Metropolia

Naveena Dhougoda

Mobile Feedback System in Android Platform

A Project of Helsinki City

Helsinki Metropolia University of Applied Sciences
Bachelor in Engineering
Information Technology
Thesis
14th April 2012

Metropolia

Abstract

Author(s) Title Number of Pages Date	Naveena Dhougoda Mobile feedback system in Android Platform, A project of Helsinki City 60 pages + 9 appendices
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Kari Salo, Principal Lecturer Anu Kiiskinen, Landscape Architect

This project was carried out for the Public Works Department of Helsinki City, and the purpose was to create a location-aware feedback application for the Android mobile platform. The created application was capable of interacting with a remote server and rendering the user interface at runtime depending upon data retrieved from the database. In addition, a web application was also required to be created to handle the database.

The mobile application was developed in the Android platform using the Java programming language in Eclipse IDE. MYSQL was used for database design and PHP was used as a server side scripting language. The web application consisted of simple HTML pages with CSS and Javascript.

During the project, a localized Android application, capable of accessing device features such as camera and GPS and creating the user interface dynamically, was developed. The web application was also created, and it acted as an interface for creating, viewing and modifying the data in the database.

The developed Android application can be used effectively to get feedback from general citizens in workshops organized by the Public Works Department of Helsinki City in the form of general text, images and location information. As the application is localized, it is accessible to people using the Finnish, Swedish and English language.

Keywords	Android, Mobile Application, Geo Location, Fragment Activity, Touch User Interface, Localization

Contents

1 Introduction.....1

2 Android Application Development.....3

2.1 Android Overview and Operating System Architecture.....3

2.2 Android Application Architecture and Components.....5

2.3 Android Development Tools and Environments.....7

2.4 Android Localization.....7

2.5 Touch-based User Interface and Its Implementation in Android Application.... 12

2.5.1 Touch-based Android User Interface.....12

2.5.2 Touch Event Handling in Android Applications.....15

2.6 Android Fragment Activity.....16

3 Web Application Development Overviews and Technology Involved.....19

4 Database Design for the Feedback Application.....21

5 Web Application.....23

5.1 Project Background and Requirement Specifications.....23

5.2 Design and Implementation of User Interface.....24

6 Android Application 34

6.1 Requirement Specifications.....34

6.2 Architectural Design.....36

6.3 Design Implementation and Development Technologies.....39

6.4 Graphical User Interface.....41

7 Testing……48

8 Results……50

9 Conclusion……52

References……53

Appendices

Appendix 1. Helper Classes for Android Application

Appendix 2. Custom Adapter Class to Create a List Item for Questionnaire Activity Class

Appendix 3. Custom MapActivity Class

Abbreviations

GPS	Global Positioning System
UI	User Interface
SDK	Software Development Kit
IDE	Integrated Development Environment
API	Application Programming Interface
OS	Operating System
ADT	Android Development Tool
ISO	International Organization for Standardization
HTML	Hyper Text Markup Language
REST	Representational State Transfer
SOAP	Simple Object Access Protocol

1 Introduction

Mobile phone industry has been developing and growing rapidly during the last couple of years. Old mobile devices with limited functionalities are being replaced with new smart phones which are more advanced and elegant. There are many device manufacturers developing these devices in different platforms such as Android, Symbian, iOS and Windows. Apart from normal mobile phone functionalities, these smart phones are used for many other varied purposes. Nowadays, with smart phones, one can, not only make calls, send messages, play games, listen to music and browse the internet but also perform many other activities where sky is the only limit. For these various purposes, there are applications developed which can be installed in the phone and used easily.

With the phenomenal development of mobile platforms and rise in the use of mobile devices, mobile application development has been a competitive field and its use has become a necessity for the service providers in various areas to provide services through mobile applications. Due to these reasons this project, which is a project of Helsinki City, has been carried out to develop a mobile application for the Android platform.

The goal of this project was to develop a mobile application in the Android platform and a web application to maintain the database for the data used in the Android application for Helsinki City Public Works Department. The department is planning to organize various workshops in the future to get public responses and feedback of their work. For this purpose, it was necessary to develop a mobile feedback application which would act as a platform to answer the questions created by the Helsinki city personnel and which would save the answers in the back end server. Hence, the application should be able to fetch questions from the database, provide a user interface for the users to answer the questions and save the answers provided by the users back to the database. The application should be capable to get the location information and capture image which means it should have access to the native camera application and the GPS (Global Positioning System) feature.

The web application is targeted for the administrators or company personnel and it acts as an interface to handle the database. With this application, users should be able to add questions into the database, modify the data, delete the data, view the answers fed through the Android devices and fetch the answers through a file.

2 Android Application Development

2.1 Android Overview and Operating System Architecture

Android is a free and open source Linux based operating system for mobile devices, developed by Open Handset Alliance led by Google. It provides open source libraries for application development, an application framework, a user interface framework, some pre-installed software and a software development kit for application development which includes tools, plug-ins and documentation. The open source policy of Android ensures that one can change or fix any deficiencies in the user interface or native application design by writing an extension or replacement. [1,4.]

The Android device first released in October 2008 was HTC G1 with the first Android version. Since then, various Android versions have been released which are Android 1.5 (Cupcake), Android 1.6 (Donut), Android 2 (Eclair), Android 2.2 (Froyo), Android 2.3 (Ginger Bread), Android 3 (Honey Comb), and Android 4 (Ice Cream Sandwich) [2]. There are many phone manufacturers who are adopting the Android platform in their mobile devices such as Samsung, HTC, Motorola, ZTE, LG and so on. Figure 1 displays two different models of Android devices.

Figure 1. Samsung Galaxy Nexus with Ice Cream Sandwich (left) and HTC G1 with the very First Android version 1.0 (right). Modified from Murph D [3] and HTC G1 Review [4].

Figure 1 shows the difference in the first and latest Android version. The device on the left is the latest Samsung device with the latest Ice Cream Sandwich Android version while the device on the right is the first HTC device adopting Android.

The Android operation system consists of various layers which are not totally separated but seep into each other with their own characteristics and purposes [5, 7]. These layers of Android architecture are illustrated below.

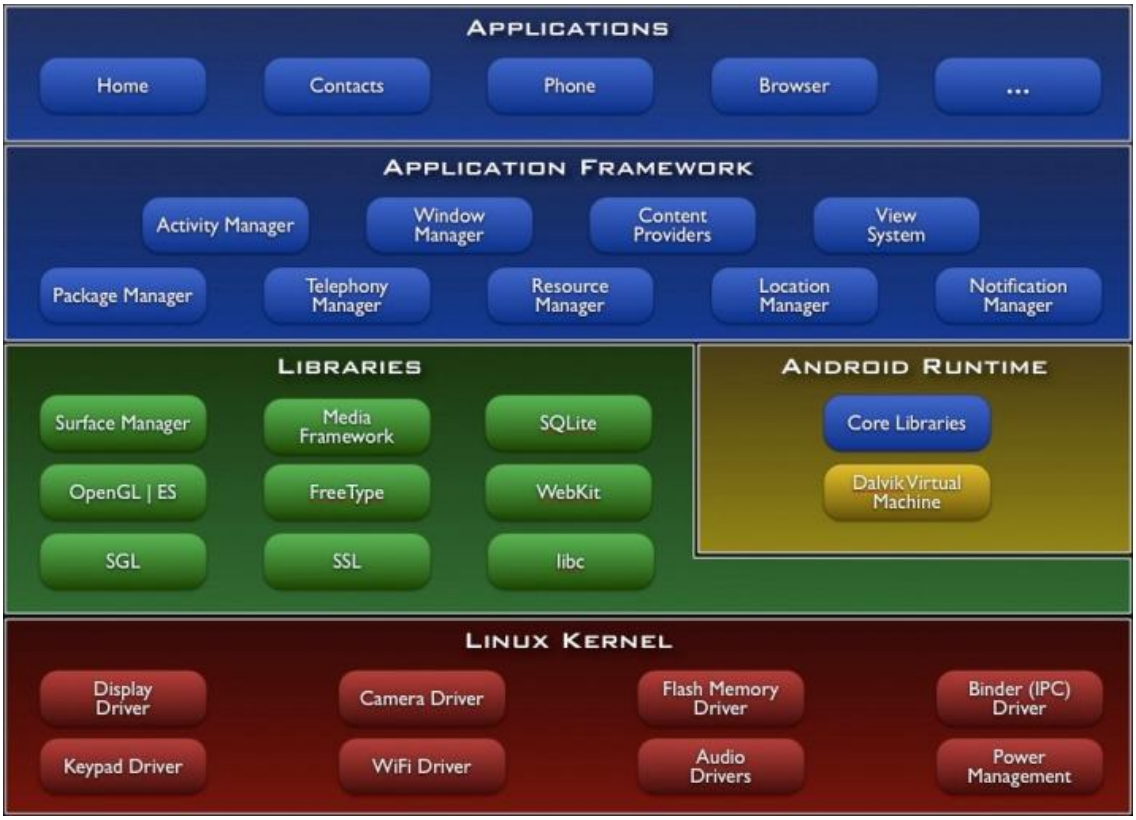

Figure 2. Android Architecture [6]

Figure 2 displays the layers that form the Android architecture. The core services such as handling hardware drivers, memory management, power management network, security and so on for Android are handled by **Linux 2.6 Kernel.** As the engine to power the applications and the libraries, **Android Runtime** forms the basis for the application framework. It includes the core libraries and Dalvik Virtual Machine. Android core libraries provide the functionality of Java core libraries and Android specific libraries. Dalvik is a register based virtual machine which ensures that multiple

instances can be run efficiently in a device. All the classes and Java libraries used for application development are provided by **Application Framework.** It also manages the user interface and provides access to the hardware interface and services of the Android device such as locations, sensors, Wi-Fi and so on. **Application layer** is the uppermost layer of Android architecture, consisting of all the available applications. These applications run in the runtime layer and use the classes and services provided by application framework. [1, 13-14.]

2.2 Android Application Architecture and Components

Component reusability is the basic feature of Android architecture. This feature enables to share and access the services and data used by other applications. This means that an application can be exposed through other applications and the data between the applications can be exchanged. [1, 15.] For instance, the images in the gallery of the device can be extracted into a photo editing application, a VoIP application can get the contact information from the phone contact list, a notification in a Facebook application or a new email arrival notification can be delivered to the users through Notification Manager and so on. In this way, different applications can interact with each other to provide users seamless and efficient services. The various services provided by the Android application framework are described below.

- Activity Manager: It controls the lifecycle of an activity and manages the activity stack [1, 15]. When the user starts an application, an activity manager will create its activity and put it in the screen, and when the user switches screens, the previous activity is moved in a holding place in a stack by the activity manager. This way, a previously used application can be started more quickly. If the application is not used for a while, this is destroyed to make space for the new application. Thus, the activity manager is responsible for application management and its tracking which improves the speed of the user interface and overall user experience. [5, 28.]

- Views: These components are used to create the user interfaces for an application. All the UI (User Interface) elements which can be seen are views such as Textview, Button, Edittext, Scroll Bars and so on. The user interface of

Android is created mostly by views and layouts. Each layout organizes views by binding them in proper order and groups. [5, 48.]

- Notification Manager: Notifications are the signals used to alert the users about certain events that may require attention or to indicate ongoing background services that have been set to foreground priority. The notification manager is a system service that provides a mechanism to inform the users about the notifications without using an activity [1, 309]. Using the notification manager, new notifications can be triggered, existing notifications can be modified and non-required notifications can be removed [1, 310]. Apart from these, it has the ability to perform the following actions.

    - Create notification icons in the status bar.
    - Flash the device light.
    - Generate additional phone effects such as vibration and ringtones [1, 309].

- Content Providers: These provide an interface for sharing information between different applications [5, 32]. This mechanism exposes the device data resources such as contacts, media store, bookmarks and phone call to retrieval and updates in a separate third party application [7]. With standard methods like insert(), update(), delete() and query(), the content provider seems similar to a standard database model. Hence, it is relatively easy to implement this service as a proxy in the database. [5, 34.]

- Resource Manager: It manages the resources such as strings, multimedia objects or external files needed for the application [1, 15]. Externalizing resources, either simple resources like strings and colors or complex resources like animations, images and themes, helps to maintain, update and manage them easily. This practice lets the application to easily define alternative resources specific to hardware configuration, language and location [1, 60]. The management of these resources is performed by the resource manager.

2.3 Android Development Tools and Environment

The set or collection of development tools and other external dependencies required for the Android application development are included in Android Software Development Kit (SDK). For each version of Android there is a separate SDK. Developers are required to have an SDK for the application development to be used alone or with the application development IDEs (Integrated Development Environment) such as Eclipse, NetBeans and so on. IDE refers to the set of tools or the environment consisting of the graphical UI that helps or provides a better template and features for the application development [8]. The Android SDK consists of the following.

- Required Libraries
- Debugger
- Emulator
- Documentation of Android Application Programming Interface (API)
- Sample Source Code
- Tutorials for Android Operating System (OS). [9.]

One can get the Android SDK from the official Android developer's page which also includes all the installation instructions. The SDK is available for the Windows, Linux and Mac platforms. In Eclipse, the SDK can be installed with the help of the Android Development Tool (ADT) plug-in. ADT is the plug-in for Eclipse IDE that provides the entire necessary environment for the Android application development in Eclipse. It extends the capabilities of Eclipse by creating an application UI, adding components based on the Android Framework API (Application Programming Interface), allowing to debug the application using Android SDK tools and so on.

2.4 Android Localization

Android being an open platform that is used worldwide, an Android application developed in one location can also be used in other parts of the world. Hence, to

obtain a larger audience or to target the application for specific locations, it is efficient to make a single application capable of displaying its contents specific to device locale configuration instead of making different applications for different locales and in different languages. This feature is known as localization. It is the process of making an application, software or a system suitable and compatible to the specific language and region by adding locale specific contents and translated text [10]. Locale does not only refer to the geographic region but also to the language and location specific customs like date formats and the system of measurements.

In the Android application development, all the resources used except the codes are externalized making them easier to maintain, upgrade and manage [1, 60]. Resources are strings, colors, dimensions, drawables, layouts, menus, searchables, styles, raw resources and other static data required in an application. An application can include multiple sets of resources customized for different device configurations which are selected automatically during runtime that best matches the device [11]. In most of the cases these resources are responsible for the user interface framework and can be changed in the application executables without the need of recompilation and redeployment. [12,58.] The default and alternative resources for the application to use, which mostly consist of XML files, except for media files, are created and placed within specifically named subdirectories of the '/res' directory of the project. Figure 3 displays the use of resources in an Android application project.

