MOBILE MEDICAL INFORMATION FOR THE DEAF

By

Siphokazi Dayile

A project document submitted in partial fulfillment of the requirements for the degree of

BSc (Honours)

University of the Western Cape

Date: November 15, 2013

University of the Western Cape

Abstract

Mobile Medical Information for the Deaf

by Siphokazi Dayile

Supervisor: Professor I Venter

Co-supervisor: Professor WD Tucker

Department of Computer Science

The language barrier, when trying to communicate with the Deaf person can be frustrating for both parties but may have serious consequences during an emergency, for example when a Deaf person must be warned about a possible dangerous situation. The Deaf community, whose language is Sign language, faces the same berries when accessing health information services as English second language user’s would. This paper focuses on a system that will address the communication gap between a Deaf person and a hearing person when visiting a pharmacist or doctor. A mobile application will be developed that could improve the health care services for Deaf patients. This application will be piloted to provide information on Human Immunodeficiency Virus, Acquired Immune Deficiency Syndrome and hypertension the information that will help Deaf people have more information on these illnesses. The information will be recorded and will then be translated into sign language using video footage. The multi-media information will be stored in a database. This system could help Deaf people to gain more knowledge about these illnesses.

Table of Contents

[Table of Contents i](#_Toc372116616)

[List of figures iii](#_Toc372116617)

[List of Tables iv](#_Toc372116618)

[Glossary v](#_Toc372116619)

[Chapter 1 1](#_Toc372116620)

[user requirements documents 1](#_Toc372116621)

[Introduction 1](#_Toc372116622)

[User’s view of the problem 2](#_Toc372116623)

[Brief description of the problem domain 3](#_Toc372116624)

[Expectations from the system 3](#_Toc372116625)

[Not expected from the system 3](#_Toc372116626)

[Conclusion 3](#_Toc372116627)

[Chapter 2 5](#_Toc372116628)

[requirements analysis document 5](#_Toc372116629)

[Introduction 5](#_Toc372116630)

[Designers interpretation of the problem 5](#_Toc372116631)

[Breakdown the solution 5](#_Toc372116632)

[Existing solution 8](#_Toc372116633)

[Testing methods 9](#_Toc372116634)

[Conclusion 9](#_Toc372116635)

[Chapter 3 10](#_Toc372116636)

[user interface specification 10](#_Toc372116637)

[Introduction 10](#_Toc372116638)

[Description of the complete user interface 10](#_Toc372116639)

[What the user interface looks like to the user 12](#_Toc372116640)

[How the user interface behaves 18](#_Toc372116641)

[How the user interacts with the system 19](#_Toc372116642)

[Conclusion 20](#_Toc372116643)

[Chapter 4 21](#_Toc372116644)

[object oriented ANALYSIS 21](#_Toc372116645)

[Introduction 21](#_Toc372116646)

[Data dictionary defining what each object represents 21](#_Toc372116647)

[Class Diagram 24](#_Toc372116648)

[Conclusion 25](#_Toc372116649)

[Chapter 5 26](#_Toc372116650)

[object orientated design 26](#_Toc372116651)

[Introduction 26](#_Toc372116652)

[State diagram 26](#_Toc372116653)

[Pseudo-code 27](#_Toc372116654)

[Conclusion 29](#_Toc372116655)

[Chapter6 30](#_Toc372116656)

[Implementation 30](#_Toc372116657)

[Introduction 30](#_Toc372116658)

[Software deployed 30](#_Toc372116659)

[Hardware deployed 30](#_Toc372116660)

[Progress 31](#_Toc372116661)

[Term2 31](#_Toc372116662)

[Term3 31](#_Toc372116663)

[Challenges 31](#_Toc372116664)

[Code documentation 32](#_Toc372116665)

[MainActivity.java 32](#_Toc372116666)

[ListViewActivity.java 39](#_Toc372116667)

[BinderData.java 50](#_Toc372116668)

[XParser.java 58](#_Toc372116669)

[Explanation.java 63](#_Toc372116670)

[SecondActivity.java 69](#_Toc372116671)

[Information.java 76](#_Toc372116672)

[Conclusion 82](#_Toc372116673)

[Chapter7 83](#_Toc372116674)

[Testing 83](#_Toc372116675)

[Introduction 83](#_Toc372116676)

[Testing procedure 83](#_Toc372116677)

[Usability testing 84](#_Toc372116678)

[Functionality testing 85](#_Toc372116679)

[Recommendations 86](#_Toc372116680)

[Conclusion 86](#_Toc372116681)

[Chapter8 88](#_Toc372116682)

[Introduction 88](#_Toc372116683)

[Installing the application 88](#_Toc372116684)

[Starting the application 88](#_Toc372116685)

[Conclusion 95](#_Toc372116686)

[Appendixes 97](#_Toc372116687)

[Appendix A 97](#_Toc372116688)

[Testing Questionnaire 97](#_Toc372116689)

[Appendix B 99](#_Toc372116690)

[Appendix C 100](#_Toc372116691)

[Appendix D 101](#_Toc372116692)

[Appendix E 102](#_Toc372116693)

[APPENDIX F 103](#_Toc372116694)

[Questionnaire 103](#_Toc372116695)

[bibliography 104](#_Toc372116696)

[Index 106](#_Toc372116697)

List of figures

*Number Page*

[Figure 1 List of related questions 6](#_Toc372116698)

[Figure 2 How the system will work 7](#_Toc372116699)

[Figure 3 Sign support design. (Blake, 2008) 8](#_Toc372116700)

[Figure 4 Android phone main screen 12](#_Toc372116701)

[Figure 5 Main menu screen showing the application icon 13](#_Toc372116702)

[Figure 6 User requested to press the tick to view the list of illnesses 14](#_Toc372116703)

[Figure 7 User will now select a list of illnesses 15](#_Toc372116704)

[Figure 8 Explains what happens to the next 16](#_Toc372116705)

[Figure 9 User plays or pause the video 17](#_Toc372116706)

[Figure 10 Play video 18](#_Toc372116707)

[Figure 11 Use case diagram 20](#_Toc372116708)

[Figure 12 Class Diagram 25](#_Toc372116709)

[Figure 13 State diagram 26](#_Toc372116710)

[Figure 14 Usability Feedback 85](#_Toc372116711)

[Figure 15 Functionality feedback 86](#_Toc372116712)

[Figure 16 Main menu screen 89](#_Toc372116713)

[Figure 17 Applications home screen 90](#_Toc372116714)

[Figure 18 List of illnesses 91](#_Toc372116715)

[Figure 19 Explains what happens to the next screen 92](#_Toc372116716)

[Figure 20 List of illness information 93](#_Toc372116717)

[Figure 21 Video Player 94](#_Toc372116718)

List of Tables

*Number Page*

[Table 1 Object Table 22](#_Toc372116349)

[Table 2 Search object 23](#_Toc372116350)

[Table 3 Recognition object 23](#_Toc372116351)

[Table 4 Retriever object 24](#_Toc372116352)

[Table 5 Project Plan term3 100](#_Toc372116353)

[Table 6 Project Plan 2 101](#_Toc372116354)

[Table 7 Project plan 102](#_Toc372116355)

Glossary

AIDS Acquired Immune Deficiency Syndrome

Apache. Is a Web Server that is distributed under an “open source” license

Class diagram. is a type of static structure diagram that describes the structure of a system by showing the systems classes, their relationships among classes

Deaf. With the capital D refers to people whose first language is sign language and who are members of a specific linguistic cultural group

GUI. Graphic User Interface allows people interaction with programs in more ways than typing; it uses graphical icons, and visual indicators, rather than text based interfaces.

HIV. Human Immunodeficiency Virus

Hypertension. Abnormally high blood pressure.

MySQL. Is an open source relational database management system that is based on the structure query language?

PHP. Is a hypertext processor that allows web developers to create dynamiccontent that interacts with the database.

SASL. South African Sign Language

**API.** Application programming interface, is a set of routines, protocols, and tools of building software application

**AVD.** Android virtual device

**SDK**. Software development kit is a set of software development tools that allows for creation of applications for a certain software package, software framework, hardware platform, computer system, operating system, or similar development platform.

**SD card**. Secure Digital Card, is an ultra-small flash memory card designed to provide high-capacity memory in a small size.

**APK**. Android application package file is a file format used to distribute and install application software.

Chapter 1

user requirements documents

# Introduction

This chapter describes the problem from the user’s point of view and briefly defines the problem domain.

Deaf with a capital ‘D’ refers to people whose first language is sign language and who are members of a specific linguistic cultural group (Buttussi, 2010). Studies have found that Deaf individuals lack access to illness information due to problems in communication, low literacy and tightly woven social networks within the Deaf community (April L, Winningham, & Wyatt, 2003).It was found in South Africa that Deaf pupils who receive their schooling through the medium of a spoken language and have been taught to lip read spend their time at school developing their lip reading and speech skills often at the cost of their general educational development (including the development of their reading and writing skills), which translates into low literacy levels (Glaser & Lorenzo) When a person lacks access to information it impacts on the person’s ability to learn thus many Deaf people are being kept from being successful in their communities because they do not have easy access to information (Kotze, 2012).

