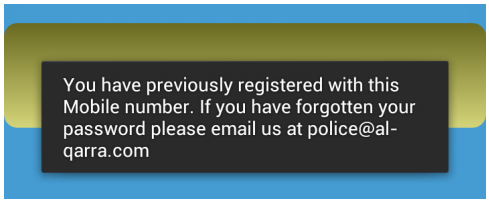


Figure 63: Help user recognise that they have registered before with the given mobile number

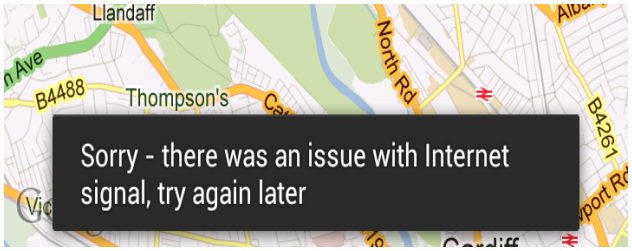


Figure 64: Diagnose message

Help and documentation

Even though it is better for the system to be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

A tutorial page was developed in order to inform the user of what exactly the application can offer and help explain each of the tasks offered by the application. When a user first launches the application they will be given three options. One option is a tutorial page with clear slide navigation function to illustrate the most effective way of using the application and what the application offers as form of functionality to its active users [Figure 65].

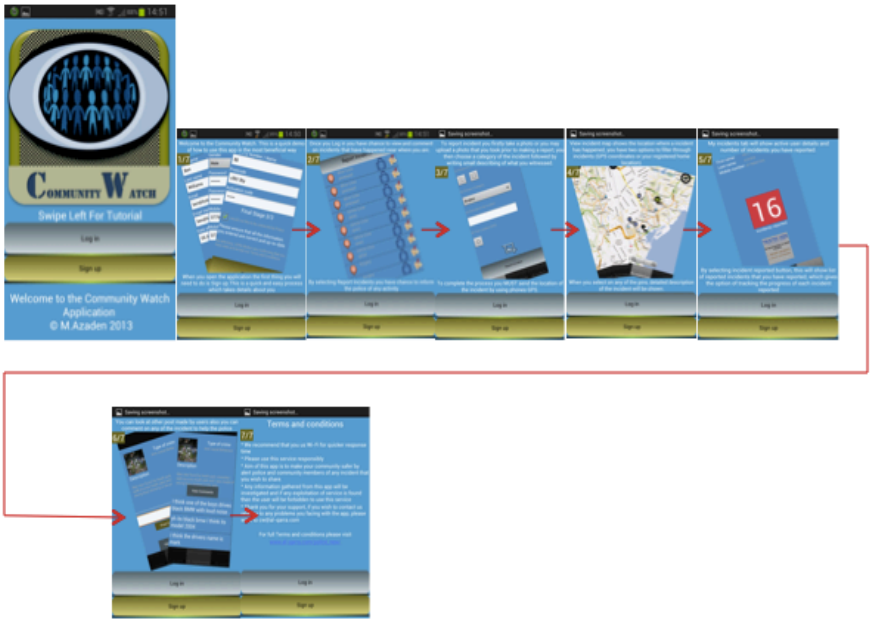


Figure 65: Tutorial page developed to help users with the application features

7.2. User Testing

7.2.1. Application testing

Once the Community Watch application was fully implemented to meet the requirements of the project, the next stage was to test the usability of the app in a real environment. A user test room was set up. This consisted of five users who were asked to use each of the features of the app, i.e. register, login, report and comment on incidents. The information gathered was reviewed by three police officers, who examined the content of the reported incidents to get a real feel of what the application can deliver to the police service.

Before testing the application, a set of tasks were designed in order to allow the user to fully explore the features of the application. The tasks were sent via email to Professor Nick.J. Avis, who is in charge of ethical documentation in Cardiff university, to ensure that the contents of the task and questionnaires comply with ethics [Appendix 2].

7. Testing – Application

Testing the application gave each of the users a working version of the prototype app, and presented them with a series of incidents illustrated by photos in a PowerPoint presentation. They were asked to use the app to "report" each incident, and then answer a questionnaire on their experience of using the application [Appendix 3].

The tasks given to the testers were:

1. Look through the tutorial page - Once this was completed they would get a good feel for the application.
2. Register – each tester were required to register with a given mobile number and activation key, as new users need a valid activation key in order to register.
3. Login – test if the login function works and if the remember me checkbox remembers user details.
4. View incidents – testers were asked to view reported incidents around their registered home location via the incident notification page.
5. Report an incident – the testers were required to use the application to report incidents given scenarios of crime photos.
6. Comment on incident – once reporting an incident was complete, they were asked to comment on incidents that were recently reported.
7. View incidents using map – testers were asked to view all the incidents reported around their GPS coordinates with the aid of Google Maps.
8. View user profile – testers were required to determine whether the presented details on the user profile were correct and presented in a clear manner.