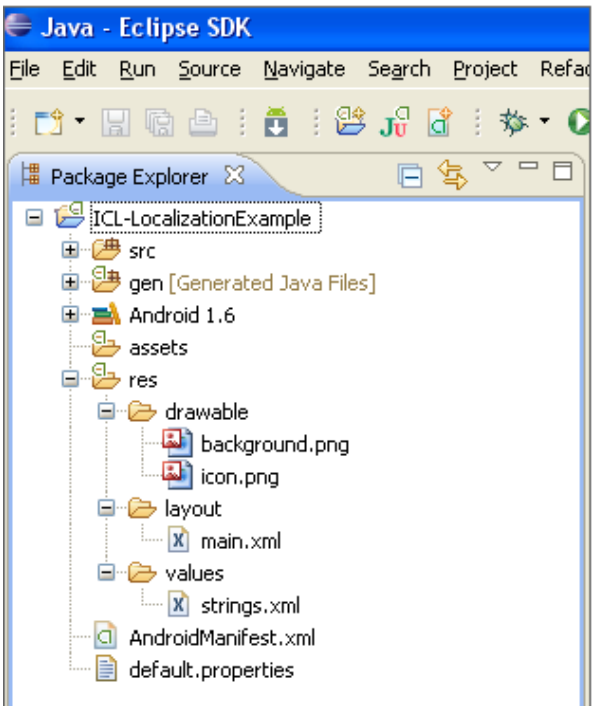

Figure 3. Sample Project Structure. Reprinted from Android Localization Tutorial [13].

As displayed in figure 3, the resources are positioned under separate specific folders under the 'res' folder of the main project tree. Images are included in the '/drawable' folder, layouts are included in the '/layout' folder and the general values are located in the '/values' folder. These folders consist of default resource values of the application.

This mechanism of externalization of resources also aids to easily define the alternative resources required for different locations and languages. Localizing an application mostly provides alternative text for different languages and alternative graphics, sounds, layouts and other locale specific resources, if required. Alternative resources for different locales are specified in separate locale specific folders which conform to the naming scheme; otherwise, the application will not compile. [11.]

The ISO format for a locale identifier is of the following syntax: language_Region, where language is represented first, as two lowercase characters, followed by the region which is represented as two uppercase characters, such as en_US for United States, en_AU for Australia and so on [14, 511]. The alternative resource folders are specified using a parallel directory structure within the res folder. The name of the res folder for the different language is appended with the locale representation, using a hyphen (-) such as values-fi for the Finnish locale, values-en_US for the USA and the English language, drawable-fr for images for the French language and layout-fi for layouts for the Finnish locale [1,71]. In figure 4, the folder structures for default value and additional locale values are presented.

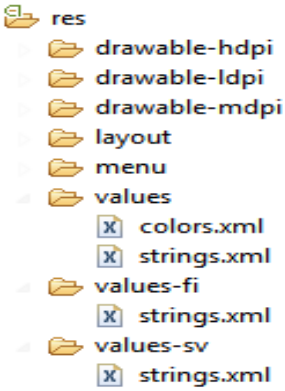

Figure 4. Folder Structure Representing Localization

Figure 4 demonstrates the layout of the folder structure required for defining the resources for each locale. The 'Strings.xml' file is used to manage the string resource of the application and the file is stored under the 'res/values' folder which consists of default values for the application [14, 514]. The other folders with prefix 'values' consist of similar string values for the specific locale. In the strings.xml files of all three folders the string item consists of language specific values but with the same name that makes a string tag or item a single object but with different values. An instance of the difference in the contents of the string.xml files for different locale is presented in listing 1.

string.xml in values:

```
<resources>
        <string name="username">Name(optional)</string>
        <string name="email">E-mail</string>
</resources>
```

string.xml in values-fi:

```
<resources>
        <string name="username">Nimi(valinnainen)</string>
        <string name="email">Sähköposti</string>
</resources>
```

string.xml in values-sv

```
<resources>
        <string name="username">Namn(Tillval)</string>
        <string name="email">E-post</string>
</resources>
```

Listing 1. Difference in the string.xml File for Different Locale

As presented in listing 1, all XML files consist of the same 'string' tags with the same attribute values. However, the value contained in the tag is language specific. Thus, the string items with the same name value in both folders are represented with the same resource ID but are selected dynamically according to the locale configuration during runtime. At first, the application looks for the locale specific resource files but if not found it checks the default folder *'values'* for the required value. If the resource is not found in the default folder as well, the application cannot compile. Hence, all the used resources should be at least defined in the default folder.

2.5 Touch based User Interface and its implementation in Android Application

2.5.1 Touch based Android User Interface

With the radical development and evolution in the multimedia devices, the technology involved in the human-device interaction has also been evolving rapidly. Along with the functional performance, user experience plays a vital role in the overall device usability. For this reason, designing better user interface and providing proper and efficient technology to feed the user input is essential in the design of any device. One of many ways of interacting with the digital devices is the touch based user interface or touch screens where the display acts as both input and output medium [15]. This technology has proved to be incomparably efficient, providing users a sense of control with direct interaction with the device using one of the human's sense mechanisms [16].

A touch screen is an intuitive computer input device that works by simply touching the display screen, either by a finger, or with a stylus, rather than typing on a keyboard or pointing with a mouse. Being the simplest, most intuitive and easiest of all other input devices, this is being adopted in wide variety of digital devices and platforms including Android. [17,239.] With the touch screen as a user interface, Android applications should adhere to certain design principles which can improve user experience and maintain the device standards. The basic design principles of UI development in touch screens are the following:

- Focus on the ways users will be using the interface
- Make the frequently used and important items clearly visible to the user
- Give proper feedback on the user's actions which helps in reducing the error rate when using the application
- Predict the ways and behavioral trends of using the interface by the users
- Design the UI capable of reversible actions [18.]

Android has its own design patterns for the user interface. According to Android, a design pattern is a general solution to a recurring problem. Considering the device design itself, there is presence of a Back key, which is fixed and always available to the users. This leaves some extra space in the application which can be used for other items. Similarly, it has a dashboard pattern which prevents the navigation of several layers within the application. This pattern consists of three parts which are Action Bar, Search Bar and Quick Action. Being similar to a conventional website banner, an action bar consists of a logo or title typically on the left and a navigation item on the right. Search Bar as its name suggests consists of a bar or box that provides a way for a user to search items and also suggestions. Quick action refers to pop up behavior that gives additional contextual actions to the users. [19.]

The user interface of the Android application is rendered in the screen of the device using an XML file. A screen is also referred to as an "Activity" which comprises of multiple view components, groups and layouts. Different types of view components in Android are shown in figure 5.

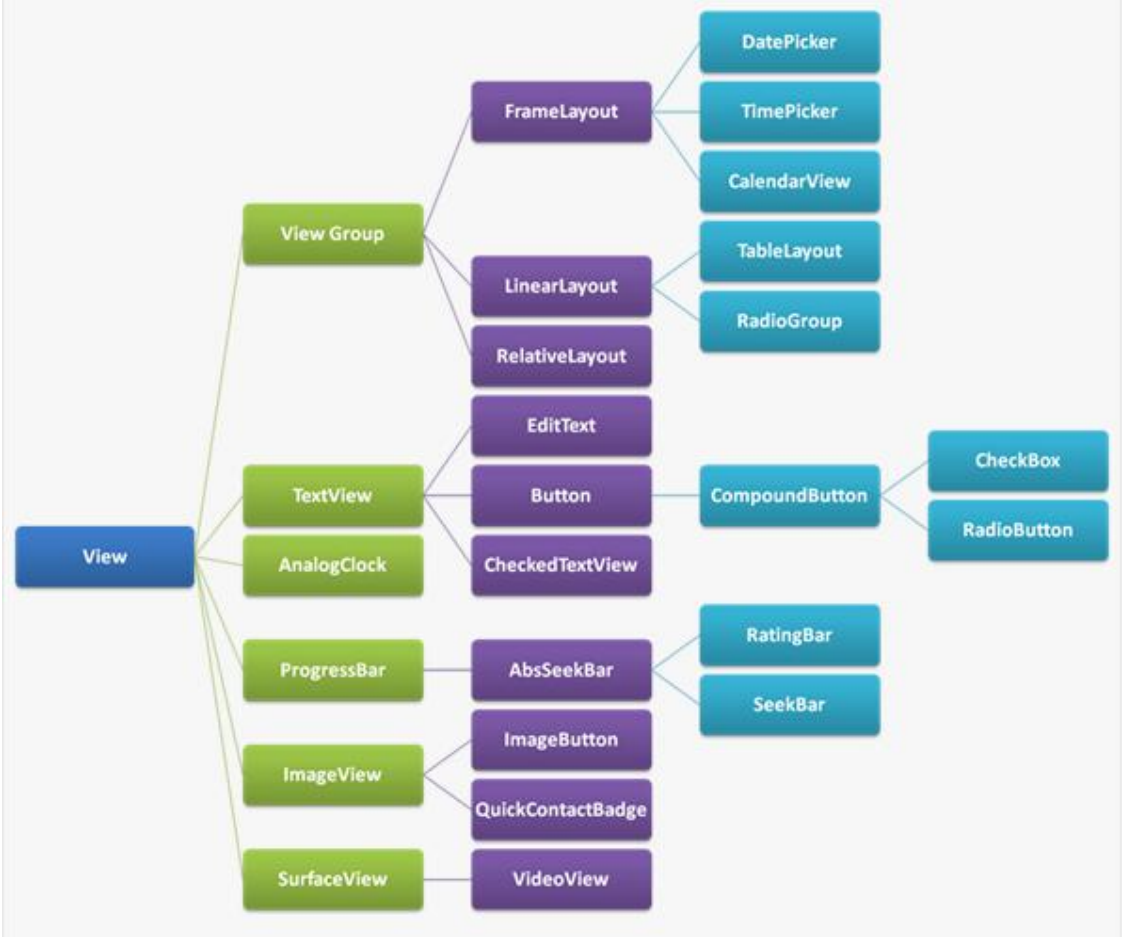

Figure 5. Classes from Android View Hierarchy - Copied from Android Tutorial [20].

As figure 5 shows, the View root class consists of different subclasses which are used to create an Android user interface. The view group class consists of the layouts which are used to arrange other views in proper order. These different types of views and layouts are used to create an activity or screen of the Android application. A sample of a user interface is illustrated in figure 6.

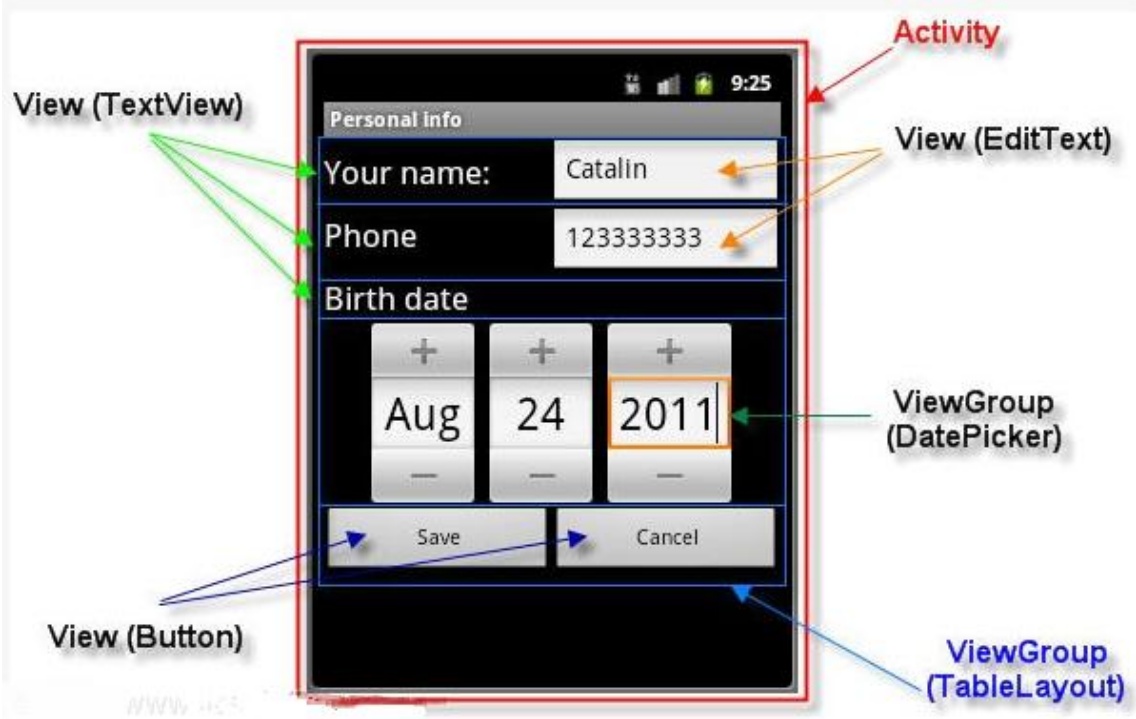

Figure 6. A Sample of Android User Interface - Copied from Android Tutorial [20].

Figure 6 shows a simple and typical user interface which consists of different types of views in a table layout. This layout arranges all the view components in proper order. The screen that consists of all the views is termed as an activity.

2.5.2 Touch Event Handling in Android Application

Almost all Android devices have touch screen as their user interface which acts as an interface for getting user input and displaying output . Hence, the Android applications should be capable of handling the touch based user input and process it accordingly. Android has provided several classes and methods which can be used to manipulate the touch events caused in the touch screen such as MotionEvent Objects, Velocity Trackers, special classes for map interfaces and classes that interpret gestures [12, 591].

When a user touches the screen of the Android device, a MotionEvent object is created which contains information about the touch position and other details. This object is

then passed to the appropriate method of the view component of the application [12, 591]. The series of events such as 'ACTION_DOWN', caused due to the initial touch, 'ACTION_UP', caused by lifting the finger up, 'ACTION_MOVE', caused by moving the finger sideways AND 'ACTION_CANCEL', caused if the touch sequence ends without actually doing anything, creates the sequence of events which are stored in the MotionEvent object. This object contains information about the action being performed, position of the touch event, amount of pressure applied, size of the touch area and time when the event is triggered and when it is ended. [12, 592.]

Another approach to handle the touch events is to register a callback handler for touch events on a View object. For this purpose, the View.OnTouchListener interface which consists of the onTouch() and onTouchEvent() methods should be implemented by the class. It should then call the setOnTouchListener() method of the View object to setup the handler for that view. The OnTouchEvent() method of the interface can be overridden to handle the touch event differently by the View object. [12, 592.]

Another class that helps in handling touch screen event sequences is VelocityTracker. This class detects how fast a finger is moving on a screen, which helps in determining what users want to do with the touch. This class consists of methods like getXVelocity() and getYVelocaity() which return the corresponding velocity of the finger in the X and Y direction respectively. [12, 603.] VelocityTracker consumes more memory in an application. Hence, this should be used sparingly and be recycled after using. [12, 604.]