The proposed mobile medical information application should provide a means for the Deaf to access information about the specific illnesses. The information will be rendered easy accessible videos in South African Sign Language (SASL).

In South Africa most Deaf people use text communication such as Short Message Services (SMS), Email and Instant Messaging (IM) to communicate with the hearing community (Buttussi, 2010). Koos Looijesteijn designed a Deaf-to-hearing communication aid that could help a Deaf person to communicate with a hearing person using a mobile phone (Looijesteijn, 2009). The design was later on implemented by Muyowa Mutemwa in his MSc project. It was a system that assisted a Deaf person to convey their medical conditions to a doctor face to face. The prototype used pre-recorded sign language videos, to communicate with the Deaf person and English text for the hearing person (Mutemwa, 2011)

# User’s view of the problem

Deaf people have limited access to health information since much of the information that is available is only in text and in languages other than SASL. Collin-thome mentioned in his research article that it is known that Deaf people are less healthy than hearing people but that health information is not provided or easy accessible to them (Collin-thome, 2008). Deaf people only use mobile phones to send messages (SMS’s) or emails; they have limited access to mobile application (Apps) because Apps are not designed with them in mind.

The user requirements were gathered through document analysis by reading books and articles unfortunately it was not possible to interview a Deaf person however I spoke to several people from the BANG group. It is however clear that there is a need for mobile medical information for the Deaf to help them gain access to information on illnesses, thus they require a system that will help them view information on illnesses and further information should be provided in SASL. The system must be easy to use, reliable and should simplify communication between the Deaf patient and a doctor or pharmacist, if a Deaf person is diagnosed with an illness in that way they can use the application to understand an illness that they are diagnosed with.

# Brief description of the problem domain

Deaf people need a system that will ease the communication between them and a doctor or pharmacist. They sometimes find it hard to follow what the doctor is saying or understand the medical terms the doctor uses. Deaf people do not have mobile application designed especially with them in mind, an application that will help them view information on illness .The applications that provide illness information is available but the information is not provided in a language that a Deaf person understands, it is only in text. This system will help Deaf users: to view medical information using a mobile phone; understand what causes the illness; provide information on how to treat it; and will then help them gain more knowledge on specific illnesses. The information will be provided by using a recorded sign language video.

# Expectations from the system

The user should be able to view all the information regarding two illnesses (HIV/AIDS and hypertension) as a video. A user should be able to view all the information needed to educate them about the illness and how to go about finding the information that the system is providing to them.

# Not expected from the system

What is not expected from the system is to provide information regarding other illnesses. The system will only provide information about two specific medical conditions and will not be comprehensive. The system will work only on Android supporting devices it will not work on other phones. The system will not provide the information in text it will be in sign language.

# Conclusion

In this chapter the user’s requirements were looked at and in the next Chapter2 the requirements based on the designer’s point of view will be analyzed.

Chapter 2

requirements analysis document

# Introduction

This chapter takes the User Requirements Documents (URD) as a starting point and looks at the problem from a designer’s point of view. However instead of diving directly to implementation details, the analysis focuses on the system and software requirements needed to implement the user requirements

# Designers interpretation of the problem

The users in this case a Deaf person requires a system that will provide easy access to information on illnesses e.g. hypertension and HIV/AIDS, the system should be reliable. Deaf users have limited access to information and Deaf people prefer to communicate in sign language (Folkins & Sadler, 2005), thus the system should provide the information as a video recorded in sign language.

The system will allow a user to access the information using a mobile phone. The system will help the user gain access to illness information and it will therefore provide the Deaf person with information that is equivalent to what is available to the hearing community. The videos will only address HIV/AIDS and hypertension.

# Breakdown the solution

The user will interact with the system using his/her mobile phone in order for the user to use the software application will be installed on the user’s mobile phone. The information that the user will be accessing will be stored on the mobile devices SD card, to retrieve the information a user needs to have a smart phone that will allow the user to view a video footage.

This is how the application will work when a user is viewing the application: the SASL video will pop up instructing a user to select a button that will take a user to a list of illnesses or click on the exit button to exit the application. The user will then view a list of illnesses that are represented with an image icon that the user is familiar with. The user will then click on the illness that they are looking for, then a list of illness information related to the one that the user have selected will pop up then a the user will have an option to pick one of the video’s that he/she wants to view , the video will the play giving information in SASL. An example of possible questions that the user might ask while viewing a SASL video is shown in, see Figure 1. Figure 2 illustrates how the application will work

The SD card will be populated with the illness information that can be accessed using a mobile phone. An Android mobile phone will be needed to capture the recorded video and the SD card will be used to save the recorded videos.

Figure 1 List of related questions

Input

The user will then see a list of information based on the illness they have selected

The user will then pick information about that illness that they would want to learn about

Select a start button

Selected button will take the user to the next screen

User picks an illness that they are looking for from the list provided

List of illnesses will be viewed

output

The video will play

Figure 2 How the system will work

# Existing solution

Sign Support is a mobile communication aid that helps a Deaf patient to be able to interact with a hospital pharmacy or a Doctor .Sign support was designed by a master’s student at UWC. Sign Support assists the pharmacy in dispensing medication to a Deaf patient. Figure 3 illustrates the design of sign support system. The application helps a Deaf person to report a medical condition to the doctor. The application uses a pre-recorded sign language video and English text to enable a Deaf person to tell a doctor how he/she is feeling (Mutemwa, 2011). The system does not have the functionality that the Deaf patient can view information about the illness after his/her visit to the doctor.

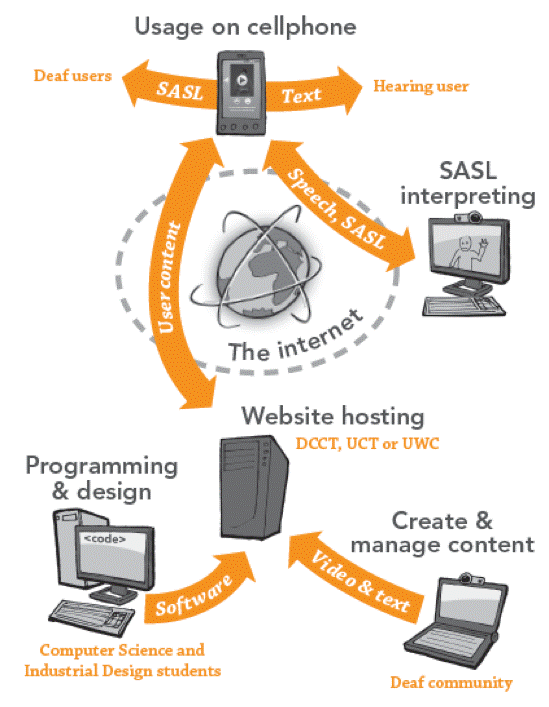


Figure 3 Sign support design. (Blake, 2008)

Existing solution Google and Wikipedia provides information to people who can read well and are not functionally illiterate. Google is a web search engine that helps a user find sites on web based on keyword searches. The user will type a key word then Google will provide a list of options written in text for the user to choose from the user will then choose one.

# Testing methods

Firstly the system will be tested by the UWC students to see if it is working properly. The students will then give feedback based on what they think of the system and suggest any changes that must be made on the system. The students will also check if there are any bugs so that they can be fixed before the system is sent to the Deaf people for testing. The system will then be tested by Deaf people because they are the ones who will be using the system. Deaf people will interact with the system by viewing the information in that way we will get the feedback of whether the system is doing what it is supposed to do and that the system produces the requirements that the user requested.

# Conclusion

In this chapter we looked at the requirements analysis based on the designer’s point of view. In the next chapter we will look at the user interface specification

Chapter 3

user interface specification

# Introduction

In the previous chapter the user requirements were analyzed based on the designers point of view and the possible methods of implementation were discussed .This chapter describes exactly what the user interface is going to do, what it looks like, and how the user interacts with the program. User interface specification includes screen shots of the interfaces that the user interacts with.

# Description of the complete user interface

The user interface (UI) is a graphical user interface (GUI) that is emulated by an Android emulator and consists of the following options:

* The start button – it the button that the user will click and send them to the list view that contains a list of illnesses.
* The Video view – is the video that will display when the user starts the application that will be a video that will give users instruction on how to use the application.
* A list of options with an illness icon and written text will be given from which the user will choose one.
* The play button – is the button that the user will use to play the video footage or pause it.
* The back button – will take the user back to the main menu where they can choose another illness or choose illness information.
* The home button – will take the user to the home screen where they can be able to view the instructions again in case they have forgotten about them.
* Exit button – this button will allow the user to exit the application.