The scenarios that the testers were reporting were as follows:

You are walking to university and you notice that a van has been vandalised with graffiti. You notice a young man walking around the van with a spray can in his right hand, wearing a red t-shirt and carrying a white plastic bag full of spray paint in his left hand. You decide to report him.

You are walking home from a hard day at work; you notice there is a man with grey hoody breaking into your neighbour's house. You decide to report him.

You are walking to your friend's house. You notice that the furniture is scattered across the main road and it's obstructing your way. You decide to report this.

You are annoyed because in the building next to your work place groups of youths gather to smoke and play football. You feel that the building should be closed, as it's not safe. You decide to report this incident.

The full contents of the PowerPoint, the questionnaire, user testing and test cases that were used for testing the application can be found in [Appendix 4-7].

7.2.2. Website Testing

Website testing was not part of the requirements, as the project's initial plan clearly focused on the creation of a mobile application, which can be used in public service. The website was developed to be used as demonstration of what the system can offer to the police service. Therefore critical tests were not conducted. To check whether the website can be used in the police service, three officers were asked to sit in test room to review the information that was generated via the Community Watch app. The information would be examined by the police officers in order to check that the displayed information on the page corresponded to the information the app user presented.

A set of general questions was asked during testing:

1. Was the information presented in clear manner?
2. Was it easy to navigate to different incidents on the map?
3. Was it easy to edit information relating to an incident?
4. Did you manage to add a new user to the trusted network system?
5. Did any of the functions need improvement?
6. Would the website and application be feasible to use in the current police service?
7. Did you require any help testing the website?

7.3. External usability testing

To start testing the application and the website with external users, I separated it into two sections. One of the sections was to test the application with the community group and the other section was to test the application and the website with police officers.

The first task was to contact a number of community groups. This was difficult as many community groups, i.e. Penarth foundation group and Heath community centre, had extremely busy schedules, which meant that they could not participate in interviews. This led to me emailing them with a set of questionnaires and copies of the PowerPoint presentation of the application.

The second task was to contact several police officers. This was simpler than I anticipated, as they were willing to have unformed interviews with me in order to test the application and review the gathered information on the website. As explained above, three police officers contributed to the testing. Each of the police officers were given a questionnaire regarding functionality of the application and a working version of the prototype app with a series of incidents illustrated by photos on the PowerPoint presentation. The questionnaire was filled out during the testing period.

The outcome of the tests will be revealed and clarified in the [8. Evaluation] section.

8. Evaluation and Future work

8.1. Application Testing Feedback

A number of issues were encountered during the testing peered; each of the issues was corrected in order to improve the application's functionality. The problems and the solutions were as follows:

Tutorial page

Problems - Too much information for a simple task.

Solution – This was addressed by editing the contacts of each page on the tutorial section to minimise overloading users with information that is not required.

Registration page

Problem - Date of birth is not working in the correct format. It should be (DD, MM, YYYY), but the application wants the user to input the details in (YYYY, MM, DD).

Solution – This was addressed using the Android data format function. This lets users type their date of birth in UK standard format, and when they send the details it will be converted to the one.com database format. In future I will be considering changing the database provider because of the issues with data format [Figure 66].

```
String dateString1 = date.getText().toString();
        SimpleDateFormat formatter = new SimpleDateFormat("dd.mm.yyyy");
                try {
                        Date dateSS = (Date)formatter.parse(dateString1);
                } catch (ParseException e1) {
                        e1.printStackTrace();
                }
        SimpleDateFormat newFormat = new SimpleDateFormat("yyyy.mm.dd");
        finalString = newFormat.format(dateSS);
```

Figure 66: Date format code

Problem - When the registration page is launched, it starts from page two of the three pages.

Solution – To fix this problem I got the left point of the scrollview and used a method that sends the scrolled to the predefined position when it gets launched [Figure 67].

```
HorizontalScrollView v = (HorizontalScrollView)findViewById(R.id.scroll);
v.smoothScrollTo( v.getLeft(), 0);
```

Figure 67: Predefine start position

Login page

There were no problems with the login page.

View incident notification

Problem - The same contents get displayed in the comments section for all the incidents on the page until the user exits the application to refresh the comment dialog.

Solution – To allocate the correct comments given incident id, the solution was to clear the array, which stores the comments before the comment dialog opens [Figure 68].

```
comment11 =new String[0];
```

Figure 68: Clear the array before adding new information i.e. user comments.

Problem - The comment section lets users use profanities.

Solution – This problem could not be addressed, as I was limited in developing a method of filtering the words the user types as this may restrict the rights and freedom of the users. This problem can be addressed in the future by using external censorship software.

Reporting incident

Problem - GPS locations do not work as well indoors, and sometimes crash the application if the location is not found.