2.6 Android Activity Fragment

Activity is an application component which is usually presented as a single screen and which acts as a window for the user interface. In an Android application, all the View components are placed in the activity. Fragment is an Android component that divides the activity into several fragments. These are the mini activities which are reusable and independent from each other but are related mini portions of the application and screen with their own events, state, application lifecycle and Back stack. This component was introduced in Android version 3, and it was especially designed for tablets although it is also used in normal Android mobile applications. [14, 545-546.]

Fragments have a lifecycle similar to that of basic Activity which is driven by its host activity. Some of the lifecycle methods of Activity and Fragment are onStart(), onResume(), onStop(), onActivityCreated(Bundle) and so on. The similarity between the lifecycle events of Activity and Fragment is illustrated in figure 7.

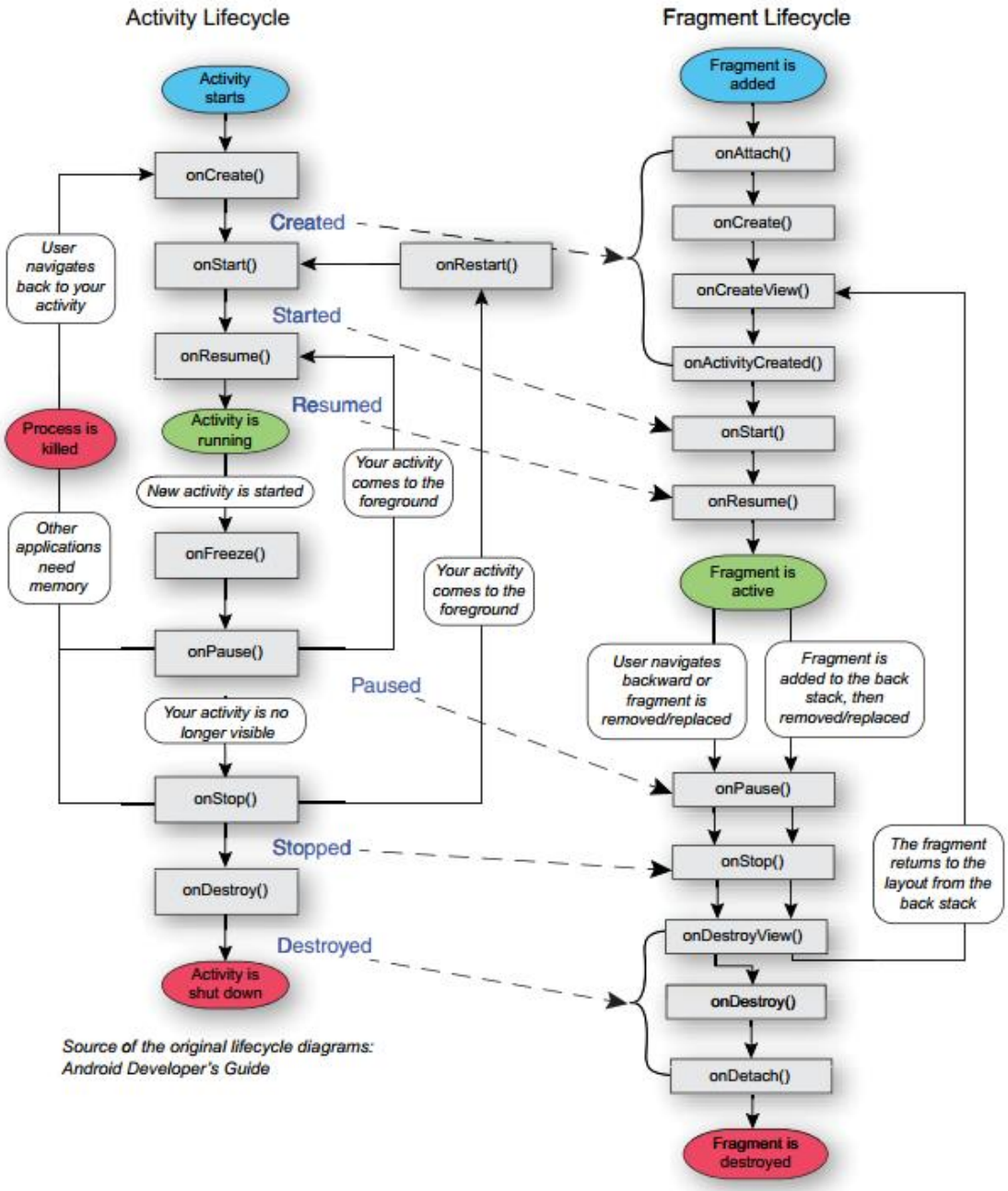

Figure 7. Lifecycle of Activity and Fragment [14, 547]

As shown in figure 7, both Fragment and Activity have a similar lifecycle. However, the lifecycle of Fragment is controlled or monitored by the lifecycle of Activity. The

onCreate() method of Activity initializes some of the initialization methods of Fragment such as onAttach(), onCreate(), onCreateView() and onActivityCreated(). Similarly, other events in the Activity lifecycle invoke similar events in the Fragment lifecycle.

Coding Fragment is similar to coding for Activity. Alike layout for Activity, fragments also require a layout to be created which is then included in the layout of the host Activity. To create a fragment, a Fragment subclass should be created with a separate layout which should be then included in Activity. [548.] Fragments are maintained and managed using Fragment Manager. Fragment Manager helps to find the Activity fragment by a resource ID or by tag, manage Fragment transactions and manage the Fragment back stack. Fragment transaction refers to several fragment operations like adding and removing fragments. This is supported by the FragmentTransaction class which provides methods to hide, add, remove and replace fragments, set animations, add fragments to the back stack and commit the transaction. [14,555.]

Implementing a Fragment is an effective way to design a flexible and efficient user interface. The Fragment acts as a sub-activity that can handle its own lifecycle and back stack. [21.] As this API is implemented only in Android version 3, for lower versions of Android, it is included in the Android compatibility package which is a static library.

3 Web Application Development Overview and Technology Involved

An application which is typically invoked in a web browser over the internet is known as a web application. It is basically comprised of Hyper Text Markup Language (HTML) documents which contain contents to be displayed, instructions to format documents and links to other documents. Based on client server computing, the documents are stored in the server which is accessed through browsers in a client computer. Besides the HTML documents containing contents to be displayed, a web application also consists of client side scripts that run in the client for user interaction and server side scripts performing server side processing and interacting with the database. [22.]

In addition to many server side techniques, PHP Hypertext Preprocessor (PHP) is the basic and most frequently used technique. It is integrated with the HTML tags, enclosed in special tags which are recognized by the server while fetching the HTML documents, and executed, appending the results to the documents. This helps in the creation of a much more dynamic and flexible page with more reusable components. All the heavy weight tasks such as connecting with the database, querying the data, fetching other general and multimedia files are performed by the PHP scripts.

Client side web scripts such as Javascripts are also embedded in the HTML pages and executed in the browser environment to help in the user interaction. The script consists of objects related to user interface entities such as HTML elements, windows, cookies or user events like mouse click, move, Button click and so on. In addition to Javascript, another technique which allows creating smooth and seamless user experience is the use of AJAX which stands for Asynchronous Javascript and XML. It uses an asynchronous protocol for client server communication in which the browser registers the set of call backs for the server to use for updating the browser data in the background. [22.]

The interaction between the client browser and server side components can be done using Hyper Text Transfer Protocol (HTTP) through which the client computer requests data and documents to the server and the server sends the response back. This protocol defines eight basic operations among which GET and POST are the most frequently used ones. Besides the HTTP protocol, nowadays, there are other alternative protocols for the client-server communication such as SOAP (Simple Object Access Protocol).

Besides the above mentioned technologies, there are many more advanced tools and technologies used in web application development. HTML, Javascript and PHP are the basic and most frequently used ones for general purposes and they are also getting advanced with time.

4 Database Design for the Feedback Application

For the project described in this thesis, MySQL was used as the relational data management system. Four tables were created in the database. The designs of the tables are displayed in figure 8.

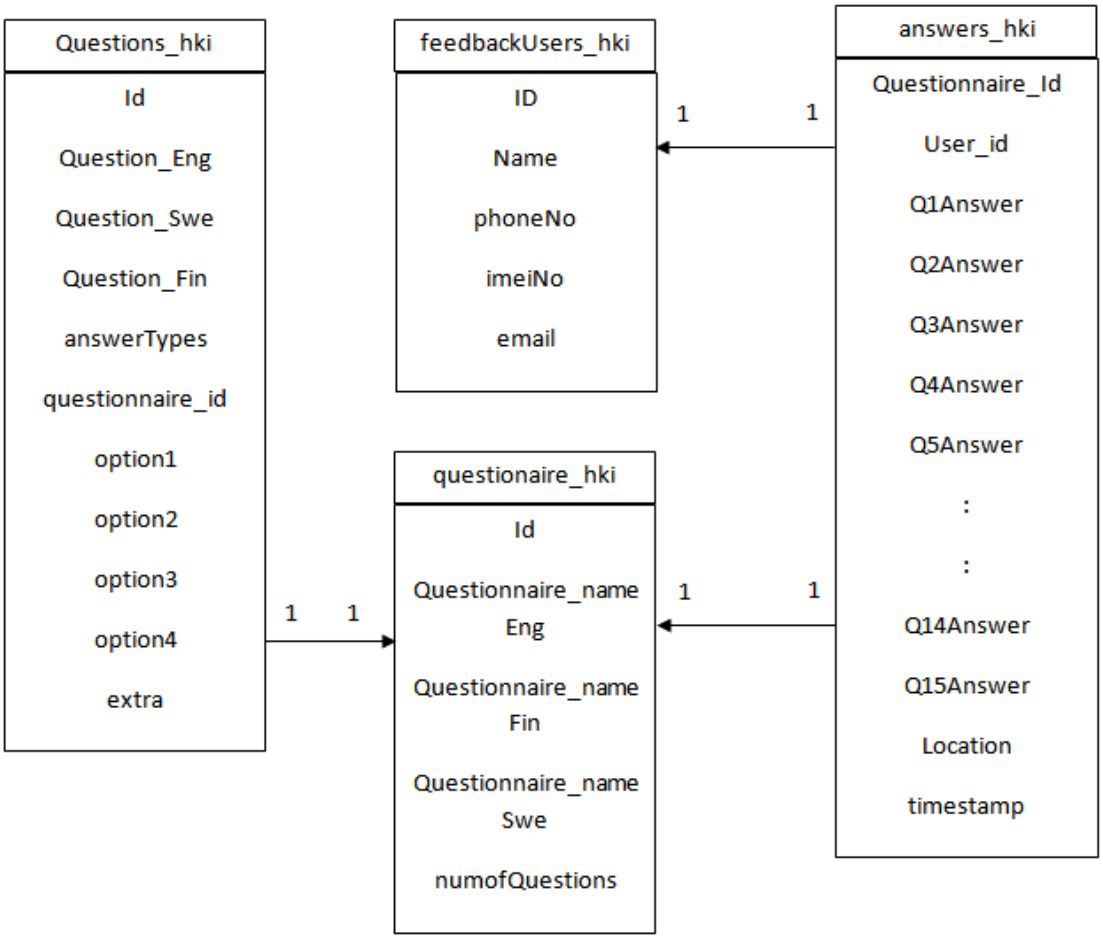

Figure 8. Database Design

As presented in figure 8, the database consists of four tables which are linked with each other in some way. The 'Questions_hki' table consists of questions in three languages, answertype field, questionnaire ID with which questions are related to, four options for the multiple choice questions and an extra field that defines the need for location data. Similarly, the 'answers_hki' table consists of the questionnaire ID of the

question, to which the answers are related, the user ID which is linked with the ID of the user in the user table, fifteen fields for answers and timestamp. As the limit of questions per questionnaire is fifteen, the answer table needs fifteen fields for fifteen answers. This design of the database is not fully optimized as the current design is good enough for the project purposes. The images are stored in a separate folder in the server with their name in the following format: UID_<User_id>QID_<questionaire_id>imageAnswer<number of question in the list>_<Datestamp>_<TImeStamp>.jpeg. In the answer field in the table, the image name is stored, together with the text answers if available.

5 Web Application

5.1 Project Background and Requirement Specification

The web application created in the project described here is used by Helsinki City personnel and acts as an interface for interacting with the database. The application will be able to perform actions such as creating questions and questionnaires, storing these data in the database, retrieving required information, modifying the data and exporting required pieces of it from the database in a file. The questions are grouped together according to their category or topic called 'Questionnaire'. Under a questionnaire there can be one or more than one question. The maximum number of questions under a questionnaire is fifteen. The questionnaires might contain different types of questions which require different types of answers such as simple textual answers or multiple choice options and image and location data. The use case diagram of the web application that defines its requirements is illustrated in figure 9.

Figure 9. Use Case Diagram of the Web Application

As shown in figure 9, the web application should be able to create lists of questions under a questionnaire and a feature should be provided so that each question could have a required answer type. The questions will be presented to the general public thus requiring the preparation of the questions in three different languages which are English, Finnish and Swedish. In addition to retrieving the questions and answers from the database and displaying them, the application should provide an interface with which users can modify and delete the question contents and extract the answers of a specific questionnaire and save them in a file for further analysis.

5.2 Design and Implementation of User Interface

The web application simply consists of a few html pages; PHP was used for interacting with the server and Javascript for the client interface. PHP script is integrated with MYSQL to interact with the database. As defined in the requirement section above, the main purposes of this web application are to create questions and questionnaires, modify and delete them, and view and extract answers from the database. Hence, there are four HTML pages which serve as an interface for each of the above-mentioned purposes including the Help page.

As the application is meant for creating questions in languages other than English, it needs to support the Scandinavian characters to be displayed properly. The character encoding used in the html pages is 'UTF-8'. When any data sent from the application needs to be inserted into the database, they are decoded before the insertion.

```
$questionaireNameEng=
      utf8_decode(mysql_real_escape_string($_POST['questiona
      ireNameEng']));

$questionaireNameSwe=
      utf8_decode(mysql_real_escape_string($_POST['questiona
      ireNameSwe']));

$questionaireNameFin=
      utf8_decode(mysql_real_escape_string($_POST['questiona
      ireNameFin']));
```

Listing 2. PHP code for UTF-Decoding and Text Escaping

As listing 2 illustrates, any data posted from the web application are decoded before being inserted into the database. This ensures the integrity of the data when they are extracted from the database to be displayed in the application. Similarly, any data users feed which needs to be inserted into the database should be escaped properly as shown in listing 2, to prevent several security vulnerabilities.

In the web application, **Add Questionnaire** is a page on which questionnaires and questions are created and submitted into the database. The options to create the required number of questions according to the questionnaire, to feed the questions and the name of the questionnaire in three different languages and to link the type of answer required to each question are provided in this interface. The flowchart in figure 10 describes the work flow during the process of the questionnaire creation.