# What the user interface looks like to the user

Figure 4 shows the start window of the Android Phone.



Figure 4 Android phone main screen

The main screen (see Figure 5) shows the icon named medical information that the user will press to start the application.

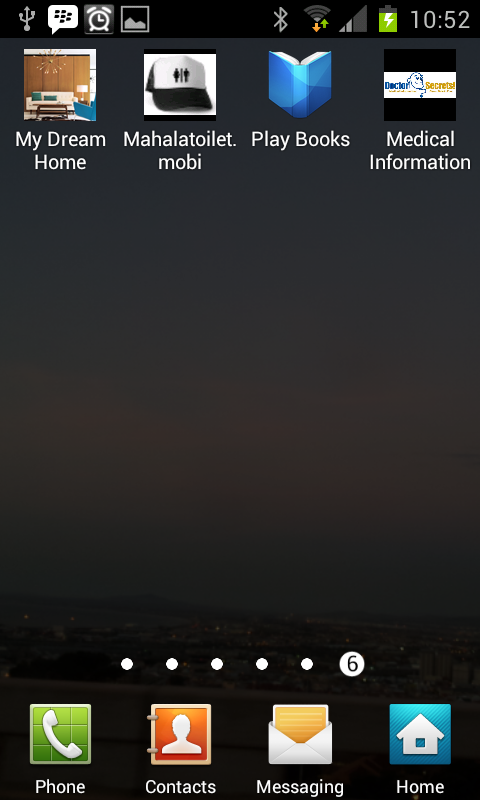


Figure 5 Main menu screen showing the application icon

The video will pop up after the user has clicked the application icon (see Figure6) the user is instructed to click the start button in order to go to the screen where they will see a list of illness or click the x to exit the application.



Figure 6 User requested to press the tick to view the list of illnesses

The user will now view a list of illness shown by an icon and written text next to the icon (see Figure 7) the example is also shown in the figure.

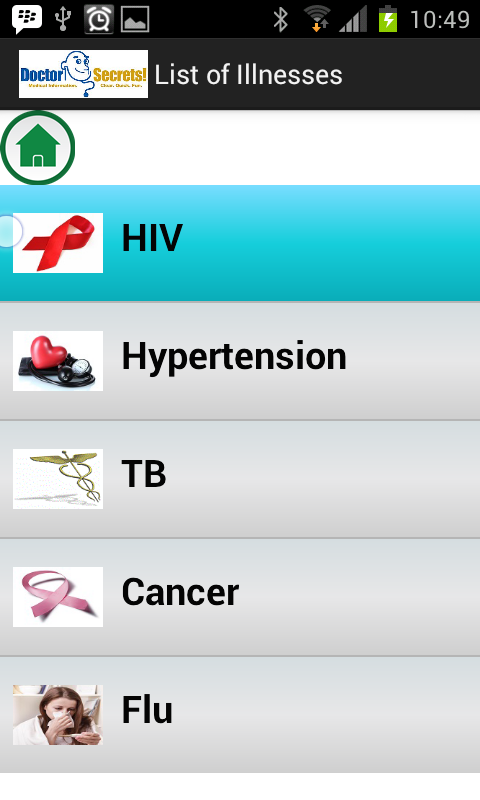


Figure 7 User will now select a list of illnesses

The video will then pop up instructing the user about what follows onto the next screen (see Figure 8), the user will have an option to click the forward button to view the questions that resulted from the illness that they have selected or click the back button to select another illness.



Figure 8 Explains what happens to the next

The user will now see the number of possible alternatives that resulted from the illness selected (see Figure 9). The user will click one of the options listed in Figure 9 to play the video of their choice.

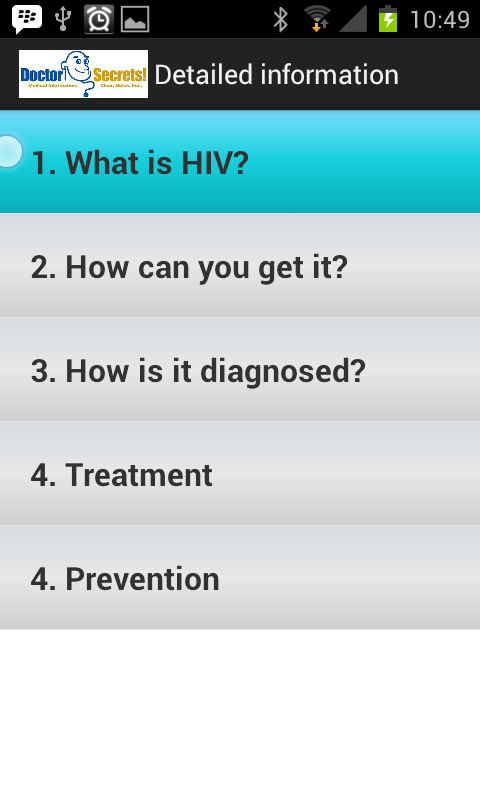


Figure 9 User plays or pause the video

The user will now play the video of their choice(see Figure 10); they can either play or stop the video and this option also allows the user to choose another video that they would want to view or can click the back button to return to the main menu or the x button to exit the application.

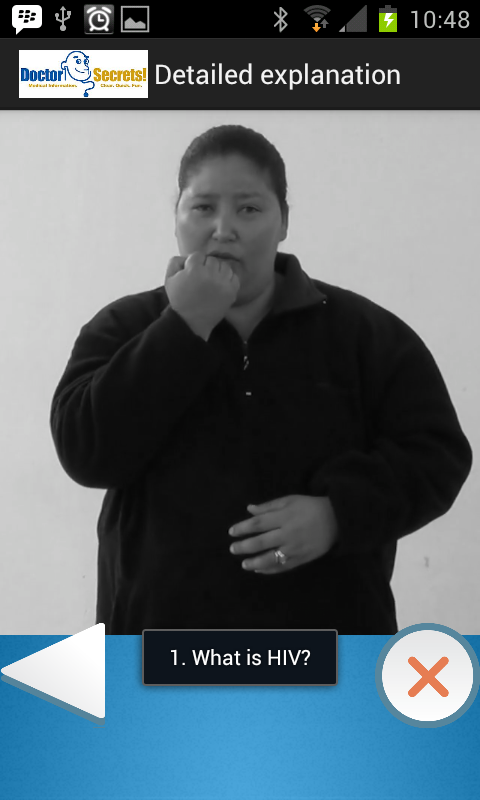


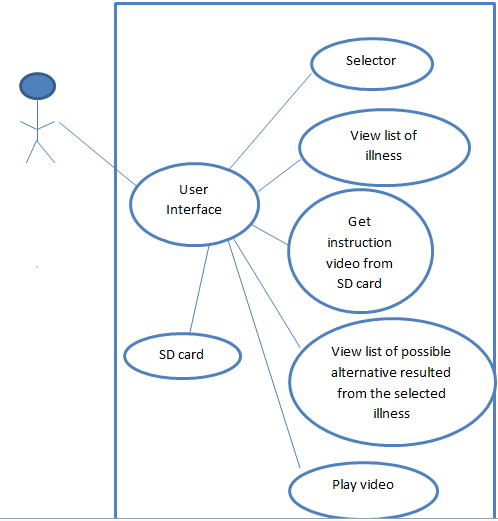
Figure 10 Play video

# How the user interface behaves

The system (application) is developed for an Android supported device. The interface allow the user to select an illness from the list of illnesses provided in order to get information about an illness, the information will be a video clip served from an Android device SD card. The interface will retrieve a list of video footage clips from the illness that the user has selected. The user can then select one of these clips and press play in order to watch it or press pause to stop the video from playing that way they can select another video to watch.

# How the user interacts with the system

The user will go to the generic home screen of an Android phone; the user will then go to the main menu screen showing the application icon, then an application icon will be visible, the user will then select the application icon that will take the user to the start screen where they instructed to press the tick button for them to view the list of illnesses. The user will then select an illness provided to them that is shown by an image icon and written text next to an image. The list of the information based on the illness that the user has selected will be retrieved. The recorded video footage will be stored in the SD card. The SD card will consist of the video footage with written text that is link to the illness information that the user have selected. The list of videos related to the illness that the user have selected will be sent to the user and then the user will select the video that they want to view. The user will then press the play button to view the video or can press the pause button; they can also press the back button that will bring them back to the main menu where they can select another illness. The use case diagram in figure 11 demonstrates how the user will interacts with the system



User

Figure 11 Use case diagram

# Conclusion

In this chapter we looked at how the user will interact with the interface and how the interface looked like to the user. In the next chapter we will look at the Object orientated design.

Chapter 4

object oriented ANALYSIS

# Introduction

The previous chapter described how the user interacts with the program and how the user interface is going to look like. In this chapter Object orientated analysis applies an object-orientated view to the problem. The relationship between the objects will be established while the class diagrams will display the attributes and methods of each class.