Solution – This was achieved by using a simple check function which checks whether the GPS connection is weak, then it uses the network signals to locate the users coordinates [Figure 69].

```
public boolean internetCON() {

        ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null ) {
           return true;
        }
        return false;
    }
```

Figure 69: Checks if the Internet single is weak. If so then return false to change the connection type from GPS to network connections.

Problem - The image gets automatically rotated 90 degrees on the page.

Solution – Use the rotate function when using matrix bitmap methods to store images [Figure 70].

```
        matrix.postRotate(90);
```

Figure 70: Rotate image

View incidents on the map

Problem - GPS location finder takes an extremely long time.

Solution – This was achieved by using a simple check function which checks if the GPS connection is weak then uses network signals to locate the users coordinates [Figure 71].

```
public boolean internetCON() {

        ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        if (netInfo != null ) {
           return true;
        }
        return false;
    }
```

Figure 71: Checks whether the Internet signal is weak. If so, then return false to change the connection type from GPS to network connections.

Problem - If more than three incidents are reported in the same location, the incident icons appear overlapped, which means that the user has to zoom in in order to select the right incident.

Solution – Using smaller icons for the incidents and making the zoom in button larger than the default size solved this.

Problem - sometimes the application took a long time to load and populate the incidents on the map. They sometimes thought that it had crashed, when it was in fact getting the information from the database.

Solution - This problem was fixed by adding a progress dialog method to the application. For example, when the user select GPS location to get the incidents that have been reported in their current area, the user will see "ProgressDialog" until all the required information is gathered and placed on the map.

User profile

Problem - Could not view the comments that I typed about the incidents that I previously reported.

Solution – After many hours of debugging, the problem was found as I was passing the wrong incident id. This was fixed immediately.

8.2. Critical Evaluation

The approach taken to bring this project to its completion has allowed it to progress without encountering too many significant problems. The background research that took place for the interim report provided sufficient evidence that the project was entirely feasible, whilst also providing me with the required knowledge to begin the implementation, and ultimately gave greater depth to the project.

There is a possibility that the Android application could be integrated with current police services, allowing the project to transform the services available to the public to create significant awareness of community led crime prevention. The test results gathered over the course of this project indicate that this could be a great success; however, given the current financial climate, the funds aren't available to further develop the application and provide it with continued support in the form of dedicated police officers. The responses I received whilst testing the application amongst police officers supported this theory. They believed that the system is not yet ready to be deployed to the public, and the fact that the UK police service is cutting down on police officers would mean that responding to reported events could be problematic.

Overall, the main goals and core requirements that were laid out in the interim report were fulfilled to what I feel is a very good standard. I was able to design and implement both a mobile application component and a website in such a way that they interacted with a single online database to give users freedom of interaction with the incident reports in the format that they desire. The project lifecycle was initially unsuited to the project and prompted a change to a more compatible incremental methodology, which in turn made my work considerably more productive and efficient. The overall approach to designing, implementing and testing was relatively successful, as most of the original plan was carried out without any interruptions. The testing stage itself highlighted a number of useful features that were not foreseen in the initial stages which could be implemented if the system received the approval of the police service. If further development of this project were to be undertaken, it could provide features that would prove invaluable to both police and community members using the system.

8.3. Future work

- When a user reports an incident, they should have the option to identify the criminal by what clothes they were wearing, i.e. red hoody, blue shoes. The app users can also help the police by identifying where the criminal was last seen

- When an incident gets reported using the community watch app, the database analyst and community group should filter the information for any offensive words before the incident appears on the app

- Police officers should have the right to remove any user from the system. This could be done using the website which is connected to the central database. A block list can be implemented into the system; if any user doesn't comply with the terms and conditions they will be blocked from using the app. The information stored in the block list (user mobile number).

- If many users report the same particular incident in one area, then the incident should all link to one incident to give a wider range of information to police. Linking may incidents together will help app users to isolate incidents on the map without needing to zoom in

- Integrate the Google server with the urrent database, so push notifications can be implemented into the app. This will alert users about live incident alerts in their community

- When registering a new user to the app, they should be text messaged a verification code to check if the given mobile number is correct

- Users should have the right to edit their own incidents for a limited amount of time. This will help users to change or add more information relating to a particular incident

- When an incident gets reported they should be fitted into categories. Users can then filter through a list of incidents to find what that they are looking for in the most effective manner

- Multi language accessibility. For example, Welsh speakers should have the right to report incidents in their mother tongue rather than misrepresenting in another language

- A highlighting function, which Google API offers, can be used to separate each location by postcode and user location. However, researching this function I discovered that Android does not support this feature. An alternative option would be to get a third party API integrated, which would allow developers to separate different locations using postcodes [12].

- Using a different method to save images to database using a file system for storing the file paths in the database rather than the image. This was brought to my attention when first developing the application, but because the service provider, one.com, did not allow a connection to the database in this fashion, I usedbase64 to store images inside the database. Using another type of database system that does not use myPHPAdmin as its main gateway would have allowed the storing of file paths in the database.