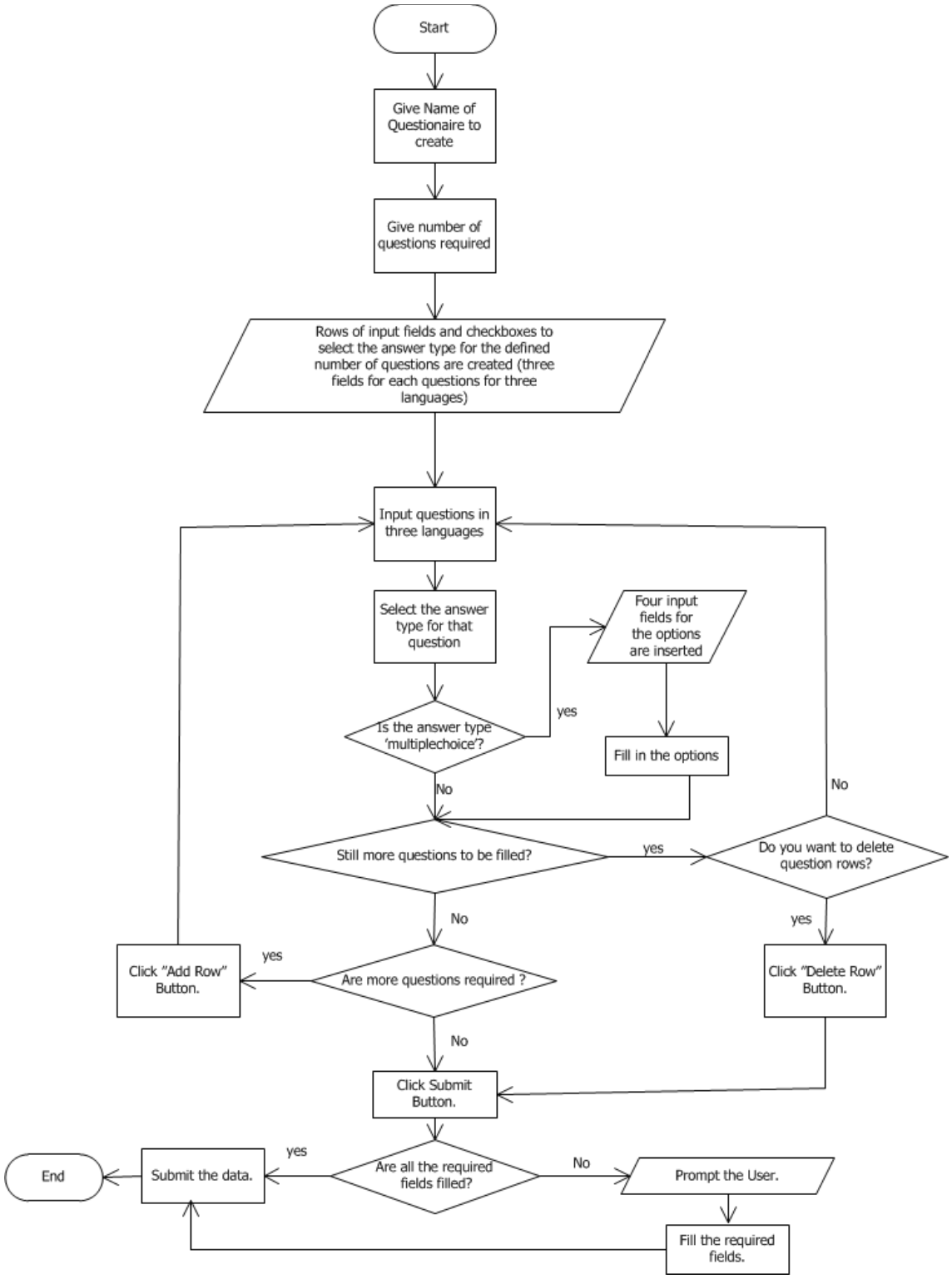

Figure 10. Flowchart of Actions Performed in 'Add Questionnaire' Page

In figure 10, the process of creating a questionnaire on the Add Questionnaire page is illustrated. Users input the number of questions related to a given questionnaire which will create a row for each question that includes three input fields for three language options and four options to select the type of answers for a specific question. Figure 11 shows a sample row that refers to the row of question field in the web application.

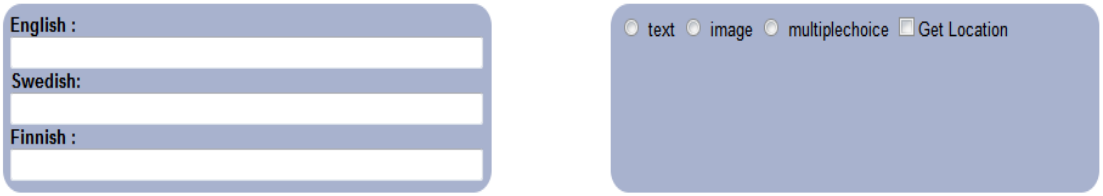

Figure 11. A Line Demonstrating Creating a Question

As shown in figure 11, after feeding the questions into respective input fields, the answer type for the question needs to be selected. One option can be chosen, the options being 'text', 'image' and 'multiple-choice'. If the user chooses the latter option, four input fields to enter the four options for the multiple choices are dynamically inserted into the same box. The 'Get Location' option can be checked or unchecked depending on the requirement of the answer to the question.

If one wants to delete or add a row to a question, there is an option to perform that action. After all the questions are created, they are submitted to the database with the submit Button. Figure 12 exhibits a sample of the user interface to create the questionnaire.

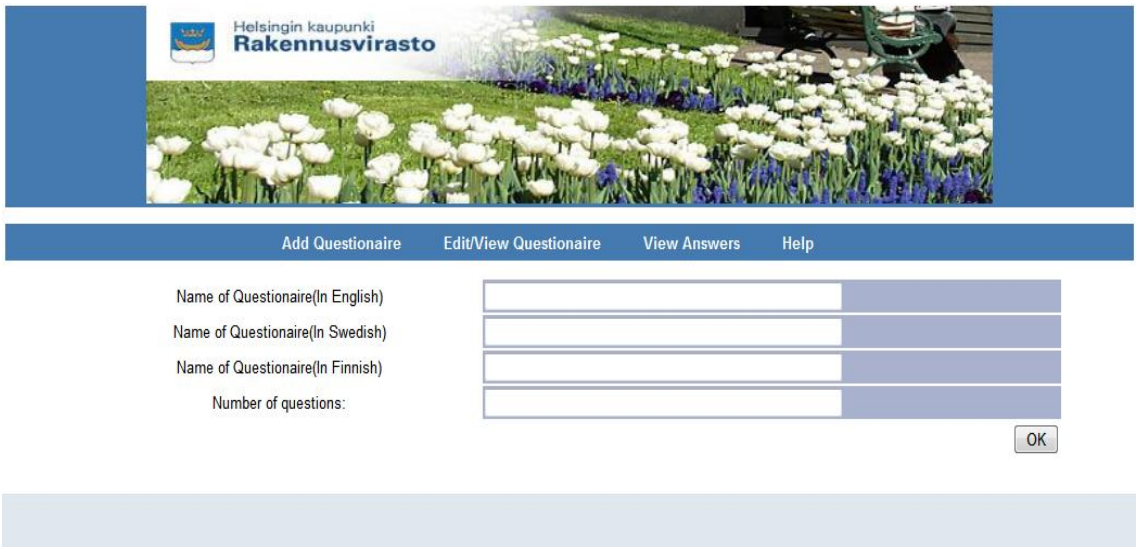

Figure 12. UI to Create Questionnaire

As illustrated in figure 12, there are input fields for entering the name of the questionnaire in different languages and the number of the related questions. When one clicks on the 'OK' Button, respective rows for creating the questions are created dynamically using the Javascript. Figure 13 displays the appearance of that user interface after clicking on the 'OK' Button.

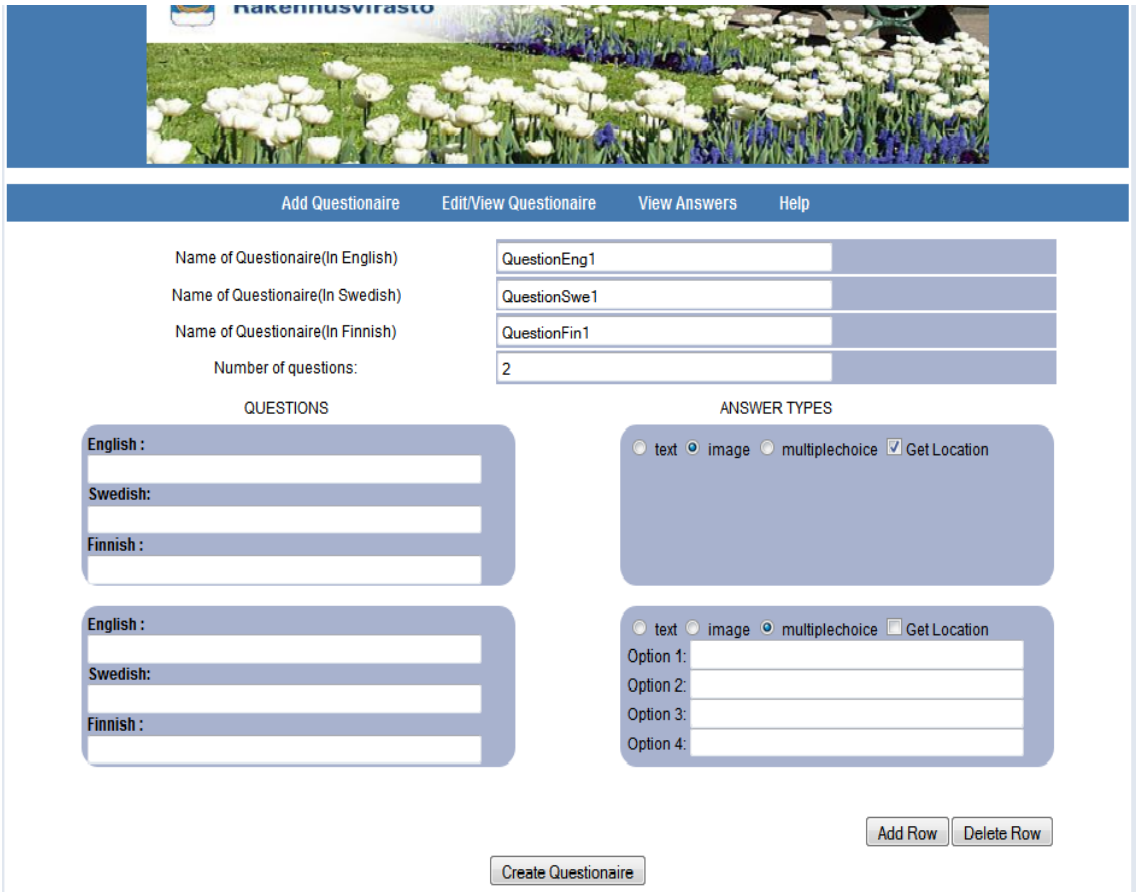

Figure 13. Interface for Creating Questions and a Questionnaire

In figure 13, the user interface after feeding the number of questions to the application is displayed. A row is created for each question including the helper buttons to add or delete the rows as required. Besides displaying the interface for creating the questions, this page also consists of links to navigate around other pages. This page is also used as an index or the main page of the web application. The input fields are integrated with a mechanism to check the data fed by the user to validate the data type and availability. When one clicks on the 'Create Questionaire' Button, the system checks that all the required fields are filled in and valid. After this validation, the data are sent to the server using the HTTP protocol. In the server side, the data are manipulated by the PHP script and stored in the database.

Edit/View Questionnaire is another page which serves as an interface for viewing the available questionnaires and related questions. When a user navigates to this page, the question data are fetched from the database and rendered on this page in the form of a table. The interface of this page is presented in figure 14.

Figure 14. A UI of 'Edit/View Questionaire' Page

As shown in figure 14, the available questionnaires and the number of questions included are presented to the users in a table. On the same line, there are two additional buttons: one allows viewing the lists of questions of that questionnaire and the other allows deleting the respective questionnaire. If one clicks on the 'Delete Questionnaire' Button, he/she will be prompted to confirm the action, and if confirmed, the selected questionnaire and questions related to it will be deleted from the database. The change will be rendered in the interface as soon as the action completes.

If the 'View' Button, linked with the questionnaire is clicked on, the related questions are fetched from the database and displayed in tabular form. This interface is displayed in figure 15.

Figure 15. Sample Question Table in Web Application

As shown in figure 15, the lists of questions in all three languages and additional information about the type of the answer required are displayed in a table. On each line, there are buttons which allow deleting and modifying the contents of the question, if clicked. If the 'Delete' Button is clicked on, the specific question is deleted with all the values related to it. This will also decrease the number of questions in the related questionnaire by one. The 'Edit' Button will change the view, presenting specific question values in a form which can be altered. Figure 16 displays the interface during this process.

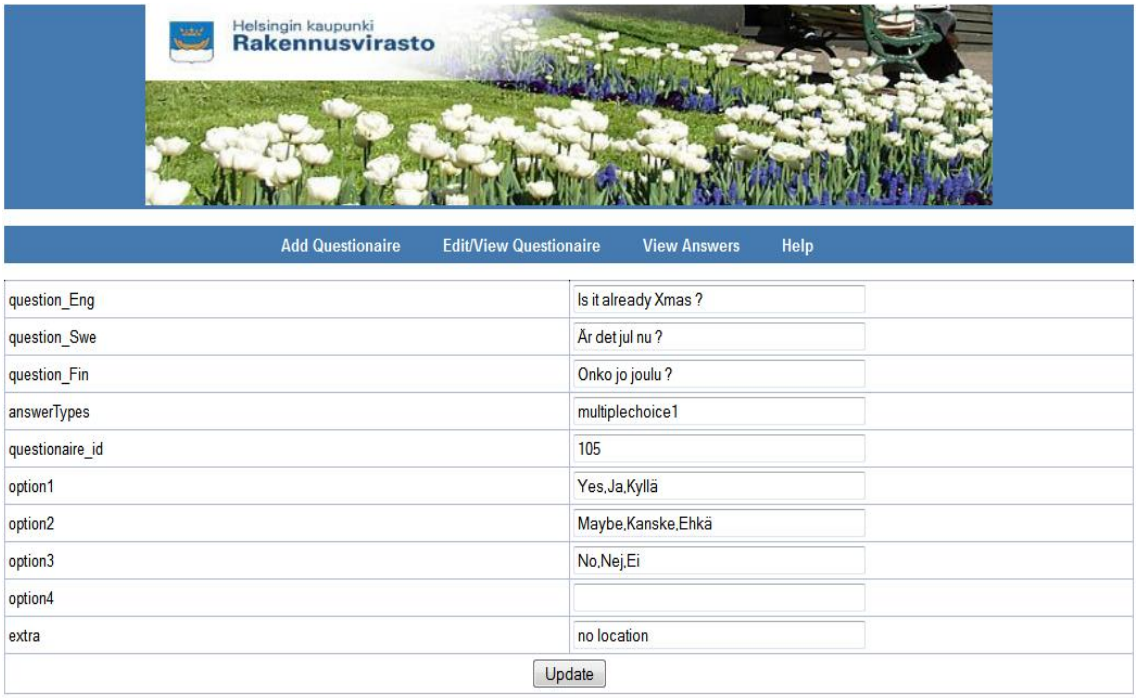

Figure 16. Interface for Editing the Question Data

As illustrated in figure 16, while editing the question data, they are presented in a form which allows the values being easily changed and updated in the database.