# Data dictionary defining what each object represents

The system has the following objects displayed see (Table1) namely search, recognition, retriever, display and video selector. The objects are explained and their attributes in Table 2, 3 and 4.Five objects namely recognition, retriever and display work in the background to help get the information from the database related to the keyword and help to sort the information according to the video that is closest to the keyword that the user inserted and then display it for the user to view and play the video.

Table 1 Object Table

|  |  |
| --- | --- |
| Objects | Description |
| Selector | This object allow the user to select a button that will take them to the next screen where they will view a list of illnesses, This object will also allow the user to choose one of the illnesses provided to them in order to view the list of information about the illness that they have selected. |
| Recognition | This object occurs in the SD card after the user has selected an illness, it will then point out to the video’s related to an illness that the user have selected. The pre-recorded videos will be stored in the SD card with written text about what the video is about. |
| Retriever | This object will get all the video’s related to the selected illness icon from the SD card before displaying the video’s to the user. The retriever will extract the video indexes from the SD card for example the name of the video. The videos information will then is retrieved from the SD card and be prepared for the display. |
| Display | This object will allow the user to see the list of video’s retrieved from the SD card; these videos will be displayed on the Android device screen. |
| Video Selector | This object will allow the user to select the video that they think matches best to the one that they were looking for. This object will help the user to press play to view the video or pause |

Table 2 Search object

|  |  |  |
| --- | --- | --- |
| Attributes | Description | Format |
| Select | This is the select that will help the user to get to the next screen where they could view the list of illnesses. This will also allow the user to select one of the illnesses displayed with an image icon. | String;  e.g. HIV |

Table 3 Recognition object

|  |  |  |
| --- | --- | --- |
| Attributes | Description | Format |
| File name | This attribute will cater for the name of the file that contains the videos | String e.g. Illness database |
| Video name | This attribute will cater for the name of the video that are stored in a file | String |

Table 4 Retriever object

|  |  |  |
| --- | --- | --- |
| Attribute | Description | Format |
| Find | This attribute will get all the video that are stored in the SD card and then retrieves it to the user. | String |

# Class Diagram

The individual objects of the system communicate to ensure that the system achieve its goal that is to provide illness information to the Deaf. The class diagram (Figure 12) illustrates the relationship between the objects and how they work together. This diagram demonstrates how each component is involved in the communication process. The user interface will consist of the button that the user will click to get to the next screen and display field, the user will select an illness icon from the one that will be provided to them, and the videos related to the selected illness will then be displayed to the user .The retriever can then get the video related to the selected icons from the SD card , The displayer will then display the video’s gathered by the retriever. Then the selector can play a video from the interface

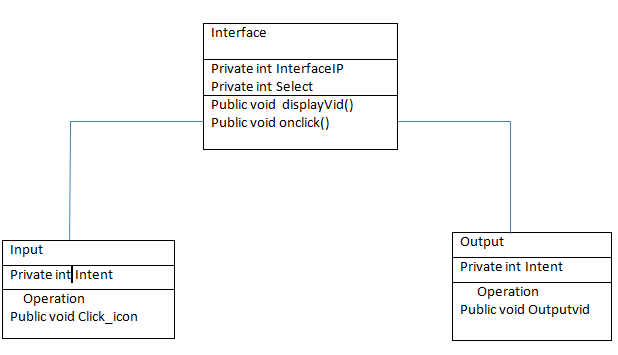


Figure 12 Class Diagram

# Conclusion

In this chapter, the system was analyzed from the object orientated perspective. The various system components were identified and the relationship between them was established. In the next chapter we will look at the object orientated design

Chapter 5

object orientated design

# Introduction

The previous chapter the object orientated applied an object orientated view to the problem. In this chapter we will take the classes in the object orientated analysis a level deeper into the realm of pseudo-code. The Object orientated design (OOD) defines the data types for the attributes. The OOD also defines the algorithm and implementation details of the class methods.

# State diagram

The diagram in (Figure13) illustrates the diagrams that display what happens when the user searches for the video.

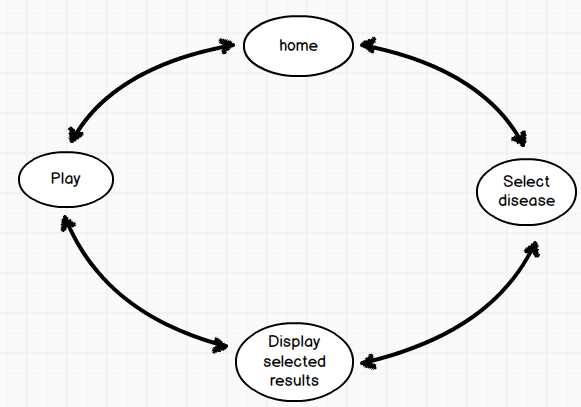


Figure 13 State diagram

# Pseudo-code

The pseudo-code below illustrates how the system will go about producing the video related to the selected illness. The pseudo-code also illustrates what will happen when the user selects an illness and how the information from the SD card will be retrieved and send to the mobile phone for the user to view the video that they want from the list of videos gathered from the SD card that are related to the selected illness.

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//the activity is being created this is where the video views will be initialised

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

protected void onStart(){

super.onStart();

//This method allows the activity to start

}

protected void onResume(){

super.onResume();

//This method makes the activity to be visible, this is where the

The video will be viewed as well as the video paths

}

protected void onStop(){

super.onStop();

// the activity is now going to stop, this method can also allow the

User to stop the video by pressing on the visible media controller

}

protected void onSelect(){

super.onSelect();

// Select an activity, this method allow the user to select an item that will be displayed to them from a file.

}

}

# Conclusion

In this chapter we looked at the pseudo-code that will help with the coding and we also looked at the state diagram that shows the events that will occur when the user insert the keyword. In the next chapter we will look at the code documentation.

Chapter6

Implementation

# Introduction

The previous chapter gave an overview of how the project implementation will be done, how the project looks like and how it will be made to function on an Android mobile phone. Different classes together with their pseudo codes were explained further to make the implementation easily understandable. The previous chapter holds the starting point for this chapter. In this chapter the code documentation will be shown and each class will be explained in more details.

# Software deployed

Android SDK- This is the software development kit that provides the API libraries and developer tools necessary to build, test and debug the application for the Android environment. Android SDK enables developers to build applications using the java programming language; it also includes an Android virtual device (AVD) that is used for testing the application. The Integrated Development Environment (IDE) enables developers to create code, test and deploy software quickly for the Android operating system.

# Hardware deployed

Android Device- The device is used to display the applications interface that the user will be able to interact with as explained in Chapter 4 in the User Interaction design.

# Progress

This project has not been stable from the time that it was started several changes had to be made, which I will describe in the following section. I will also explain what I have accomplished so far.

# Term2

In term one the users interface allowed the user to insert a query keyword that allowed the system to search for the video that provided the users illness information from a file stored in the database.

# Term3

We have discovered, that since our main focus is Deaf people who mostly are functionally illiterate, it was decided to review the use words or text, to use a familiar icon that will represent an illness. The image icon will be displayed for the user to select the one illness that they would want to learn about. The interface also allows the user to navigate between screens viewing information about the illnesses that they have selected. The interface gives the users an option to move forward, back or to select another illness. The videos will be stored in the file on the SD card.

# Challenges

Every programming project has its challenges and this project was not an exception. While developing the application we encountered several technical problems that affected the application’s functionality. A further problem was not being able to find the right path for accessing the pre-recorded videos from the SD card. Eclipse posed a problem with its version control one had to change the API’s to suit the project being developed. Creating the scrollable list view to hold all the illness icons proved to be such a challenge, however at the end we managed to create a scrollable list view with image icons that represents an illness. It was too difficult to combine the list view and the video view one screen it made the application crash each time, it was thus not included.

# Code documentation

The source code that is shown below illustrates the application’s functionality to retrieve videos. It allows the user to click on an image button to show a video but also allow the user to exit the application by using the exit button. The classes that are represented below have different functionality and they are explained in more details.

# MainActivity.java

/\* @author Siphokazi Dayile

\* Class discription:

This is an important class of the application because it is the one that is invoked when the application starts. This class displays the video player that will allow the user to view an instruction video it also gets the video path from the SD card and then displays it to the screen. This class calls the ListViewActivity.java class by using Image button; the user will click on the image button to go to the next class. This class consists of the video player, image buttons that allow the user to proceed with the application and also consists of the image button that will exit the application.