9. Result

9.1. Goals Achieved

The majority of the requirements of the project were fulfilled in each section of the project, i.e. the application, database and website.

The application carried out the fundamental tasks that were required, i.e. reporting incident, viewing incidents and commenting on incidents. After the testing period, it was clear from the feedback gathered that this system was viable and could be used on a day-to-day basis.

The database was successfully created with all the intended tables and information needed in order to successfully report and investigate stored details of the incidents.

The website successfully presented the stored information about the users and incidents reported with the aid of Google Maps to indicate the location in which the incident was reported. The website offered police officers the ability to edit any offensive words which were posted by users of the application. Also, the website offered interaction between the app users and the police service via level investigation for each reported incident to alert users on the activities that carried out for each of the reported incidents.

9.2. Goals not achieved

The first goal which I feel could have been implemented in more professional manner would be the live incident feed activity on the application, which was implemented using standard methods of transaction that Android offers to developers. T his could have been improved by using a notification server to interact with all the users of the app. The notification server would have worked as the centre hub of all the incidents reported via the application. For example, if the user reports incidents, these incidents will be saved in the database and the notification server would notify all the users close to the incident's GPS location.
One of the original ideas of a notification server which would keep users up to date with a live feed of incidents was thrown out because the service provider one.com did not allow any access from Google API. This means if the project was to be used in the police service then I would have to consider changing my database service provider in order to offer live notification feeds.

The second goal which I feel needs to be addressed is that if the application were to be used in the police service, the transactions between the application and the database need to be improved as they can take a considerably long time to report incidents if the mobile internet signal is low. This could be improved by using algorithms which send the image to a directory on the server and creating a new table in the database which would store the file path and incident ID in order to link the detail of the incident with given image. This can change the conventional way of sending an image and speed up the process of reporting incidents, as there will no longer be a need to use base64 algorithms to convert the image to bit string format.

10. Conclusion

Having successfully implemented a system that gives community groups the ability to report, view, and alert the community and the police service to any incidents using a smart phone via the aid of the Android platform. As can been seen from the results gathered through testing, the system allows users to report and view incidents around their home or current GPS location. The GPS system used for this application is very accurate, with a degree of accuracy close to 10 metres from the device's actual location.

Data mining in this context has proven to be possible, as can be seen through the use of a website which uses the information collected by the Community Watch application to examine the incidents reported and Google Maps to coordinate the location of the incident.

The main goals and aims of the project have been addressed, giving evidence that the practicality of this project, which was to implement a prototype mobile application that enables community groups to communicate with the police service rather calling 999 or 101. Therefore, the use of the Android platform to create a community watch application is entirely feasible. If time was not an issue I would have liked to create the application for use on multiple platforms (iPhone, Blackberry and Windows) with the future work improvements implemented.

It is also worth noting the possible applications of this in the field as a recent BBC news article [19] highlighted the need for a more technological police force to boost preventative crime fighting. Budget cuts within the police force have resulted in a drop in the number of police officers on duty and left police with technology vastly out of touch with current trends. The integration of technologies such as the community watch app into modern crime fighting techniques could help combat a falling police presence with an increased community awareness. Smart phone applications can act as a modern day alternative to the old police "tardis" phone boxes that used to be on every street. They would offer a new means through which the general public could contact their local police station to report incidents. The major issue regarding the deployment of the app to a real world environment would be the potential for the abuse of incident reporting, which has already been seen in the current 999 emergency number. In the past people have phoned the 999 number for a variety of thoughtless reasons, including a well publicised recent incident where a Manchester United fan called the police because he thought the sending off of one of the players was a crime. Such issues can be resolved by the careful implementation of a trusted network that has been mentioned throughout this report.

11. Reflections

In reflection on the processes undertaken to complete this project, I have learnt that proper time management is an essential aspect of getting tasks done within an acceptable time frame; leaving enough time to begin or progress with other tasks that are necessary to complete the remainder of the project. I found throughout the year that this project was undertaken that there were several occasions where I could have better managed my time, especially in the first semester of the year where I had a lot of spare time available to further the project. This would have meant that the implementation stage of the project could have been pushed forward and testing and data collection could have been done a lot earlier. In future works I will endeavour to write up a strict time plan of what I wish to achieve in the time allocated, and aspire to stick to it, thus providing a mechanism, which will ensure that all tasks are complete. Furthermore, I believe that taking a multi-tasking approach may be acceptable to complete tasks whereby several parts of the application can be implemented together to reach the end goal quicker.