As the name suggests, the **View Answers** page retrieves the answer data from the database and displays the answers in the interface in a tabular form for each questionnaire. These answers are submitted by the Android application users and are saved in the database. Each table represents the answer data of the questions of specific questionnaires. Each line of the table displays the answers of the questions of the specific questionnaire submitted by a particular user. Hence, the first column consists of the names of the users and the remaining column header consists of the questions of the questionnaire. Figure 17 displays the view of this page.

Add Questionaire | Edit/View Questionaire | View Answers | Help

Select A Questionaire personal Info Get AnswerFile

Questionaire: personal Info / Personåås / Persönätiööö

Users	Where do you live??	Take picture of your favourite place.
Anonymous	Image captured; Click here to change. No Location Selected	No Location Selected IMAGE: UID_54QID_103imageAnswer2_121511_19:35:34.jpeg
Anonymous	Helsinki No Location Selected	No Location Selected IMAGE: UID_48QID_103imageAnswer2_121211_19:07:50.jpeg
sabisamsng	Vantåååä Vålaxvägen 786 07450 Porvoo	Vålaxvägen 786 07450 Porvoo IMAGE: UID_43QID_103imageAnswer2_120911_16:44:42.jpeg
mesmsung	Helsinki Neulapadontie 1 00920 Helsinki	Neulapadontie 1 00920 Helsinki IMAGE: UID_42QID_103imageAnswer2_120811_22:46:28.jpeg
tst	Helsinki Säteribrinken 19 02600 Espoo	No Location Selected IMAGE: UID_40QID_103imageAnswer2_120811_15:27:27.jpeg
naveehtc	Vantåååä Grindåkersstigen 02650 Espoo	Gransbacken 6 02650 Espoo IMAGE: UID_39QID_103imageAnswer2_120811_15:20:55.jpeg
devsamsung	Vantåååä No Location Selected	äöåäöåäö No Location Selected IMAGE: UID_31QID_103imageAnswer2_120811_14:55:05.jpeg
devsamsung	Espöö Gransbacken 6 02650 Espoo	No Location Selected IMAGE: UID_31QID_103imageAnswer2_120811_14:53:22.jpeg

Questionaire: XmasTestTwo / JulTestTvå / JouluTestiKaksi

Users	Is it already Xmas ?	Where does Santa Claus live ?	Take a winter picture
			ok No Location Selected IMAGE:

Figure 17. A Section of the Interface for Viewing the Answer Data

Figure 17 displays the interface for viewing answers in the web application. The interface consists of lists of tables for each questionnaire which includes questions as a column header, and the answer data are presented on a row submitted by a particular user. Besides just being able to view the answer data, there is also a feature which allows exporting the data in a file, specific to the questionnaire. In the top section of the web page, there is a drop down list of available questionnaires which can be selected by the users, allowing them to create the related answer file. The extracted file will be in the Excel file format and it consists of tabular answer data of the selected questionnaire.

6 Android Application Design and Implementation

6.1 Requirements Specification

As mentioned in the introduction to this thesis, the Android application is a general feedback system which fetches the questions from the database, displays them to the users according to the type of question and sends the answer fed by the users to be saved in the database. The application should be a localized application which means that the language of the application should be specified according to the language the device is using. Figure 18 illustrates the actions which are performed by the Android application and the features it requires to access from the device.

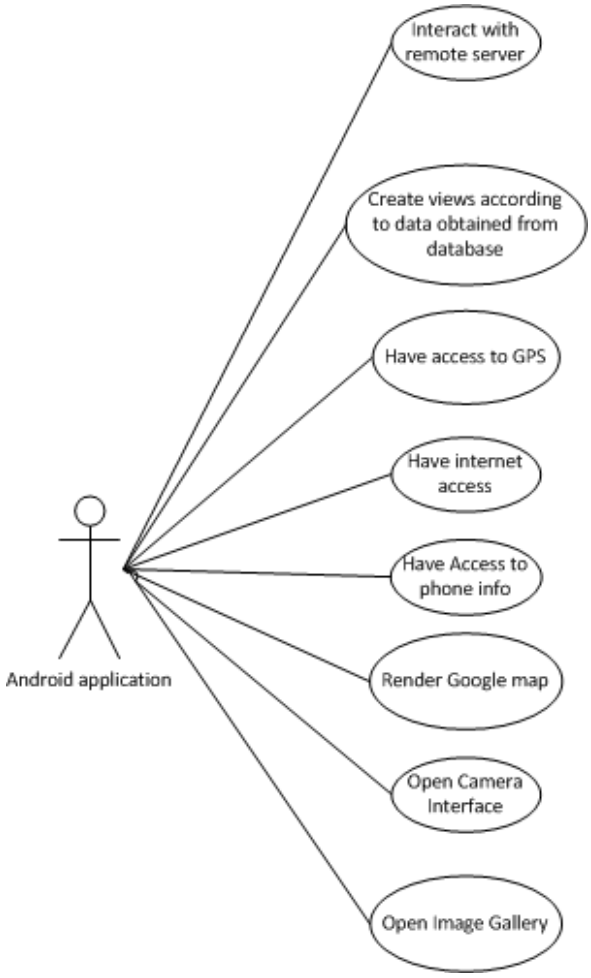

Figure 18. Use Case diagram of Android Application

As illustrated in figure 18, the Android application should be able to access the GPS to get the location information and also the camera and gallery of the device which are required to get the image data to be fed as an answer. In addition, the application should be able to check the network connection of the device to connect to the remote server. The International Mobile Equipment Identity (IMEI) number of the device is also accessed by the application to be submitted in the database, which is used to identify the users. The user interface of the application should be created depending upon the type of answer required by the question. For instance, if the answer type of a question is Image, the interface should include two buttons for retrieving the image, one for using the camera of the device and the other for selecting an image. Similarly, when the answer of a question should include a location, the interface should be integrated with a button which should detect the current location of the user when pressed and prompt him/her to confirm the accuracy of the location. If not accurate, a map view of the detected location is displayed to the user for him/her to select the accurate location.

By using this application, users should be able to view the questions and answer them. The use case diagram in figure 19 displays different actions a user can perform through the application.

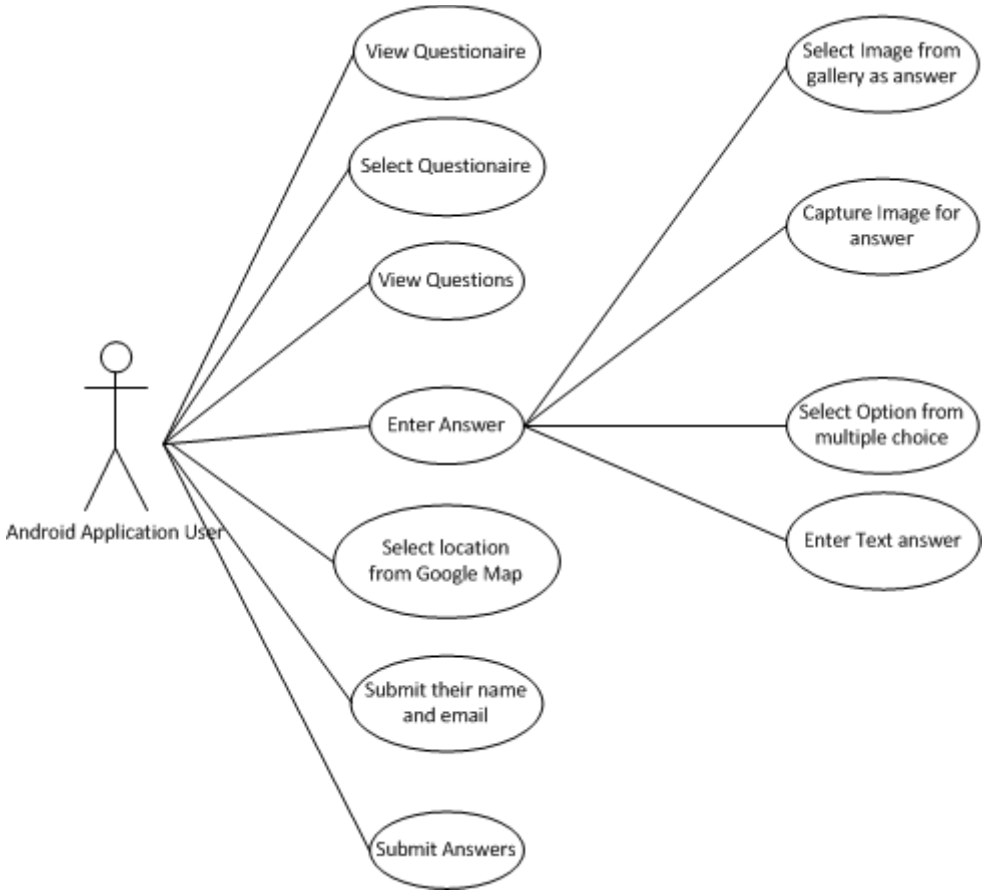

Figure 19. Use Case Diagram of Android Users

As illustrated in figure 19, users of the application should be able to view the questionnaires and select them to answer the related questions. They should be able to answer the questions in the form of text, location or image. For the location selection, users should be able to use a map in the user interface. Before submitting the answers, the option to include one's name and email together should be provided to the user which can be optional as well.

6.2 Architectural Design

The Android application for this project has been designed as a client application which is used as a user interface. It can communicate with the database located in the servers. The designed workflow of the application and its states in the process of its use are presented in the activity figure 20.

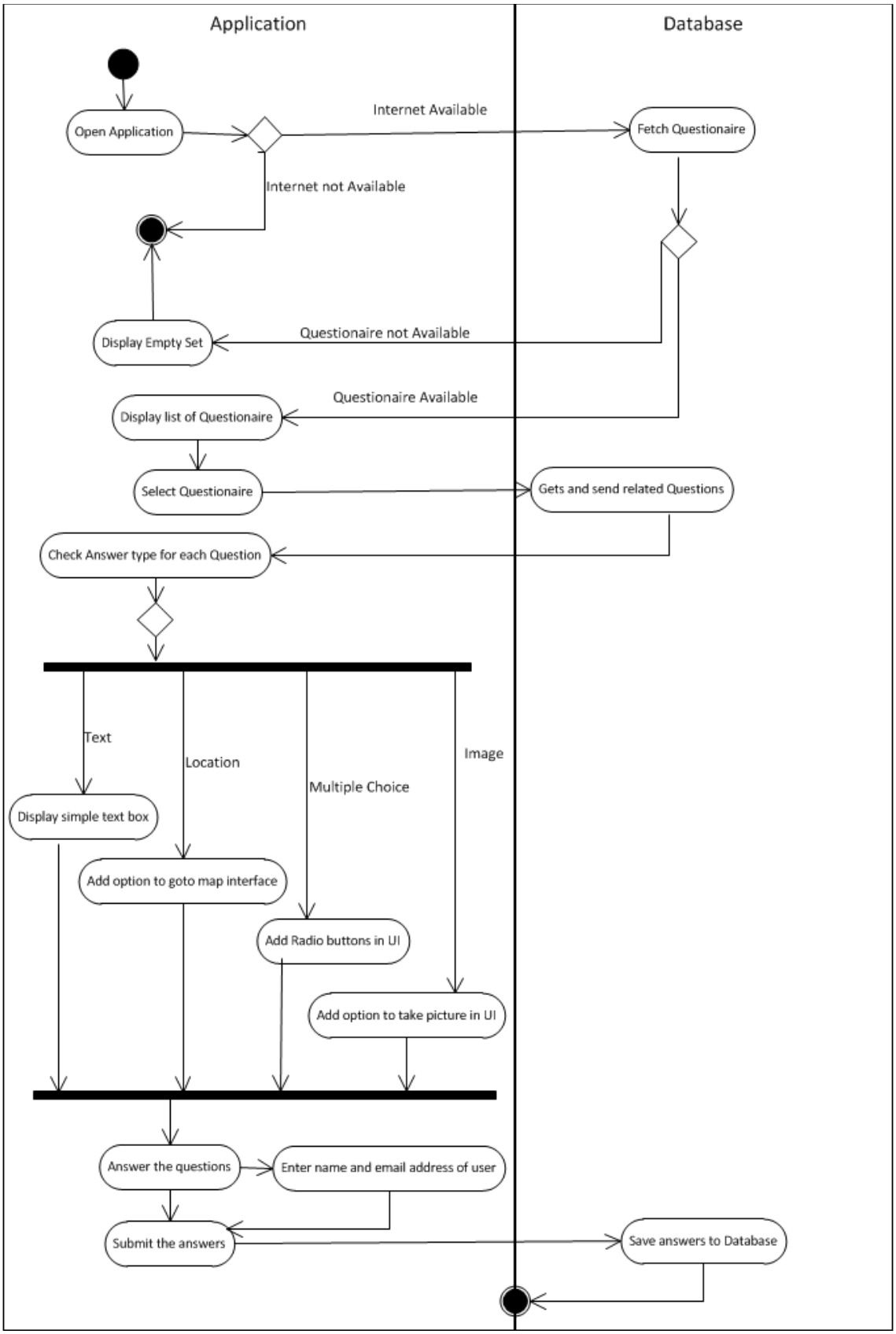

Figure 20. Activity Diagram of Android Application Usage

Figure 20 illustrates the sequences of possible actions performed including its start and end state within the application and database. When the application is started, the availability of the internet is checked. If available, the application gets connected with the database and the data are retrieved whereas if not available the application goes to the end state. After successful retrieval of the data, the questions are displayed in the user interface according to the answer types. Along with the states of the application, the figure demonstrates the typical workflow during the system usage.

As the Android applications are created using the Java programming language, the application developed in this project consists of the Java objects the instances of which are created after retrieving the question data from the database. The application is designed with four Java objects which are 'Questionnaire', 'Question', 'Answer' and 'HttpHandler'. The class diagram presented in figure 21 describes the properties and method of these independent classes in more detail.