\* www.developer.android.com

\*/

package com.example.mynewproject;

import java.io.File;

import java.util.List;

import android.os.Bundle;

import android.os.Environment;

import android.os.Vibrator;

import android.app.Activity;

import android.content.Intent;

import android.view.Menu;

import android.view.View;

import android.widget.Button;

import android.widget.ImageButton;

import android.widget.ListView;

import android.widget.MediaController;

import android.widget.VideoView;

public class MainActivity extends Activity {

/\*This is where the essential properties of

the video and Image buttons are initialised \*/

private static final View VideoView = null;

VideoView v; /\*Declaring video view \*/

ImageButton start;/\*Declaring image button\*/

ImageButton End;/\*Declaring image button\*/

private Vibrator vib;/\*Declaring vibrator\*/

/\*This method initialises all the activities.It calls

the setContentView(int) with the layout resource that

defines the user interface and using the findViewById(int)

to retrieve the the user interface that the user will interact with \*/

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

start= (ImageButton) findViewById(R.id.imageButton2); //Retrieves the imageButton2 from the xml

End =(ImageButton) findViewById(R.id.imageButton1); // Retrieves the imageButton1 from the xml

vib =(Vibrator)this.getSystemService(VIBRATOR\_SERVICE);

Fileclip=new File(Environment.getExternalStorageDirectory().getPath()+

"/passcode.mp4"); //trying to get to the sdcard

if (clip.exists()) {

v=(VideoView)findViewById(R.id.VideoView);

v.setVideoPath(clip.getAbsolutePath());

//Use a media controller so that you can scroll the video contents

//and also to pause, start the video.

MediaController mediaController = new MediaController(this);

mediaController.setAnchorView(VideoView);

v.setMediaController(mediaController);

v.requestFocus();

v.start();

}

/\*This method sets the imagebutton to be clickable,

so that it can go to the next activity\*/

start.setOnClickListener(new View.OnClickListener(){

@Override

/\*This method is called when the button is clicked\*/

public void onClick(View v) {

// TODO Auto-generated method stub

vib.vibrate(100); // Makes the image button to vibrate

Intent i = new Intent(MainActivity.this,ListViewActivity.class); // Calls the next class

startActivity(i); // allow the class to start

}

});

/\*This method sets the button to be clickable and after the button has been clicked the user will then exit the application \*/

End.setOnClickListener(new View.OnClickListener() {

/\*This method is invoked when when view is clicked\*/

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

/\*On select the image button exit that will end the application \*/

vib.vibrate(100); // allow the button to vibrate when clicked

Intent startMain = new Intent(Intent.ACTION\_MAIN);

startMain.addCategory(Intent.CATEGORY\_HOME);

startMain.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(startMain);

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

}

# ListViewActivity.java

/\*@auther Siphokazi

class discription:

This class is called by the MainActivity.java class that is responsible for playing the video player that pops up when the application starts and displays the two image button. The image button named end is an exit button that will terminate the application when clicked and the other Image button named start calls the ListViewActivity.java when clicked. This class is responsible for listing the illnesses in a list view, the list that is displayed consist of an image and the text next to it .This class calls an xml file that has all the text and images listed using tags, the xml works the same way as the flat file. This class allows the user to click on one of the items that are displayed in the list and calls the SecondActivity.java class. This class maps all the tags that are saved on the xml file including the name of the illness, the text and the video path that will be sent to the SecondActivity.java class by using intent. In the xml the items are given an identity(id) that will uniquely identify each item that is saved in the file, this class will call the data that is saved in the xml file by using its unique id and sends it to the next class.

www.androidhive.info\*/

package com.example.mynewproject;

import java.io.File;

import java.io.IOException;

import java.io.InputStream;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import android.os.Bundle;

import android.app.Activity;

import android.content.Intent;

import android.util.Log;

import android.view.Menu;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ImageButton;

import android.widget.ListView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.SimpleAdapter;

public class ListViewActivity extends Activity {

// XML node keys

static final String KEY\_TAG = "data"; // parent node

static final String KEY\_ID = "id";

static final String KEY\_NAME = "name";

static final String KEY\_ICON = "icon";

static final String KEY\_vid = "vid";

ImageButton home;

// List items

ListView list;

BinderData adapter = null;

List<HashMap<String,String>> DataCollection; //hashmap for the list view

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

home= (ImageButton) findViewById(R.id.imageButton1);/\*Retrieves the image button from the main xml by its id\*/

/\*Here we parsing the xml data and then fill in the appropriate list that will be then be bound to a listview\*/

try {

DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();

Document doc = docBuilder.parse (getAssets().open("data.xml")); /\*gets the data that is saved in the xml\*/

/\*Creating a new hash map \*/

DataCollection = new ArrayList<HashMap<String,String>>();

// normalize text representation

doc.getDocumentElement ().normalize ();

NodeList List = doc.getElementsByTagName("data");

HashMap<String,String> map = null;

//This for loop it loops through all the data nodes in the data.xml

for (int i = 0; i < List.getLength(); i++) {

map = new HashMap<String,String>();

Node firstNode = List.item(i);

if(firstNode.getNodeType() == Node.ELEMENT\_NODE){

Element firstElement = (Element)firstNode;

// adding the id child node to the hashmap

NodeList idList = firstElement.getElementsByTagName(KEY\_ID);

Element firstIdElement = (Element)idList.item(0);

NodeList textIdList = firstIdElement.getChildNodes();

//--id

map.put(KEY\_ID, ((Node)textIdList.item(0)).getNodeValue().trim());

//2.adding the child name to the hashmap

NodeListcity List=firstElement.getElementsByTagName(KEY\_Name);

Element firstNAmeElement = (Element)nameList.item(0);

NodeList textNameList = firstNameElement.getChildNodes();

//--name

map.put(KEY\_NAME, ((Node)textNameList.item(0)).getNodeValue().trim());

//3.adding the child icon to the hashmap

NodeListiconList=firstElement.getElementsByTagName(KEY\_ICON);

Element firstIconElement = (Element)iconList.item(0);

NodeList textIconList = firstIconElement.getChildNodes();

//--icon

map.put(KEY\_ICON,((Node)textIconList.item(0)).getNodeValue().trim());

//4. adding the child vid the hash map

NodeListvidList=firstElement.getElementsByTagName(KEY\_VID);

Element firstVidElement = (Element)VidList.item(0);

NodeList textVidList = firstVidElement.getChildNodes();

//--vid

map.put(KEY\_Vid,((Node)textVidList.item(0)).getNodeValue().trim());

//Add to the Arraylist

DataCollection.add(map);

}

}

/\*getting the adapter by passing the xml data array list, It also binds the data that is read by a parser and updates the same to the xml elements which are the text view and the image view\*/

BinderData bindingData = new BinderData(this,DataCollection);

list = (ListView) findViewById(R.id.list);/\*aretrieves the list

from the main xml by its id\*/

Log.i("BEFORE", "<<------------- Before SetAdapter------------->>");

//Sets the adapted list data as a data source to the listview

list.setAdapter(bindingData);

Log.i("AFTER", "<<------------- After SetAdapter-------------->>");

home.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

//vib.vibrate(100);

Intent i = new Intent(ListViewActivity.this,MainActivity.class);

startActivity(i);

//setContentView(R.layout.diagnosishiv);

}

});

//Click event for single list row

list.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

/\*Invokes a new class when a particular item from the list

is selected by putting the required parameters in an intent\*/

Intent i = new Intent();

i.setClass(ListViewActivity.this,Explanation.class);

// parameters

i.putExtra("position", String.valueOf(position + 1));

//sending data to new activity

V i.putExtra("name",DataCollection.get(position).get(KEY\_NAME));

i.putExtra("icon", DataCollection.get(position).get(KEY\_ICON));

i.putExtra("vid", DataCollection.get(position).get(KEY\_VID));

// start the sample activity

startActivity(i);

}

});

}

catch (IOException ex) {

Log.e("Error", ex.getMessage());

}

catch (Exception ex) {

Log.e("Error", "Loading exception");

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.main, menu);

return true;

}

}

# BinderData.java

/\* @ author Siphokazi Dayile

Class discription:

This is one of the important classes in the application; it works with the ListViewActivity.java class by helping it to bind the image view and text view. It is responsible for binding the content of the data from each list view row item. This class set the text and image control that is found in the list\_row.xml, the xml is the user interface design and in this case it is a list view. This class is called in the ListViewActivity.java class to help bind the text View and Image View.

www.developwe.android.com

www.androidhive.com\*/

package com.example.mynewproject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import android.app.Activity;

import android.content.Context;

import android.graphics.drawable.Drawable;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

import android.widget.TextView;

public class BinderData extends BaseAdapter {

// XML node keys

static final String KEY\_TAG = "data"; // parent node

static final String KEY\_ID = "id";

static final String KEY\_Name = "name";

static final String KEY\_ICON = "icon";

LayoutInflater inflater;

ImageView thumb\_image;

List<HashMap<String,String>> DataCollection;

ViewHolder holder;

public BinderData() {

// TODO Auto-generated constructor stub

}

public BinderData(Activity act, List<HashMap<String,String>> map) {

this.DataCollection = map;

inflater = (LayoutInflater) act

.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

}

public int getCount() {

// TODO Auto-generated method stub

// return idlist.size();

return DataCollection.size();