I have also learnt to deal with obstacles faced when the implementation of a project hits a dead end. It was often the case that I would not be able to progress further in the implementation of this project due to some unexpected circumstance or perhaps a lack of knowledge of the Android platform. For instance, when implementing the application side of this project, I was faced with the task of integrating Google Maps API into the application. This posed serious problems due to my unfamiliarity with Google API and having never programmed using such a platform. To overcome such issues, I learnt to conduct proper research into the platform and learnt the methods required to implement certain features before diving into the implementation. I believe this to be a transferable skill that I can use in the future to perform the research required of the implementation to prevent arriving at unforeseeable obstacles.

The transferable skills that I have learnt during the duration of this project are:

Conducting proper research and record keeping to make it comparatively easier when writing up and presenting written material such as this report. In future I believe this process will be of great use and give a broad outlook within the field of the project being developed, ensuring that time is not wasted experimenting and implementing aspects of the system that have been done previously.

The importance of thoroughly designing the system before implementation. This has proven to be a valuable process as it has allowed for a much more sound implementation and avoided many of the issues that may have arisen if the designs were not put in place. Furthermore, the designs provided a mechanism for me to adequately assess and evaluate my work. In future I will continue to thoroughly design projects before coming to the implementation stage, and more than likely to a higher standard than that which has been achieved in this project.

Over the course of my university degree I have been provided with a thorough foundation in programming languages such as Java, XML, HTML, MySQL, Bootstrap and PHP. Throughout the duration of my final year project I have continuously developed and built upon my programming skills in a wide range of different languages. I have also come to understand a number of features and functionalities during the project which have helped to give me a broader understanding of Android and web development. Overall, the year long development process for this project has provided me with a great many lessons, upon which I can build in the coming years. It is my belief that not only was this project carried out to its maximum potential given my skillset, but in the process it also greatly changed that skillset for the better.

12. References

[1]. IncrementalMethodology
Mills H. O'Neill D. Linger R. Dyer M. Quinnan R. "The Management of Software Engineering" IBM Systems Journal, 24(2), pp 410-477, 1980

[2]. AndroidDevelopers requirements
Core requirements for Beginners - Android Developers. 2012. [ONLINE] Available at: http://developer.android.com/training/basics/firstapp/building-ui.html. [Accessed 21 April 2013].

[3]. AndroidDevelopers Fragments
Start with Fragments - Android Developers. 2012. [ONLINE] Available at: http://developer.android.com/guide/components/fragments.html. [Accessed 21 April 2013].

[4]. Bootstrap Twitter
Getting started - CSS Bootstrap samples. 2011. [ONLINE] Available at: http://developer.android.com/guide/components/fragments.html. [Accessed 21 April 2013].

[5]. Royal Mail
How Postcode work - Post Office. 2009. [ONLINE] Available at: http://www.postoffice.co.uk/postcode-finder/how_it_works/. [Accessed 21 April 2013].

[6]. NBCNEWS
Thomson Reuters. Technology: Evernote resets 50 milions passwords after hackers access user data. 2013. [ONLINE] Available at: http://www.nbcnews.com/technology/technolog/evernote-resets-50-million-passwords-after-hackers-access-user-data-1C8659106. [Accessed 21 April 2013].

[7]. NBCNEWS
Gary Hill. Hackers target Twitter, access about 250,00 user accounts . NSPIS Case Study. 2013. [ONLINE] Available at: http://www.information-age.com/technology/security/2171288/hackers-access-up-to-250k-twitter-accounts. [Accessed 21 April 2013].

[8]. Wikipedia
Cryptographic hash function. MAC, Message authentication code. 2013. [ONLINE] Available at: http://www.information-age.com/technology/security/2171288/hackers-access-up-to-250k-twitter-accounts. [Accessed 21 April 2013].

[9]. EncryptingPasswords
Atwood J. Coding Horror: You're Probably Storing Passwords Incorrectly. 2011. [ONLINE] Available at: http://www.codinghorror.com/blog/2007/09/youre-probably-storing-passwords- incorrectly.html. [Accessed 21 April 2013].

[10]. Mobile App Usability
Mike Brown. 10 Heuristics for User Interface Design. 2011. [ONLINE] Available at: http://www.mobileapptesting.com/jakob-nielsen-on-mobile-app-usability/2011/04/. [Accessed 21 April 2013].

[11]. Jakob Nielsen's Ten Heuristics
Nielsen J. 10 Heuristics for User Interface Design. 2012. [ONLINE] Available at: http://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 21 April 2013].

[12].Dyngeometry
Google Map state marker - Dynamic Geometry. 2008. [ONLINE] Available at: http://www.dyngeometry.com/web/Home.aspx. [Accessed 21 April 2013].

[13].Android Book Wei-Meng Lee, 2012. Beginning Android 4 Application Development. 1 Edition. Wrox.

[14].Android Book 2 Mark Murphy, 2011. Beginning Android 3. 1 Edition. Apress.

[15].Stack Overflow
Newest 'android' Questions - Stack Overflow. 2012. [ONLINE] Available at: http://stackoverflow.com/questions/tagged/android. [Accessed 21 April 2013].