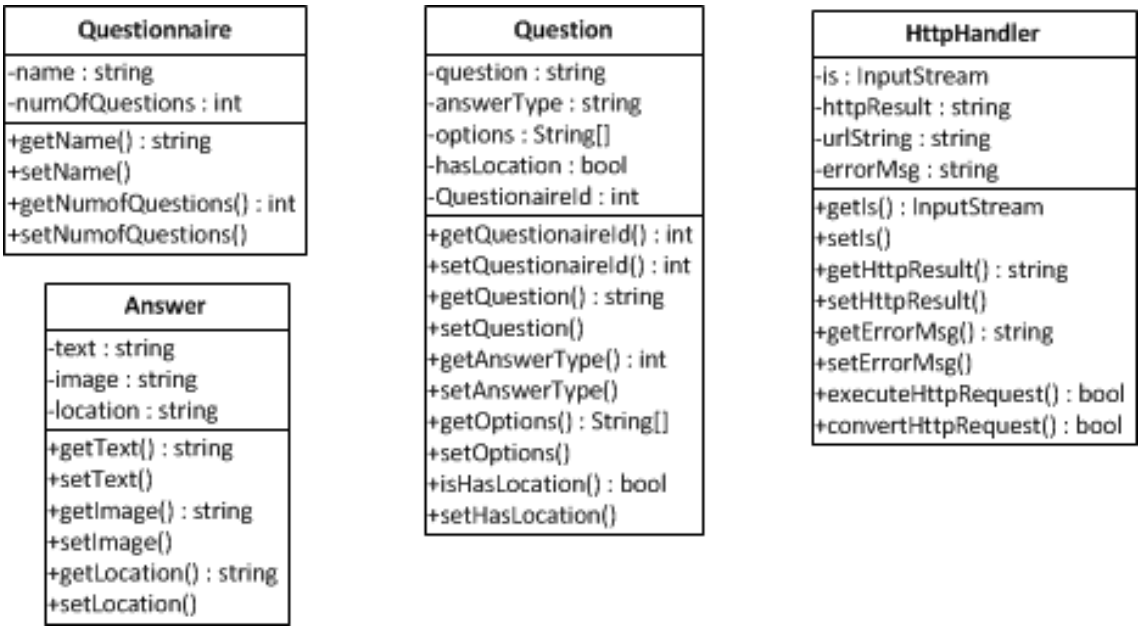

Figure 21. Class Diagram of Java Object in Application

Figure 21 displays the design of classes used in the application. The instances of the classes 'Questionnaire' and 'Question' are created with the data retrieved from the database and used to store this information. These instances are used to create TextViews respective context. The answers given by the users are used to initialize the instances of the 'Answer' class which stores the values till its submission. The 'HttpHandler' class is a simple class that consists of the methods connecting the application with the database and exchanging the data using the HTTP protocol. The definitions of these classes are presented in appendix 1.

Apart from these classes, the application also consists of other third party classes which are used as utility classes to assist in the user interface design and also to help perform the application functionalities. These different classes cooperate and interact with each other in all operations of the application.

6.3 Design Implementation and Development Technologies

The application for the Android platform was developed in Eclipse Integrated Development Environment (IDE), integrated with Android Software Development Kit (SDK), using the Java programming language. The communication with the server was performed by a simple HTTP protocol in which the server side script is in PHP. The PHP script uses MYSQL language to query the database to manipulate the data. This application has been designed in such a way that all data sent to the server should be in String form. Hence the image data, which can be attached as an answer by the user, is also encoded into String using a helper class called 'Base64.java'. The question and questionnaire data are transferred from the server to the application in the form of JSON and are parsed accordingly when received by the application.

As mentioned in the web application section, the questions and questionnaires are stored in the database in three languages. When sending the request for these data, the application extracts the locale information being used by the device and sends this

information along with the request. This information is obtained by calling the getDisplayLanguage() method of the 'Locale' object of the Java.util package. This information is then used to query the data in a specific language so that there would not be any extra work for extracting it in all the languages.

The list of questionnaires and the questions are displayed as two separate Activities. After the questionnaires are obtained, they are used to create a custom adapter of the ArrayAdapter class which consists of lists of View objects, composed of TextView and ImageView. This list is rendered in the activity that extends the ListActivity class. The implementation of this custom adapter is presented in appendix 2. Users can select any questionnaire from the list, which is then used to query the related questions from the database. The list of questions is then rendered in the activity that extends from FragmentActivity. Each question is displayed in a separate fragment of Activity. The fragments and all the views for the fragments are created dynamically during runtime according to the number of questions obtained and the type of question data.

The question data are used to create instances of the 'Question' object the properties of which are used to create the type of views being rendered in the screen. In each fragment there is a standard TextView for the question and an EditView for entering a textual answer. Then, according to the other properties of the 'Question' object, other views are added. If the 'answertype' is 'Image', two buttons to access Camera and Gallery are inserted and if the value of 'haslocation' is set to true, a button to access the GPS feature of the phone is added in the fragment. Similarly, if the answer type is 'MultipleChoice', the 'options' property of the 'Question' object is initialized and a list of RadioButtons is added in the fragment with the initialized option value. In the last fragment of the activity, there are additional EditViews to provide the name and email address of the user and a button to submit the answer. Clicking on the 'Submit Button' initializes an event the purpose of which is to extract the answers fed by users to create an array of the 'Answer' object. This array is then sent to the server to be stored in the database.

Accessing the device features like Camera, Gallery and GPS from the application is performed by using Intents inside the application. The buttons to access those features are implemented to listen to the click event which starts specific intents when fired. When the button to capture Image is clicked, Camera Intent is called. This starts the camera, with which the user can take a picture to be included as an answer. If the button clicked is to select an image from the Gallery, the Gallery is opened for the users to select the desired image. After the image data are taken, they are converted into String type and stored in AnswerArray. When the 'Get Location' Button is pressed, using the GPS feature, the current location information is extracted and presented to the user to be confirmed. If the location is not accurate enough, a MapView referring to that location is displayed from which the specific location can be selected. The MapView is implemented by creating a custom class that extends the MapActivity class, which consists of several helper methods which are meant to interact with the view. Appendix 3 presents the class definition.

6.4 Graphical User Interface

The user interface of the application is designed using two activities for the list of questionnaires and questions separately. Questions are divided into separate fragments of the activity for each question. A screenshot of the lists of questionnaires in the screen is presented in figure 22.

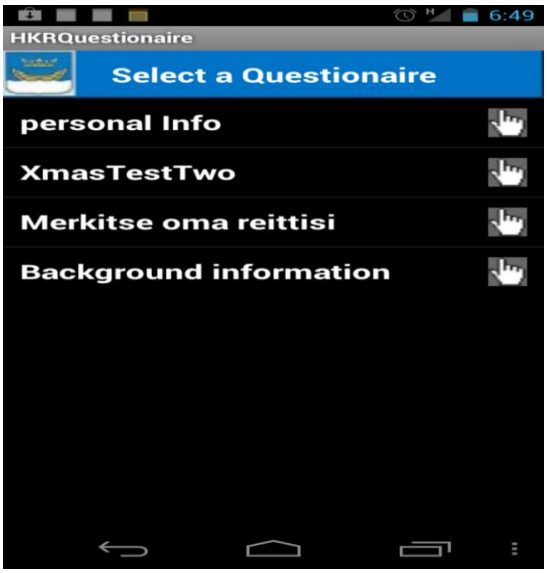

Figure 22. User Interface with List of Questionaires

As shown in figure 22, each row of the list consists of the name of the questionnaire on the left and of a small image of a finger representing touch on the right part of the screen. The finger image instructs users to touch the questionnaire to select it. After the questionnaire is selected, the screen is switched to another screen displaying the first question of the question list of the questionnaire. Figure 23 displays a sample screen of the question displayed in a fragment activity and the sections on the screen.

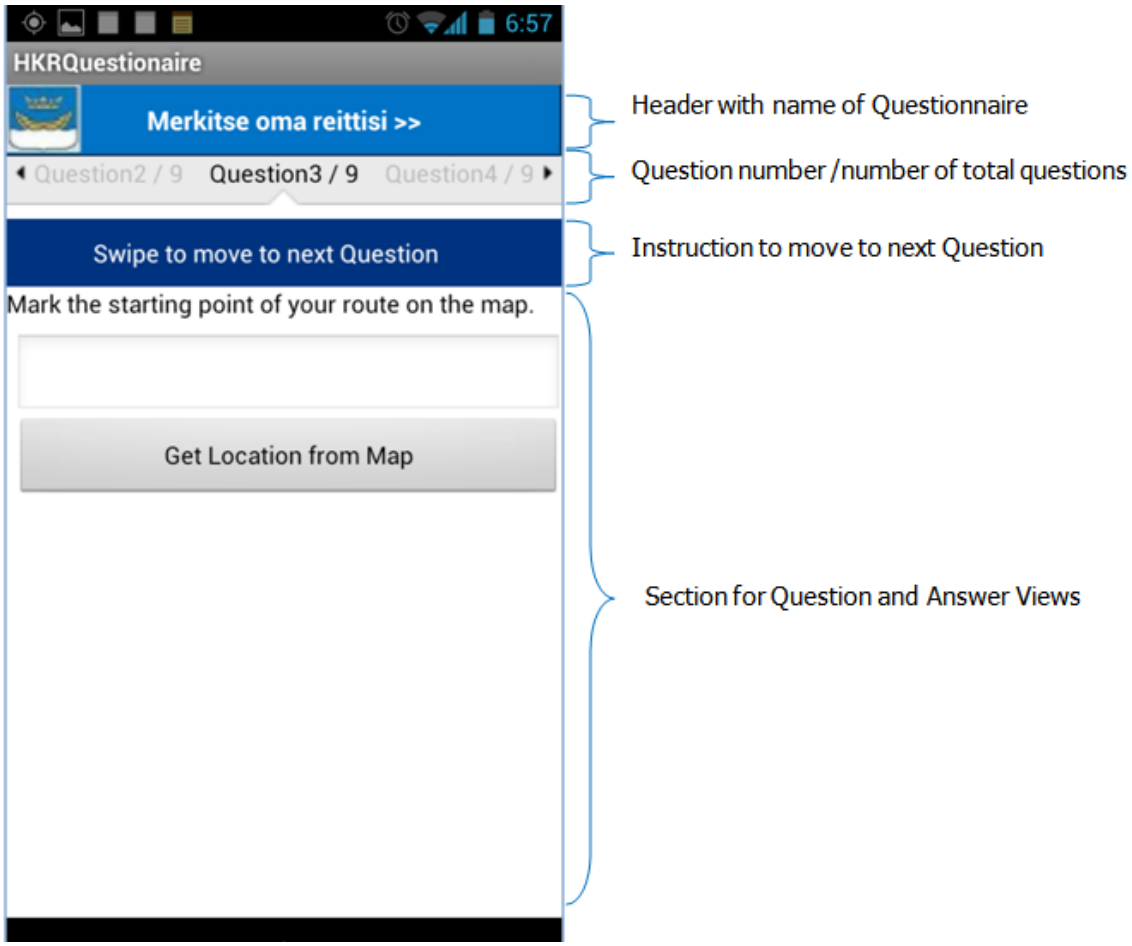

Figure 23. Sample Question Screen

As illustrated in figure 23, a question fragment consists of only one question. On the screen, other information such as the name of the related questionnaire, number of question in the list out of the total number of questions and the instruction on how to move to the next question is also clearly presented. Figure 24 displays the swiping of one question screen to another question screen.

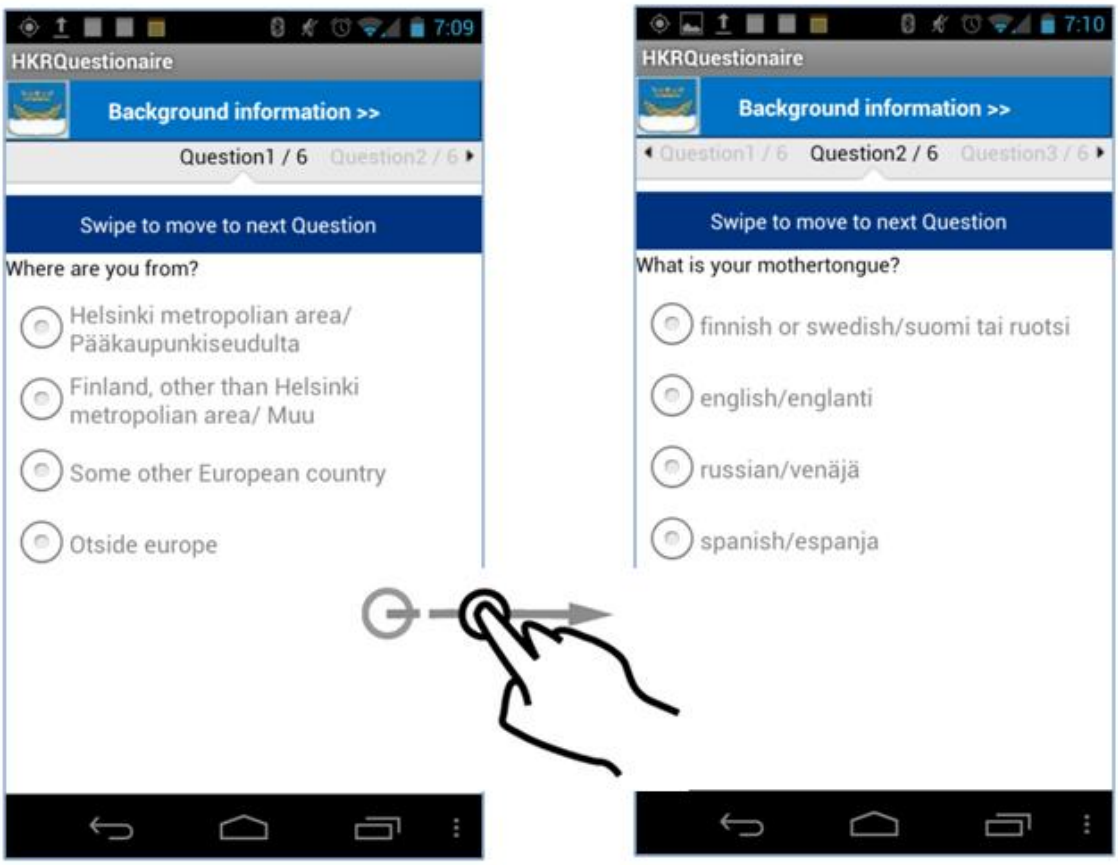

Figure 24. Screens Displaying Question and their Transition - Modified from Gesture: One flick [23].

As illustrated in figure 24, to move across the question screen, users should swipe from right to left or vice versa. The figure also displays a sample of the screen with a question that requires users to select an option from the Radio Buttons. Figure 25 displays the screen with questions that require image and location as an answer.