}

public Object getItem(int arg0) {

// TODO Auto-generated method stub

return null;

}

public long getItemId(int position) {

// TODO Auto-generated method stub

return 0;

}

public View getView(int position, View convertView, ViewGroup parent) {

View vi=convertView;

if(convertView==null){

vi = inflater.inflate(R.layout.list\_row, null);

vi = inflater.inflate(R.layout.list\_item\_tracks, null);

holder = new ViewHolder();

holder.tvNAme = (TextView)vi.findViewById(R.id.tvName); // name

holder.tvImage =(ImageView)vi.findViewById(R.id.list\_image); // thumb image

vi.setTag(holder);

}

else{

holder = (ViewHolder)vi.getTag();

}

// Setting all values in listview

holder.tvCity.setText(DataCollection.get(position).get(KEY\_CITY));

//Setting an image

String uri = "drawable/"+ DataCollection.get(position).get(KEY\_ICON);

int imageResource = vi.getContext().getApplicationContext().getResources().getIdentifier(uri, null, vi.getContext().getApplicationContext().getPackageName());

Drawable image = vi.getContext().getResources().getDrawable(imageResource); // gets the image

holder.tvWeatherImage.setImageDrawable(image); // sets the image

return vi;

}

/\*This class serves as a place holder for the android UI element that is defined

in the xml file\*/

static class ViewHolder{

TextView tvName;

ImageView Image;

}

}

# XParser.java

package com.example.mynewproject;

import java.util.ArrayList;

import org.xml.sax.Attributes;

import org.xml.sax.SAXException;

import org.xml.sax.helpers.DefaultHandler;

/\*

\* Default Notification handler class for receiving ContentHandler

\* events raised by the SAX Parser

\*

\* \*/

public class XParser extends DefaultHandler {

ArrayList<String> idlist = new ArrayList<String>();

ArrayList<String> citylist = new ArrayList<String>();

ArrayList<String> iconlist = new ArrayList<String>();

//temp variable to store the data chunk read while parsing

private String tempStore = null;

public XParser() {

// TODO Auto-generated constructor stub

}

/\*

\* Clears the tempStore variable on every start of the element

\* notification \*/

public void startElement (String uri, String localName, String qName,

Attributes attributes) throws SAXException {

super.startElement(uri, localName, qName, attributes);

if (localName.equalsIgnoreCase("id")) {

tempStore = "";

} else if (localName.equalsIgnoreCase("city")) {

tempStore = "";

}

else if (localName.equalsIgnoreCase("icon")) {

tempStore = "";

}

else {

tempStore = "";

}

}

/\*

\* updates the value of the tempStore variable into

\* corresponding list on receiving end of the element

\* notification

\* \*/

public void endElement(String uri, String localName, String qName)

throws SAXException {

super.endElement(uri, localName, qName);

if (localName.equalsIgnoreCase("id")) {

idlist.add(tempStore);

}

else if (localName.equalsIgnoreCase("city")) {

citylist.add(tempStore);

}

else if (localName.equalsIgnoreCase("icon")) {

iconlist.add(tempStore);

}

tempStore = "";

}

/\*

\* adds the incoming data chunk of character data to the

\* temp data variable - tempStore

\*

\* \*/

public void characters(char[] ch, int start, int length)

throws SAXException {

super.characters(ch, start, length);

tempStore += new String(ch, start, length);

}

}

# Explanation.java

/\*@author Spokazi

This class is called by the ListViewActivity.java.It's responsibility is to get the data that is sent by the listViewActivity.java class and display it on screen.The class displays the video and then gets the video path that has been sent by the previous class. This class enables the user to see the video playing that will give explanation on what the user is going to do next, it also displays the image button that when clicked it will take the user to the next class which is the SecondActivity.java class or the previous class ListViewActivity.java class. When the buttons are clicked they vibrate so that the user will know that they have moved from one screen to the other.

\*/

package com.example.mynewproject;

import android.app.Activity;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import android.os.Environment;

import android.os.Vibrator;

import android.view.View;

import android.widget.ImageButton;

import android.widget.MediaController;

import android.widget.Toast;

import android.widget.VideoView;

/\*This class allows the user to view an explanation

video and also be able to move from another screen

by using the image button\*/

public class Explanation extends Activity{

private VideoView video;

ImageButton next; //declared image button next

ImageButton prev; // declares image button prev

private Vibrator myVib;

/\*All the activities are initialised in this method \*/

@Override

public void onCreate(Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.gethiv);

/\*gets the position to be displayed \*/

Intent i = getIntent();

int position = i.getExtras().getInt("position");

int vid = i.getExtras().getInt("vid");

next=(ImageButton)findViewById(R.id.next\_btn);/\* Retrieves the image button that allows the user to go to next class\*/

prev=(ImageButton)findViewById(R.id.back\_btn);/\*Retrieves the image button that allow the user to view the previous class\*/

myVib =(Vibrator)this.getSystemService(VIBRATOR\_SERVICE);

video = (VideoView) this.findViewById(R.id.videoView1);

Uri videoUri = Uri.parse(Environment.getExternalStorageDirectory()

.getPath()

+ "vid.mp4");

video.setVideoURI(videoUri);

video.setMediaController(new MediaController(this));

video.start();

/\*This methods sets the prev image button to be clickable\*/

prev.setOnClickListener(new View.OnClickListener(){

/\*When the button is clicked the ListViewActivity.java is invoked\*/

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent i = new Intent(GetHIVandAIDS.this,ListViewActivity.class);

startActivity(i);

myVib.vibrate(100);

}

});

/\*This method sets the image next button to be clickable \*/

next.setOnClickListener(new View.OnClickListener() {

/\*When the button is clicked the SecondActivity.java class is invoked\*/

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent intent = new Intent(GetHIVandAIDS.this,SecondActivity.class);

startActivity(intent );

myVib.vibrate(100);

}

});

}

}

# 

# SecondActivity.java

/\*@author Siphokazi

This class is invoked after the user has clicked on the next button in the Explanation.java class. It is responsible for displaying the list of illness information based on what the user have clicked from the ListViewActivity.java class.

\*/

package com.example.mynewproject;

import android.app.ListActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;

public class SecondActivity extends ListActivity {

// XML node keys

static final String KEY\_TAG = "data2"; // parent node

static final String KEY\_ID = "id";

static final String KEY\_NAME = "name";

static final String KEY\_DESCRIPTION = "discription";

static final String Key\_VID = "video"

/\*This is where all the activities are created \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

ListView lv = getListView(); // gets listview

lv.setTextFilterEnabled(true);

try {

DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();

DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();

Document doc = docBuilder.parse (getAssets().open("data2.xml")); // gets the data that is saved in the xml

/\*Creating a new hash map \*/

DataCollection = new ArrayList<HashMap<String,String>>();

// normalize text representation

doc.getDocumentElement ().normalize ();

NodeList List = doc.getElementsByTagName("data2");

HashMap<String,String> map = null;

//This for loop it loops through all the data nodes in the data.xml

for (int i = 0; i < List.getLength(); i++) {

map = new HashMap<String,String>();

Node firstNode = List.item(i);

if(firstNode.getNodeType() == Node.ELEMENT\_NODE){

Element firstElement = (Element)firstNode;

// adding the id child node to the hashmap

NodeList idList = firstElement.getElementsByTagName(KEY\_ID);

Element firstIdElement = (Element)idList.item(0);

NodeList textIdList = firstIdElement.getChildNodes();

//--id

map.put(KEY\_ID, ((Node)textIdList.item(0)).getNodeValue().trim());

//2.adding the child name to the hashmap

NodeList cityList = firstElement.getElementsByTagName(KEY\_Name);

Element firstNAmeElement = (Element)nameList.item(0);

NodeList textNameList = firstNameElement.getChildNodes();

//--name

map.put(KEY\_NAME, ((Node)textNameList.item(0)).getNodeValue().trim());

//3.adding the child discription to the hashmap

NodeListdesciptionList=firstElement.getElementsByTagName(KEY\_VID);

Element firstdesciptionElement = (Element)descriptionList.item(0);

NodeListtextDescriptionList=firstDescriptionElement.getChildNodes();

//--icon

map.put(KEY\_DESCRIPTION, ((Node)textDescriptionList.item(0)).getNodeValue().trim());

//4. adding the child vid the hash map

NodeListvidList=firstElement.getElementsByTagName(KEY\_VID);

Element firstVidElement = (Element)VidList.item(0);

NodeList textVidList = firstVidElement.getChildNodes();

//--vid

map.put(KEY\_Vid, ((Node)textVidList.item(0)).getNodeValue().trim());

//Add to the Arraylist

DataCollection.add(map);