[16].Android Tutorial Development & Programming
Android Apps | Android Tutorial | Android SDK Development & Programming. 2011. [ONLINE] Available at: http://www.edumobile.org/android/category/android-apps/. [Accessed 21 April 2013].

[17]. One.com
One.com Web Hosting - Domain - Hosting - E-mail - One.com. 2012. [ONLINE] Available at: https://www.one.com/en/. [Accessed 21 April 2013].

[18]. W3Schools
W3Schools Online Web Tutorials. 2012. [ONLINE] Available at: http://www.w3schools.com/. [Accessed 21 April 2013].

[19]. BBC UK News
Danny Shaw. Tom Winsor: Police should focus on crime prevention 'not catching criminals'. 2013. [ONLINE] Available at: http://www.bbc.co.uk/news/uk-22333662. [Accessed 29 April 2013].

13. Appendix

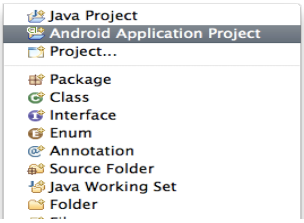
[1]. Android Tutorial

To create an android application the fundamentals the user will need the following:

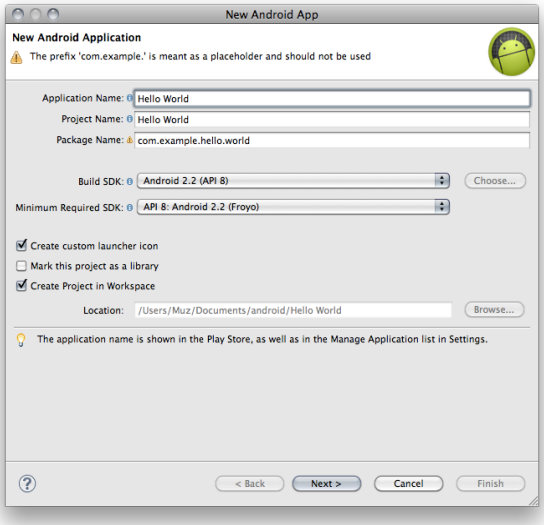
- An Android software development kit (SDK) which provides a number of file containing all of the Android SDK classes necessary to build your application along with documentation and samples to help new programmers navigate through the packages offered by the SDK. Another useful feature Android SDK offers to developers is screen emulator that can be used to test the application.
- Eclipse is required to integrate the tools that Android SDK offer, in order to integrate the SDK with Eclipse software you will need to use Android Development Tools (ADT), which allows easy migration to give the user easy usability to create new projects and interfaces on the android platforms.

The above two requirements need to be installed and setup to start programing android applications. To give taste of android programming this tutorial will consist of simple functions that need to be implemented in order to create "Hello World" application.

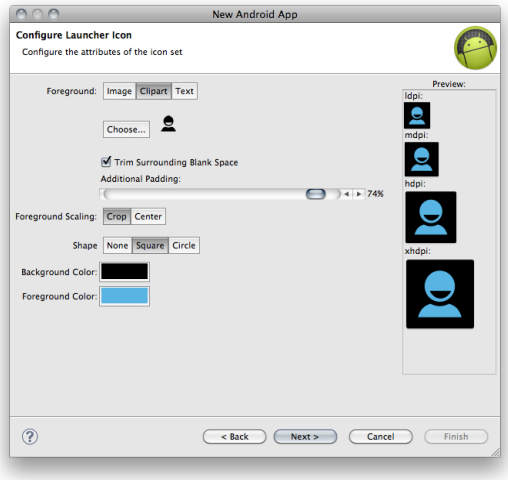
When Eclipse loads, the first that need to be done is to click on "File" tab and scroll down to "New" then list of options will be displayed, the one you need to click on to create android application is "Android Application Project" as this can be seen below



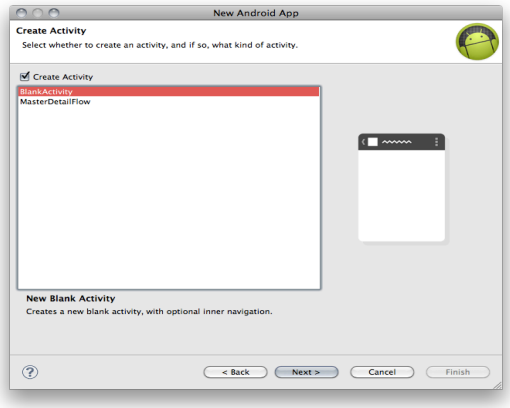
The next stage is to name the application in this case its "Hello World" and the android SDK version we will be using is Android 2.2. When the user is satisfied click on "Next" button. This can be seen in below



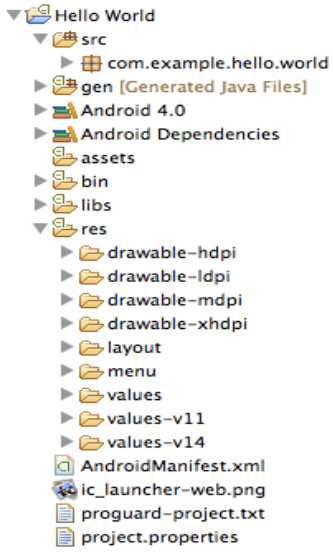
The following stage is to choose an icon for the application, this could be predefined images or use image from your computer.