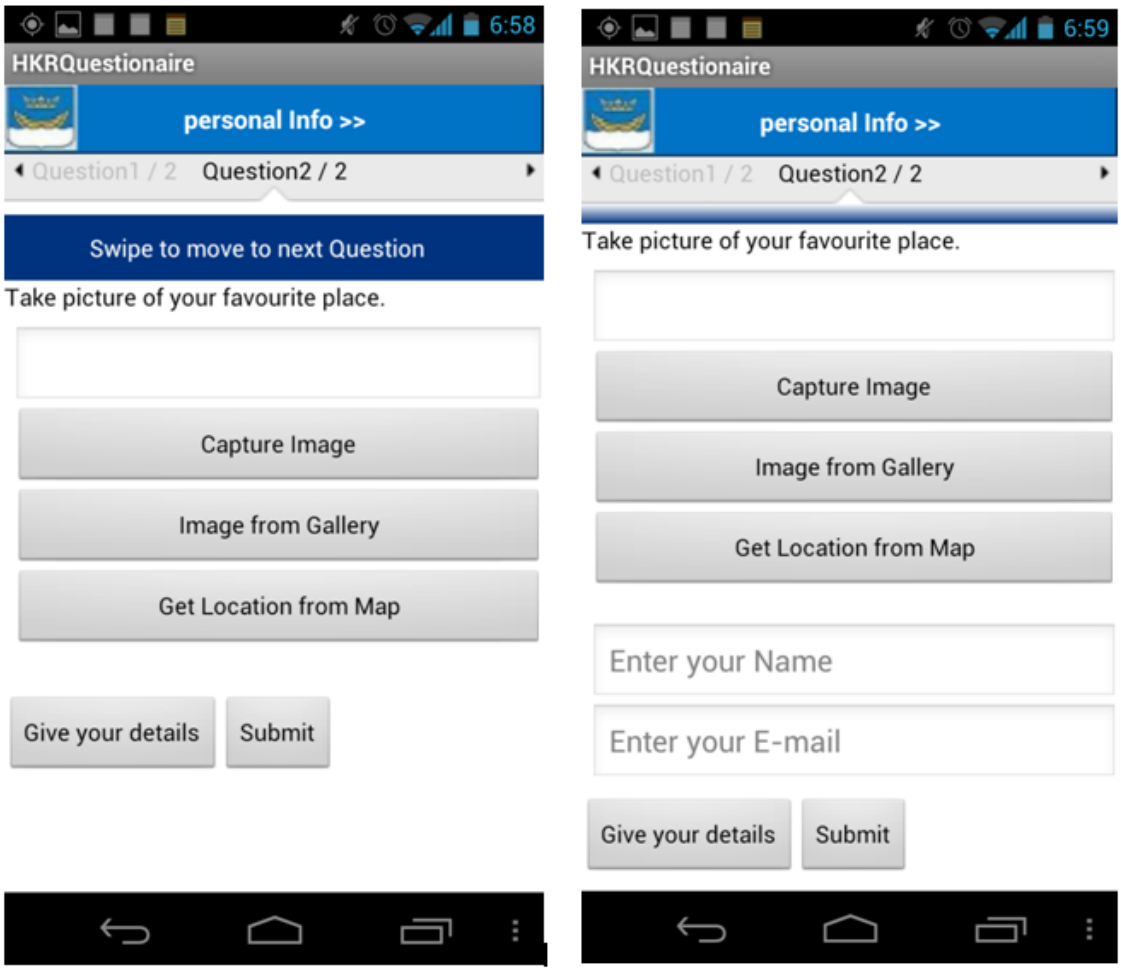

Figure 25. Screens Displaying Additional Buttons

Figure 25 is a sample of the last question screen where the additional 'Give Details' Button and 'Submit' Button are integrated. When a user clicks the 'Give Details' Button, two textboxes to enter the user details are appended in the interface as shown in the right screen of the figure. The 'Capture Image' Button starts the camera of the device while the 'Image from Gallery' Button navigates to the Gallery of the device from where the user can select the image. Similarly, clicking on the 'Get Location from Map' Button triggers the methods to extract the current GPS location data which is then prompted to the user for confirmation. Figure 26 presents the involved process.

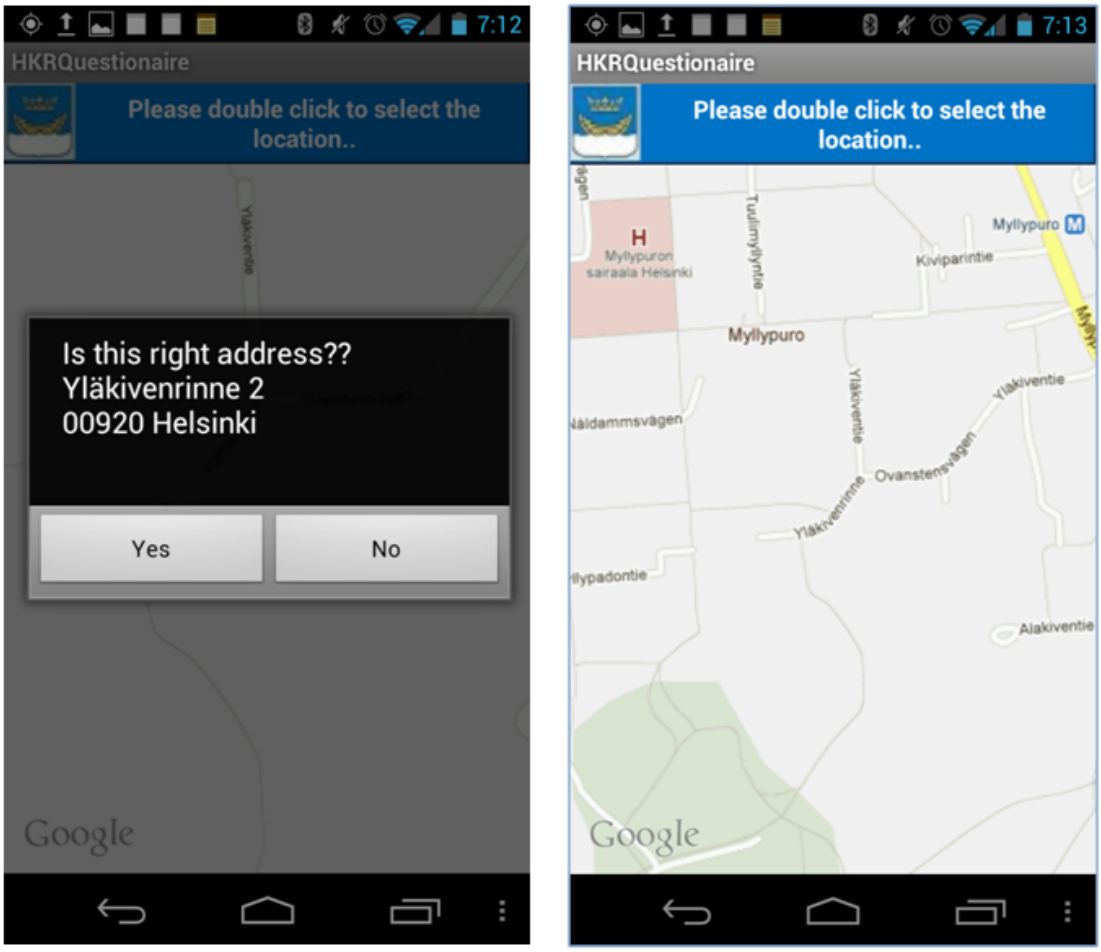

Figure 26. Screen Displaying Location Extraction

As shown in figure 26, users are prompted with the current location data for confirmation, after clicking on the button to get the location. If the extracted location is not accurate, and option 'No' is selected, a map view of the respective location is displayed for the users to select the accurate location on the map.

In addition to the presented interfaces, interaction with the users is also performed by the use of dialogue boxes and toasts. For seamless user experience, any possible errors are handled carefully and users are notified about them. Instructions to navigate around the screens have been provided properly in all three languages and there is

also a menu item where required instructions have been provided. Figure 27 illustrates the provided menu titles.

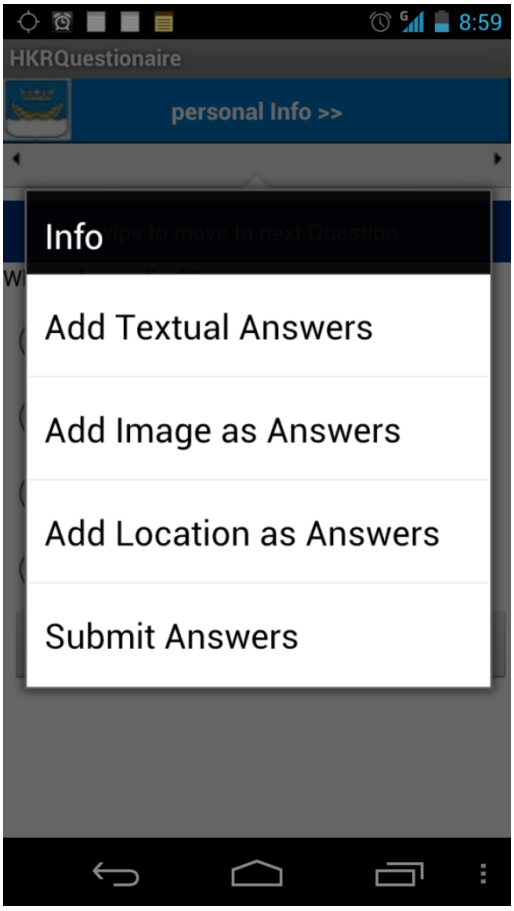

Figure 27. Menu Screen

As illustrated in figure 27, the instructions for answering a question have been separated into specific titles and can be easily accessed by clicking on the desired option.

7 Testing

To test the usability of the application and to get feedback on it, a testing session was organized by Mrs. Anu Kiiskinen in a meeting room in a restaurant located in Helsinki, Finland, where her colleagues from the work place were invited as test users. It was a group session and the participants were from different fields creating diversity in the user group. Mrs Kiiskinen had prepared a set of questionnaires and related questions using the web application for the testing purpose. All the devices used for this purpose were HTC Wildfires on which the applications had been installed beforehand.

I started the session by explaining the purpose of the Android application and reasons for its development. I also demonstrated how to use the application in general and how to interact with the user interface. It appeared that none of the participants had used an Android device before. Hence, it took some time for them to understand the interface of the device. The actual testing of the application was performed outdoors because of the nature of the questions such as the necessity of some specific image as an answer. The participants were divided into groups of 2-3 people, and they were scattered so that they could use the application on their own.

While testing the application, the application crashed during its usage with some participants because of a bug. Due to this reason, very few participants were actually able to submit the answers successfully. The cause of the bug was later found out to be the improper handling of the fragment activities. Due to this reason, when the users tried to move back and forth between the question fragments – which they frequently did - the stack of the fragments was not properly stored in the memory, causing the whole application to crash. However, this was properly fixed later. During this session, there was only couple of useful feedback data from the participants such as the suggestion to change the length and size of the instructive toast messages and to display the map of the current location address instead of hard coded coordinates. These were also implemented in the final version of the application.

Apart from the above mentioned testing session, some other minor testing was also performed in the Public Works Department office of Helsinki City where the participants consisted of people working in the Helsinki City departments. These minor tests were performed mostly to get feedback of the Android application and the web application which helped a lot in their proper development.

8 Results

After the project, a mobile application for the Android platform capable of performing all the desired functions has been created along with the web application to handle the database. The mobile application was developed for Android version 2.2. Hence, it is compatible with devices with Android version 2.2 or above. The application is capable of connecting with the database, retrieving the data and displaying it in the views which are created during runtime, on the basis of the types of data to be displayed. It is also localized which means that the resources and language used in the application are based on the device locale configuration. Users can fill in the text answers, use the camera and gallery of the device within the application, get the location data, and submit the answers to be stored in the database.

The user interface has been designed considering the wide range of users who will be using the application as most of the targeted users of the application might be new to the Android platform. The application is capable of handling most of the error situations which might occur during usage such as unavailability of the internet, location data and much more. However, there are several limitations due to the device itself and other factors which cannot be eliminated. For instance, fetching the GPS information might depend on the location and network connection. The fetched GPS information might also be stored in the cache of the device which might provide stale location information.

The quality of the image captured in the application cannot be maintained when sending the data to the remote server due to the larger file size. Hence, the data quality has to be compensated to be able to successfully store the data in the server. In addition, as all the view components of the application are created dynamically, to save those instances easily and to make the implementation simple, only the portrait orientation of the application is supported.

Similarly, Android devices of different brands might consist of different features causing the application to behave differently. The application has been tested in a few devices which are HTC Wildfire S, Samsung Galaxy I9000, Samsung Galaxy I9100 and Samsung Galaxy Nexus. As an instance of the incompatibility of the application in different device models, it was observed during testing that capturing image and getting images from the gallery is handled differently in HTC devices and Samsung Galaxy devices, causing the application to crash in the latter model. Hence, the implementation had to be altered to make it work in both devices. However, that still caused the application to crash in Samsung Galaxy Nexus which was then ignored as the application ran properly in other Samsung devices.

Similarly, the web application capable of handling the data in the database is also developed using basic HTML, PHP and Javascript. When using the application to create the questionnaires and questions, it has to be kept in mind that users are responsible for filling in all the required data properly and according to the instructions. As the interface of the mobile application depends on the data fed into the database, proper attention is needed when creating that information. For example, 'answer type' should be properly chosen according to the question requirement. Similarly, the options in all three languages should be provided according to the instructions when the question is of the multiple option type.

Thus, despite of several limitations, the final product of the project fulfills most of the requirements and is properly functional to be used by the company. However, if required, both of the applications can be further developed to provide additional needs or integrated with other existing systems.

9 Conclusion

The goal of the project was to create an Android mobile application which could interact with the backend server, have access to various features of the device such as camera, gallery and GPS and be localized. This goal has been successfully achieved after the project's completion. The user interface of the mobile application depends on the data extracted from the database, making the application more dynamic and versatile. In addition, as the application is localized, the resources and language used depend upon the locale configuration of the device which currently supports three locales which are English, Finnish and Swedish.

Along with the mobile application, a web application is also developed which acts as an interface to create and handle the data to be stored in the database, which is required for the mobile application. Through this application, questionnaires and questions can easily be created, grouped and modified. The answers fed by the users can also be viewed and can be downloaded as a file for the respective questionnaires.

The mobile application can be used as an efficient application for giving feedback for the pre-selected queries. However, the feedback system is not open. In addition, even though the application provides the interface to give answers as text and image and to choose from the multiple options, other ways of answering questions have not been implemented such as selecting multiple options and providing other types of feedback like video or audio. However, if required, these additional features could also be integrated in the future. Similarly the problem of cached GPS data can also be sorted out, if the project is to be carried out again.

In conclusion, the project has met the specified goals providing the functional web application and the Android application as a final product. The Public Work Department of Helsinki City can use the applications for their desired purpose of getting feedback from the general citizens in the workshops organized by them in the future.