}

}

lv.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

Intent i = new Intent();

i.setClass(ListViewActivity.this,Explanation.class);

// parameters

i.putExtra("position", String.valueOf(position + 1));

//sending data to new activity

i.putExtra("name",DataCollection.get(position).get(KEY\_NAME));

i.putExtra("vid", DataCollection.get(position).get(KEY\_VID));

// When clicked, show a toast with the TextView text

Toast.makeText(getApplicationContext(),((TextView) view).getText(),Toast.LENGTH\_SHORT).show();

startActivity(i);

}

});

}

/\*Updating the parsed data into the listview \*/

listAdapteradapter=new SimpleAdapter(SecondAcitity,this,list,R.layout.List\_item,new String[]{

"item\_id",TAG\_ID,"illness info" TAG\_Description}, new int[]{R.id.List\_id, R.id.discription});

//updating the listview

setListAdapter(adapter);

}

}

# Information.java

/\*@author Siphokazi

This class is invoked by the SecondActivity class it displays the video that will give explanation about the illnesses that the user have selected. It displays the video with the image buttons that will allow the user to go to the previous class Explanation class to select another question that will give more information on the illness , the user has another option of selecting the exit button to move terminate the application.\*/

package com.example.mynewproject;

import java.io.File;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.os.Environment;

import android.os.Vibrator;

import android.view.View;

import android.widget.ImageButton;

import android.widget.MediaController;

import android.widget.Toast;

import android.widget.VideoView;

/\*This class is responsible for displaying a video and image buttons \*/

public class Information extends Activity {

private static final View VideoView = null;

VideoView v;

ImageButton next;

ImageButton prev;

private Vibrator Vib;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.whatish);

/\*gets the position to be displayed \*/

Intent i = getIntent();

int position = i.getExtras().getInt("position");

int vid = i.getExtras().getInt("vid");

next = (ImageButton) findViewById(R.id.next); // Retrieves image button next

prev = (ImageButton) findViewById(R.id.prev); //Retrieves image button prev

Vib =(Vibrator)this.getSystemService(VIBRATOR\_SERVICE);

File clip=new File(Environment.getExternalStorageDirectory().getPath()+

"/vid.mp4"); //trying to get to the sdcard

if (clip.exists()) {

v=(VideoView)findViewById(R.id.VideoView);

v.setVideoPath(clip.getAbsolutePath());

//Use a media controller so that you can scroll the video contents

//and also to pause, start the video.

MediaController mediaController = new MediaController(this);

mediaController.setAnchorView(VideoView);

v.setMediaController(mediaController);

v.requestFocus();

v.start();

}

/\*Sets the image button onclick\*/

next.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent startMain = new Intent(Intent.ACTION\_MAIN);

startMain.addCategory(Intent.CATEGORY\_HOME);

startMain.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

startActivity(startMain);

Vib.vibrate(100);

}

});

/\*Sets the image button on click\*/

prev.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// TODO Auto-generated method stub

Intent k = new Intent(WhatIsHIV.this,SecondActivity.class);

startActivity(k);

Vib.vibrate(100);

}

});

}

}

# Conclusion

In this chapter an approach for implementing the project was discussed in detail and the source code was shown with inline comments. The project was implemented on an android mobile phone. The code documentation given shows the video view, image button and list view with images.

Chapter7

Testing

# Introduction

The previous Chapter focused on the applications implementation. It gave detailed documentation of the code used and explained how each part work to make the application functional, usable and how each component contributes to the project .In this Chapter, the system testing will be carried out to ensure that the application meets the user requirements that were discussed in Chapter 1 and specification .This Chapter will discuss the usability and functionality of the system and then evaluate and document the results. The main objective of this testing is to explore the user interface as to identify features that could be improved.

# Testing procedure

The test was conducted with 10 students from the University of the Western Cape, 5 were from the Computer Science department, 3 from EMS faculty, and 2 from the law faculty using one android device (Samsung S plus) with the application installed. In order to evaluate how the application work two testing methods were taken into consideration; Usability testing and functionality testing, both of these testing methods were conducted in a controlled setting, we looked for one of the available lecture room on campus and we found one in the computer science department , the testing took about an hour. The participants were given sweets for participating in the testing and that way it made them to be willing to help with the testing. The testing requires participants to use the application and later they were asked to complete a questionnaire to find out their thoughts about the application, the questions that were asked are shown in appendix A.

# Usability testing

10 participants were involved in the usability testing to ensure stable results. The testing session lasted approximately one hour .The participants were given a walkthrough the application to make sure that they understand how the application works and what was required for them to do. The purpose of the usability test was to assess the usability of the applications interface design. The participants were then asked to use the application after they were given a shown how to use the application, the participants were then asked to take the questionnaire to give feedback about their experience with the application, the questions that were asked are in appendix A. The participants were asked to rate how easy it was to use the system and 10% (1/10) thought that the application was difficult to use, 30% (3/10) of the participants thought that the application was moderate, 50% (5/10) of the participants felt that the application was easy to use and 10% (1/10) felt that the application was very easy to use (see figure14).

Figure 14 Usability Feedback

# Functionality testing

Functionality testing involves testing whether or not the user’s requirements were met or not. We performed functionality testing using persuasive technology to find out whether or not the user’s requirements were met. In Chapter 1 we mentioned that the users requires a system that will help them gain access to illness information and that it must be easy to use , we then informed the participants about the goal that we wanted to achieve and we then asked them to say whether or not the users requirements were met after they have used and understood what the system was supposed to do.80%(8/10) felt that the users requirements were met and the 20% (2/10)felt that the users requirements were not met because they felt like there were functionalities that were missing that they suggested that we must add that will be revealed in the recommendations. Some felt like it would be nice to have a video that and the list view interface on one screen, a video that will instruct the user to click on an illness detailed information that they want.

Figure 15 Functionality feedback

# Recommendations

The users gave positive feedback about the application, they felt that they could recommend the system to other people and that it was easy to use, however they suggested that we should add more functionality to the application like to give the users an option that would allow the user to find a doctor in a case of an emergency. The participants felt like it would be a good idea if we could have an option where the users can have a dictionary where they could ask or search for words that they do not understand.

# Conclusion

This Chapter discussed the methods that were used to test the application. The graphical representation of the results was given. The testing went very well because people were available and willing to help and the participants performed the testing strategies successfully there were no problem that was faced during the testing. During the usability test the participants tested the applications interface and most felt that the application is easy and it is not difficult to learn or use. They also mentioned that they could recommend the application to people.

Chapter8

User Guide

# Introduction

In the previous two Chapters the implementation and testing is discussed in more details, the project results were viewed in the two chapters. This Chapter gives instructions and guidelines on how to use the application. We will further discuss the guidelines or methods which the user will deploy in using the application to its fullest potential.

# Installing the application

The user will have to install the application and it will depend on where they would want to install it on their mobile devices, this application is suitable for the android devices therefore a user needs to have an android device to install this application. The user must install the APK file that is found by going to http://www.cs.uwc.ac.za/#downloads.

# Starting the application

The user will start the application by going to the Android device main screen where they will see the applications icon named medical information as shown in Figure 16. The user will select the icon to start the application so that they can be able to interact with the applications functionalities.

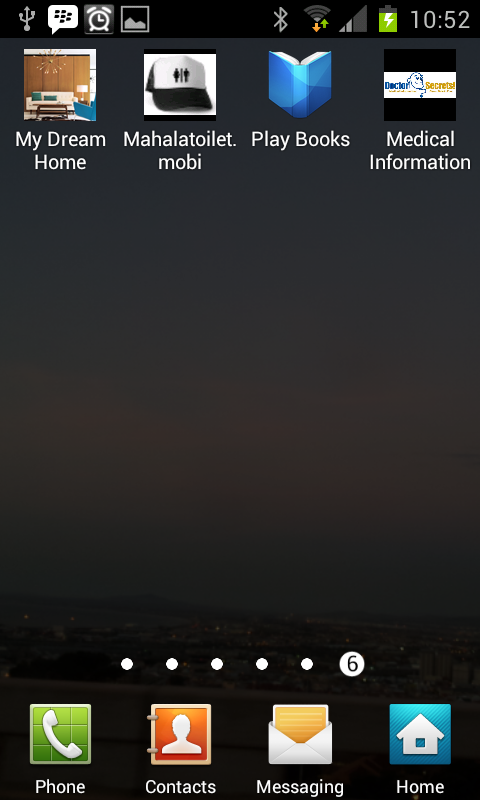


Figure 16 Main menu screen

Figure 16 shows the application home menu screen. The applications home screen consists of the video player and two buttons that the user will interact with. The video will pop up instructing the user on how to use the application, it will also instruct the user to select one of the buttons as shown in the figure below, the user has an option to select the button that has the correct sign to continue with the application and that will take the user to the list of illnesses or click the button represented with an x to exit the application as shown in the figure 17.