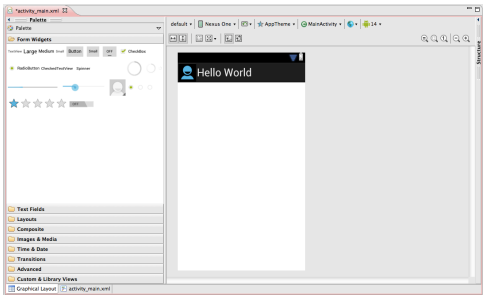
The final stage is to create blank activities from android framework and click on "Next" button and to complete the process just click on "Finish" button.



When a new Android project is created a user has access to a number of file displayed below.



First thing that we will be looking at in the program is the interface which is located in (res/layout/), Android offer a Graphical Layout of the XML files used to create layouts this gives the rights to the developers to drag and drop components on to the interface of the program.



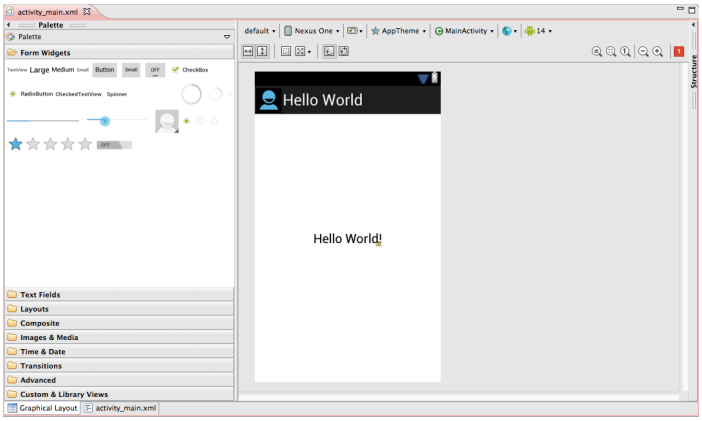
To add text to the page of the program just drag “Large” from list of components android offer, this can be found on the left hand side of the screen. To edit the information of the default text, change the view to “.xml” which could be found on the bottom left hand side of the screen. The program will user TextView to display message, to change the content of the message just find the key word “android:text” in the xml file and replace it with the desired text i.e. android:text="Hello World!"

```
<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" >

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="201dp"
    android:text="Hello World!"
    android:textAppearance="?android:attr/textAppearanceLarge" />

</RelativeLayout>
```

To view the outcomes of the modification just switch back to the “Graphical layout”.



To initialise above xml layout and link it to the application, in order to initialise above xml layout and link it to the application, the first step is to open the directory file (src/) and open the “MainActivity.java”.

```
package com.example.hello.world;

import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;

public class MainActivity extends Activity {

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
 // calls upon the reference within the R.java  class.
   }
   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      getMenuInflater().inflate(R.menu.activity_main, menu);
      return true;
   }
}
```

Android contains named “R.java” which can be found (gen/) folder. The R.java file consists of all references to all the objects created within the application, this file gets generated automatically and should not be modified under any circumstances.

```
package com.example.hello.world;
public final class R {
   public static final class attr {
   }
   public static final class drawable {
      public static final int ic_action_search=0x7f020000;
      public static final int ic_launcher=0x7f020001;
   }
   public static final class id {
      public static final int menu_settings=0x7f070001;
      public static final int textView1=0x7f070000;
   }
   public static final class layout {
      public static final int activity_main=0x7f030000;
   }
   public static final class menu {
      public static final int activity_main=0x7f060000;
   }
   public static final class string {
      public static final int app_name=0x7f040000;
      public static final int menu_settings=0x7f040001;
      public static final int title_activity_main=0x7f040002;
   }
   public static final class style {
      public static final int AppTheme=0x7f050000;
   }
}
```

Every application gets given an AndroidManifest.xml file this is used to store information about the following

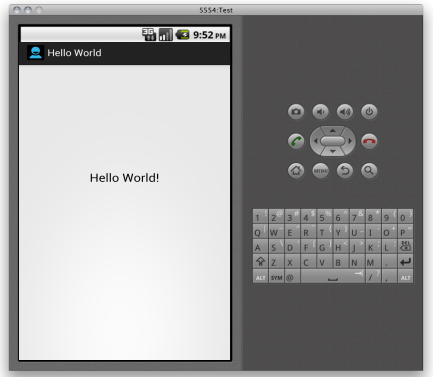
- Permissions i.e. Internet, GPS access.
  <uses-permission android:name="*android.permission.INTERNET*"/>

- Application compatibilities i.e. minimum SDK needed to run this application is version 8 which is 2.2 android framework.
  android:minSdkVersion="*8*"
  android:targetSdkVersion="*15*" />

- Class names that are part of the program, also new class names have to be declared in the manifest in order to use within the application.
  <activity

  android:name="*.MainActivity*" >

Now the application is complete and its time to test the outcome of the program using emulator provided by SDK, this could be done by clicking on "Green Run" button.