References

1 Meier R. Professional Android 2 application development. Indianapolis, IN: Wiley Publishing, Inc.; 2010.

2 [X]cube Labs. The Android story [online]. Dallas, TX.
URL: http://www.xcubelabs.com/the-Android-story.php. Accessed 11 February 2012.

3 Murph D. Samsung's Galaxy Nexus gets official [online]. Engadget; 11 September 2011.
URL: http://www.engadget.com/2011/10/18/samsungs-galaxy-nexus-gets-official-Android-4-0-4-65-inch-hd/. Accessed 11 February 2012.

4 CameraPhonesPlaza. HTC G1 review [online]. 21 April 2011.
URL: http://www.cameraphonesplaza.com/htc-g1-review/. Accessed 11 February 2012.

5 Gargenta M. Learning Android. Sebastopol, CA: O'Reilly Media, Inc.; 2011.

6 Android platform [online]. 7 February 2012.
URL: http://developer.Android.com/guide/basics/what-is-Android.html.
Accessed 11 February 2012.

7 Bray T. Be careful with content providers [online]. 6 May 2010.
URL: http://Android-developers.blogspot.com/2010/05/be-careful-with-content-providers.html. Accessed 4 March 2012.

8 Nourie D. Getting started with an Integrated Development Environment (IDE) [online]. 24 March 2005.
URL: http://java.sun.com/developer/technicalArticles/tools/intro.html.
Accessed 12 February 2012.

9 TechnoPedia. Android SDK [online].
URL: http://www.techopedia.com/definition/4220/android-sdk.
Accessed 12 February 2012.

10 Localization and iternalization on Android, the easy way [online]. 19 June 2010.
URL: http://Androidforbeginners.blogspot.com/2010/06/localisation-internationalisation-on.html.
Accessed 24 March 2012.

11 Android platform [online]. 12 March 2012.
URL: http://developer.Android.com/guide/topics/resources/localization.html.
Accessed 25 March 2012.

12 Hashimi S, Komatineni S, MacLean D. Pro Android 2. New York, NY: Apress; 2010.

13 ICanLocalize. Android localization tutorial [online]. OnTheGoSystems, Inc.
URL: http://www.icanlocalize.com/site/tutorials/Android-application-localization-tutorial/. Accessed 24 March 2012.

14 Ableson Frank W, Sen R, King C, Ortiz Enrique C. Android in action. Shelter Island, NY: Manning Publications Co; 2012.

15 Pushing the interface [serial online]. Mechanical Engineering: 2010;132(11):26-27.
URL: http://web.ebscohost.com.ezproxy.metropolia.fi/ehost/detail?vid=5&hid=105&sid=149b96cd-a4a2-40cd-ae5c-fb51ccc558e6%40sessionmgr115&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ%3d%3d#db=afh&AN=55095649.
Accessed 8 March 2012.

16 Shneiderman, B. Touch screen now offers compelling uses. Human-Comput 1991;8(2):93-94.

17 Amit Dhir. The digital consumer technology handbook:
a comprehensive guide to devices, standards, future directions, and programmable logic solutions. Newnes: Burlington, MA; 2004.

18 Google Inc. Android user interface design tips [online].
URL: http://docs.google.com/fileview?id=0BxEWAcbuD.zg1NGNiZmVhNDgtMWIyNi00MTU4LTkwYjEtNGQxODkzOTMzNjM0&hl=en. Accessed 8 March 2012.

19 Mckinzie D. Designing for Android[online]. 30 June 2011.
URL: http://coding.smashingmagazine.com/2011/06/30/designing-for-Android/.
Accessed 8 March 2012.

20 IT&C Solutions. Android tutorial - procedural vs. declarative design of user interface [online].
URL: http://www.itcsolutions.eu/2011/08/27/Android-tutorial-4-procedural-vs-declarative-design-of-user-interfaces/. Accessed 9 March 2012.

21 Darcey L. Create flexible Android UIs with fragments [online]. 6 June 2011.
URL:http://www.developer.com/ws/Android/development-tools/create-flexible-Android-uis-with-fragments.html. Accessed 14 April 2012.

22 Jazaveri M. Some trends in Web application development [serial online]. Future of Software Engineering: 2007;199-213.
URL:http://ieeexplore.ieee.org.ezproxy.metropolia.fi/xpl/articleDetails.jsp?tp=&arnumber=4221621&contentType=Conference+Publications&searchField%3DSearch_All%26queryText%3Dsome+trends+in+web+application+development.
Accessed 11 April 2012.

23 GestureWorks. Gesture: One finger flick [online].
URL: http://gestureworks.com/support/supported-gestures/one-finger-flick/.
Accessed 21 March 2012.

Appendices

Appendix 1. Helper Classes for Android Application

Questionaire Class:

```
package com.hkiRakkenusQuestionaire;

public class Questionaire {

		private String name;
		private int numOfQuestion;


		public Questionaire(String name) {
			super();
			this.name = name;

		}

		public String getName() {
			return name;
		}
		public void setName(String name) {
			this.name = name;
		}
		public int getNumOfQuestion() {
			return numOfQuestion;
		}
		public void setNumOfQuestion(int numOfQuestion) {
			this.numOfQuestion = numOfQuestion;
		}


}
```

Question Class:

```
package com.hkiRakkenusQuestionaire;

import java.io.Serializable;


public class Question implements Serializable{

		private static final long serialVersionUID = 1L;
		private String question;
		private String answerType;
		private String[] options;
		private boolean hasLocation;
		private int questionaireId;

		public Question(String question, String answerType,boolean hasLocation, int
questionaireId) {
			super();
```

```
this.question = question;
                        this.answerType = answerType;
                        this.hasLocation = hasLocation;
                        this.questionaireId = questionaireId;
            }

            public int getQuestionaireId() {
                        return questionaireId;
            }

            public void setQuestionaireId(int questionaireId) {
                        this.questionaireId = questionaireId;
            }

            public String getQuestion() {
                        return question;
            }

            public void setQuestion(String question) {
                        this.question = question;
            }

            public String getAnswerType() {
                        return answerType;
            }

            public void setAnswerType(String answerType) {
                        this.answerType = answerType;
            }

            public String[] getOptions() {
                        return options;
            }

            public void setOptions(String[] options) {
                        this.options = options;
            }

            public boolean isHasLocation() {
                        return hasLocation;
            }

            public void setHasLocation(boolean hasLocation) {
                        this.hasLocation = hasLocation;
            }
}
```

Answer Class:

```
package com.hkiRakkenusQuestionaire;

public class Answer {
            private String text;
            private String image;
            private String location;
```

```
public Answer() {
                        this.text = null;
                        this.image = null;
                        this.location = null;
            }
            public String getText() {
                        return text;
            }
            public void setText(String text) {
                        this.text = text;
            }
            public String getImage() {
                        return image;
            }
            public void setImage(String image) {
                        this.image = image;
            }
            public String getLocation() {
                        return location;
            }
            public void setLocation(String location) {
                        this.location = location;
            }


}
```

HttpHandler Class:

```
package com.hkiRakkenusQuestionaire;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;

import Android.util.Log;

public class HttpHandler{
            private InputStream is;
            private String httpResult;
            private final String urlString =
"http://dev.mobile.metropolia.fi/naveenad/getQuestionaireAndroid.php";
            private String errorMsg;
```

```
public HttpHandler() {
		super();
	}

	public InputStream getIs() {
		return is;
	}

	public void setIs(InputStream is) {
		this.is = is;
	}

	public String getHttpResult() {
		return httpResult;
	}

	public void setHttpResult(String httpResult) {
		this.httpResult = httpResult;
	}

	public String getErrorMsg() {
		return errorMsg;
	}

	public void setErrorMsg(String errorMsg) {
		this.errorMsg = errorMsg;
	}

	//set inputStream
	public boolean executeHttpRequest(ArrayList<NameValuePair> nameValuePair) {
		try {
			HttpClient httpclient = new DefaultHttpClient();

			HttpPost httppost = new HttpPost(urlString);
			httppost.setEntity(new
UrlEncodedFormEntity(nameValuePair));
			HttpResponse response =
httpclient.execute(httppost);
			HttpEntity entity = response.getEntity();
			setIs(entity.getContent());
			return true;
		}
		catch(OutOfMemoryError m){
			Log.e("log_tag", "out of memory" + m.toString());
			return false;
		}
		catch (UnsupportedEncodingException u){
			//POST section
```

```
return false;
                        }

                        catch (NullPointerException e){
                                //POST & GET section
                                return false;
                        }
                        catch (ClientProtocolException e){
                                //POST section
                                return false;
                        }
                        catch (IOException e){
                                // POST & GET section
                                return false;
                        }
                        catch (Exception e) {
                                Log.e("log_tag", "Error in http connection" +
e.toString());
                                setErrorMsg("Error in http connection" +
e.toString());
                                return false;
                        }


                }
                //convert inputStream to String
                public boolean convertHttpResponse() {
                        StringBuilder sb = null;
                        try {
                                BufferedReader reader = new BufferedReader(new
InputStreamReader(
                                                                is, "iso-8859-1"), 8);
                                sb = new StringBuilder();
                                sb.append(reader.readLine() + "\n");
                                String line = "0";
                                while ((line = reader.readLine()) != null) {
                                        sb.append(line + "\n");
                                }
                                is.close();
                                setHttpResult(sb.toString());
                                return true;
                        } catch (Exception e) {
                                Log.e("log_tag", "Error converting httpResult " +
e.toString());
                                return false;
                        }
                }
}
```

Appendix 2. Custom Adapter Class to Create a List Item for Questionnaire Activity Class

```
package com.hkiRakkenusQuestionaire;

import java.util.ArrayList;
import Android.content.Context;
import Android.view.LayoutInflater;
import Android.view.View;
import Android.view.ViewGroup;
import Android.widget.ArrayAdapter;
import Android.widget.ImageView;
import Android.widget.TextView;

public class CustomAdapter extends ArrayAdapter<String> {
        public CustomAdapter(Context context, int textViewResourceId, ArrayList<String>
objects) {
            super(context, textViewResourceId, objects);
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            ViewHolder holder;
            if(convertView == null){
                LayoutInflater inflater = LayoutInflater.from(this.getContext());
                convertView = inflater.inflate(R.layout.list, parent, false);
                holder = new ViewHolder();

                holder.text = (TextView)convertView.findViewById(R.id.questionaireName);
                holder.image = (ImageView)convertView.findViewById(R.id.image);

                convertView.setTag(holder);
            }else{
                holder = (ViewHolder) convertView.getTag();
            }
            holder.text.setText(getItem(position));

            return convertView;
        }

    }

    final class ViewHolder{
        TextView text;
        ImageView image;
    }
```

Appendix 3. Custom MapActivity Class

```
package com.hkiRakkenusQuestionaire;

import java.io.IOException;
import java.util.List;
import java.util.Locale;
import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.DialogInterface;
import Android.content.Intent;
import Android.location.Address;
import Android.location.Geocoder;
import Android.os.Bundle;
import Android.view.MotionEvent;
import Android.view.GestureDetector.OnDoubleTapListener;
import Android.view.GestureDetector.OnGestureListener;
import com.google.Android.maps.GeoPoint;
import com.google.Android.maps.MapActivity;
import com.google.Android.maps.MapController;
import com.google.Android.maps.MapView;

public class LocationView extends MapActivity implements OnGestureListener,
OnDoubleTapListener{
                private MapView mapView;
   private MapController mc;
   private String address;

   GeoPoint p;
                /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      Bundle bundle = this.getIntent().getExtras();
      double lat = bundle.getDouble("latitude");
      double lng = bundle.getDouble("longitude");
      setContentView(R.layout.mapview);
      mapView = (MapView) findViewById(R.id.mapView);
      //mapView.setStreetView(true);
      mapView.setBuiltInZoomControls(true);
      mc = mapView.getController();

      p = new GeoPoint(
         (int) (lat * 1E6),
         (int) (lng * 1E6));

      mc.animateTo(p);
      mc.setZoom(17);
      mapView.invalidate();
      address = convertLatLongToAddress(lat,lng);
      confirmLocationDialogue(address);
   }
```

```
@Override
		protected boolean isRouteDisplayed() {
			return false;
		}


		@Override
		public boolean onDown(MotionEvent e) {
			// TODO Auto-generated method stub
			return false;
		}

		@Override
		public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX,
						float velocityY) {
			return false;
		}

		@Override
		public void onLongPress(MotionEvent e) {
		}

		@Override
		public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,
						float distanceY) {
			return false;
		}

		@Override
		public void onShowPress(MotionEvent e) {
		}

		@Override
		public boolean onSingleTapUp(MotionEvent e) {
			return false;
		}

		@Override
		public boolean onDoubleTap(MotionEvent e) {
			GeoPoint p = mapView.getProjection().fromPixels((int)e.getX(),
(int)e.getY());
			address = convertLatLongToAddress(p.getLatitudeE6()/
1E6,p.getLongitudeE6()/ 1E6);
			confirmLocationDialogue(address);
		    return true;
		}

		private String convertLatLongToAddress(Double lat, Double longitude) {
			String adress = "No Address Found";
			Geocoder gc = new Geocoder(this, Locale.getDefault());
			try {
				List<Address> addresses = gc.getFromLocation(lat,
longitude, 1);
				StringBuilder sb = new StringBuilder();
```

```
if (addresses.size() > 0) {
                                                        Address address = addresses.get(0);
                                                        for (int i = 0; i <
address.getMaxAddressLineIndex(); i++)

                sb.append(address.getAddressLine(i)).append("\n");
                                                        adress = sb.toString();
                                                }
                                } catch (IOException e) {
                                }
                                return adress;
                }

                @Override
                public boolean onDoubleTapEvent(MotionEvent arg0) {
                                return false;
                }

                @Override
                public boolean onSingleTapConfirmed(MotionEvent arg0) {
                                return false;
                }

                private void confirmLocationDialogue(final String address){
                                //Confirm the address
                                 AlertDialog.Builder builder = new AlertDialog.Builder(this);
                                   builder.setMessage(this.getString(R.string.confirmAddress) +"?
"+ address)
                                        .setCancelable(false)
                                        .setPositiveButton(this.getString(R.string.yesText), new
DialogInterface.OnClickListener() {
                                           public void onClick(DialogInterface dialog, int id) {
                                                 //int ordinateArray[] =
{p.getLatitudeE6(),p.getLongitudeE6()};
                                                        Intent resultIntent = new Intent();
ordinateArray);
                                                        resultIntent.putExtra("ordinates",
address);
                                                        setResult(Activity.RESULT_OK,
resultIntent);
                                                LocationView.this.finish();
                                           }
                                        })
                                        .setNegativeButton(this.getString(R.string.noText), new
DialogInterface.OnClickListener() {
                                           public void onClick(DialogInterface dialog, int id) {
                                                dialog.cancel();
                                           }
                                        });
                                   AlertDialog alert = builder.create();
                                   alert.show();
                }
}
```