Final stage of this process is to compile the application and create .apk file in order to give access to users to run the application on many different mobile device. To compile an application, developer needs to register their personal information with android api to get Keystore file. This is user to identify the author of the application, also this gives the right to the developer to upload their application without worrying about hackers getting access to the code.

Developers can use default Keystore provided by the Android SDK this can be used only for debug applications. But the default Keystore could not be used to get Map API key to access Google maps.

To create Keystore , just click on the application you wish to compile on Eclipse and scroll down to "Export application signed" a popup window will appear requiring the following :

- Program name: "Hallo World"
- Keystore name: "test.keystore"
- Keystore password: "android"
- Key alias: "androidtestkey"
- Key password: "test"

To complete the process select on the directory you wish to save the Keystore and click on Finish button. This process creates .apk file of the application and store the Keystore in “android” folder on the users computer.

Google Map API key

To integrate Google Map in to the application, you will need to use Keystore file to define your application in order to get Google Map API key.

1. Register the MD5 fingerprint of the certificate that will be used to sign an application. The Google Maps registration service then provides a Maps API Key generates corresponding certificate given MD5 fingerprint.

2. Boot up terminal in your window or mac operating system and direct the terminal to the location which the Keystore file is saved, as default its saved under (`C:\Users\USERNAME\.android)`.

3. `Type the following command in order to generate` MD5 fingerprint for your application.

   keytool -list -alias androidtestkey -storepass android - keypass android –keystore test.keystore

4. `This will return` MD5 fingerprint

   androidtestkey, Apr 16, 2012, PrivateKeyEntry,

   Certificate fingerprint (MD5):

   11:AC:73:73:A4:D1:16:4A:BD:44:A2:6C:

5. now to submit the MD5 fingerprint to Google developers site , just follow this link “https://code.google.com/apis/” .This website requires the MD5 fingerprint and gives string value similar to the following:

   0HaL4YXG-4MQEf9f56xBv_q1vD9LAOsyoqWoD8g

6. The final process is adding A map function to the program this can be done by copying the xml code given by Google Maps API and declaring it in any of the XML pages of the application you wish. The Maps API Key can be used multiple times in any Android application as this Maps API Key is uniquely identifies the author and developers machine.

```
<com.google.android.maps.MapView
        android:id="@+id/map"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:progress="15"
        android:apiKey="0HaL4YXG-4MQEf9f56xBv_q1vD9LAOsyoqWoD8g"/>
```

[2]. Professor Nick Avis, Email confirmation

Dear Mustafa Azaden

Thanks for the email. With the controls and actions you have outlined
I do not anticipate any issues

Please let me know if any issues do arise
Good luck with your study

Kind regards
Nick

Quoting Mustafa Azaden <AzadenM@cardiff.ac.uk>:
> Dear Professor N.J. Avis
>
> I have been developing a mobile app for my final year project, to
> allow citizens to report crime and disorder in the community, and
> now I am in the testing stage for the usability side of the app. I
> plan to ask several fellow students to test the functionality of
> the app. I will assemble them in a room, give each of them a
> working version of the prototype app, and present them with a
> series of incidents illustrated by photos in a powerpoint
> presentation. They will be asked to use the app to "report" each
> incident, and then answer a questionnaire on their experience in
> using the software.
> I have shown my slides to my supervisor Alun Preece and due to the
> content, specifically the images, he advised I send them to you so
> that you could review them, together with the questionnaire (which
> is focussed only on technical aspects of the software). Please note
> the first slide contains a "warning" that the content will include
> images of common "street" crime and disorder.
> I hope to run tests with fellow students (all of whom have
> volunteered to participate) next Wednesday, so would deb grateful
> for a quick reply.
>
> Regards
> M.Azaden

[3]. Test room layout

[4]. PowerPoint –Testing tasks

The PowerPoint used during testing phase can be found in extras.zip file.

[5]. Questionnaires

The Questioners used during testing phase to find the problems with the application can be found in extras.zip file.

[6]. User Testing

User feedback from questioners can be found in extras.zip file.

[7]. Test Cases

To test the application I input simple test cases and review the expected and actual outputs. This can be found in extras.zip file.

[8]. Code listing

The code listing developed during implementation section can be found in Project_code.zip file.