Figure 17 Applications home screen

If the user clicks on the button represented with a tick a list of illnesses will pop up represented by an illness icon that the users are familiar with, Figure 18 below show the list of illnesses with an image and written text that represents the name of an illness.

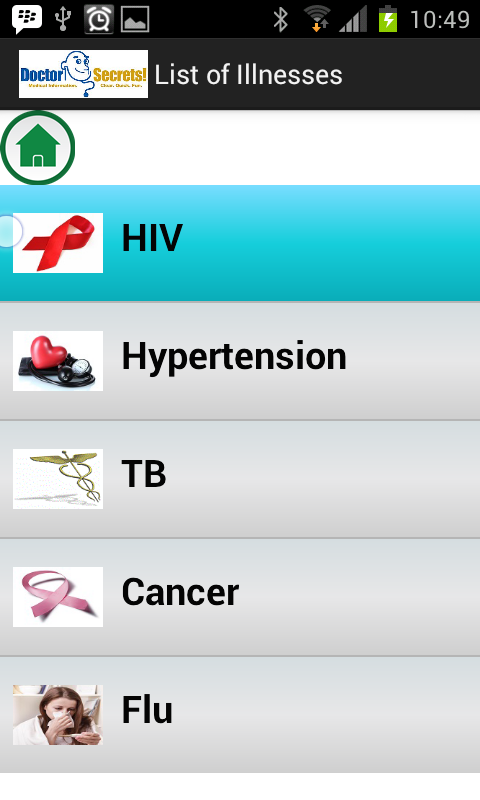


Figure 18 List of illnesses

The user will then select one of the illnesses as shown in figure 18 to view a video that will instruct them what follows onto the next screen. The user will then have to click on the forward button or go back to the previous screen where they can select another illness (see Figure 19).



Figure 19 Explains what happens to the next screen

If the user selects the next button, the user will then see the number of possible alternatives that resulted from the illness selected (see Figure 20). The user will be given possible questions about the illness to choose from, the user will then select one of the questions to view the illness information in sign language.

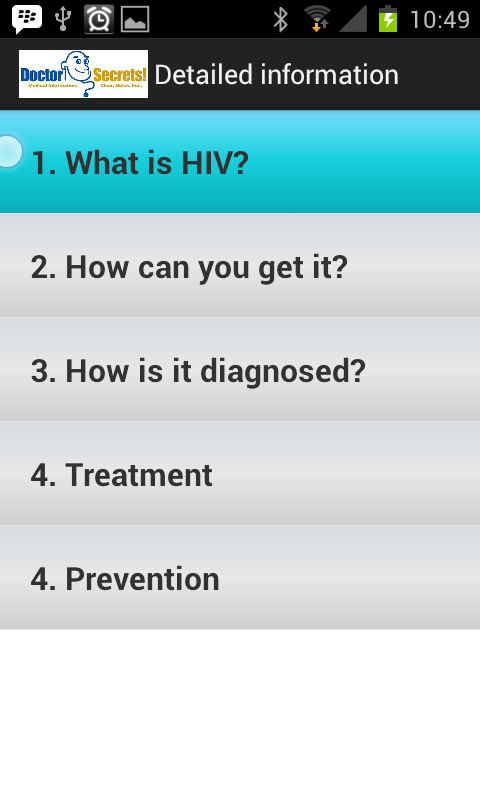


Figure 20 List of illness information

The user will then select one of the illnesses to view the video that will have the information about the illness that they have selected in sign language. The user will then have to click as shown in Figure 19, a video will then pop up as shown in Figure 20 giving them information about the illness information that they have selected. After watching the video the user will then have three option that is, to go back to the list of illness information by clicking on the back button to select another illness or click button represented with an x to exit the application.

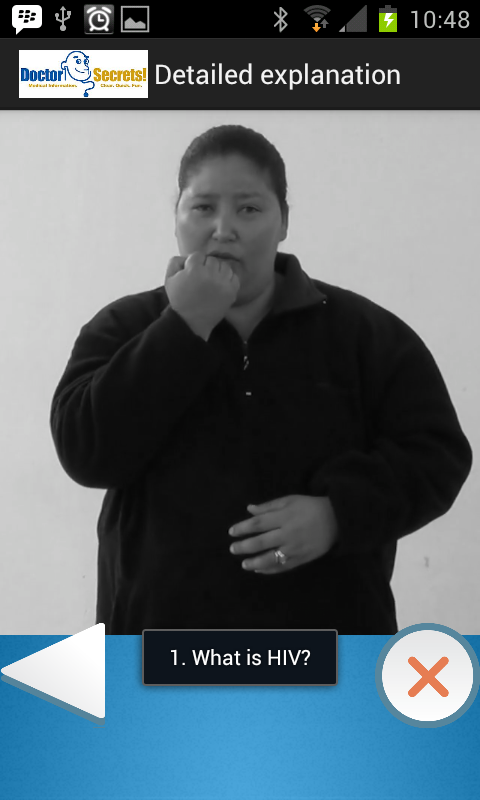


Figure 21 Video Player

This application has different buttons and each button has its own functionality.

The Exit button – this button will allow the user to exit the application in that way it will take them straight to the Androids home menu screen.

Home button – The home button that is represented by a small house will allow the user to go to the home screen of the application where they can view the instruction video.

The back button – This button that is represented with an arrow that shows that the user can go back; it allows the user to go to the previous activity.

The tick button – This button allow the user to continue with the application where they could view the list of illness.

# Conclusion

This chapter gives a user more information on how to interact with the system; it also gives them the functionality of the system.

Appendixes

# Appendix A

# Testing Questionnaire

1. Did you understand what the system is all about and what you were required to do with it?

|  |  |
| --- | --- |
| Yes | NO |

If not what is it that was not clear?

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. How easy it was using the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Very Easy | Easy | Moderate | Difficult | Very difficult |

1. What feature of the application was most appealing to you and why?

--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------------

1. What aspect of the application was not appealing to you?

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------------------

1. Would you recommend the application to other people?

|  |  |
| --- | --- |
| Yes | NO |

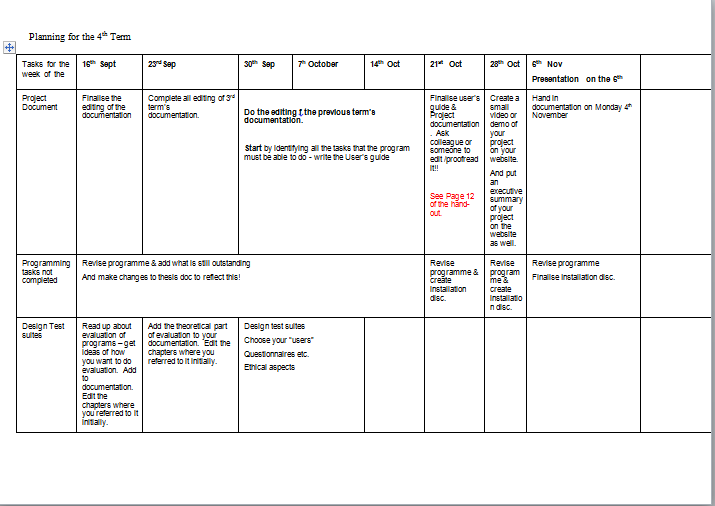
If not why?

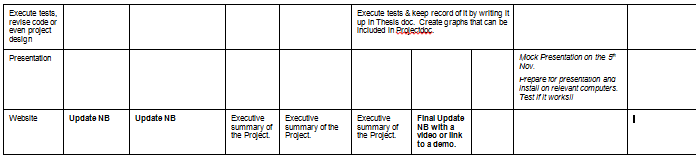
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

1. Are there any final comments or suggestions you would like to make about the application?

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

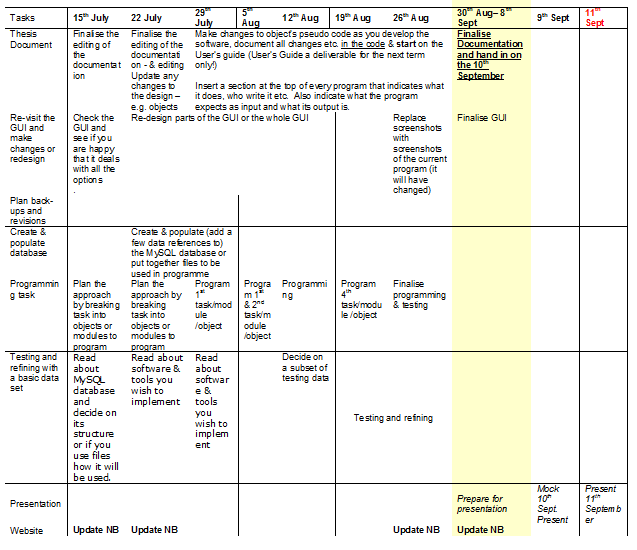
# Appendix B





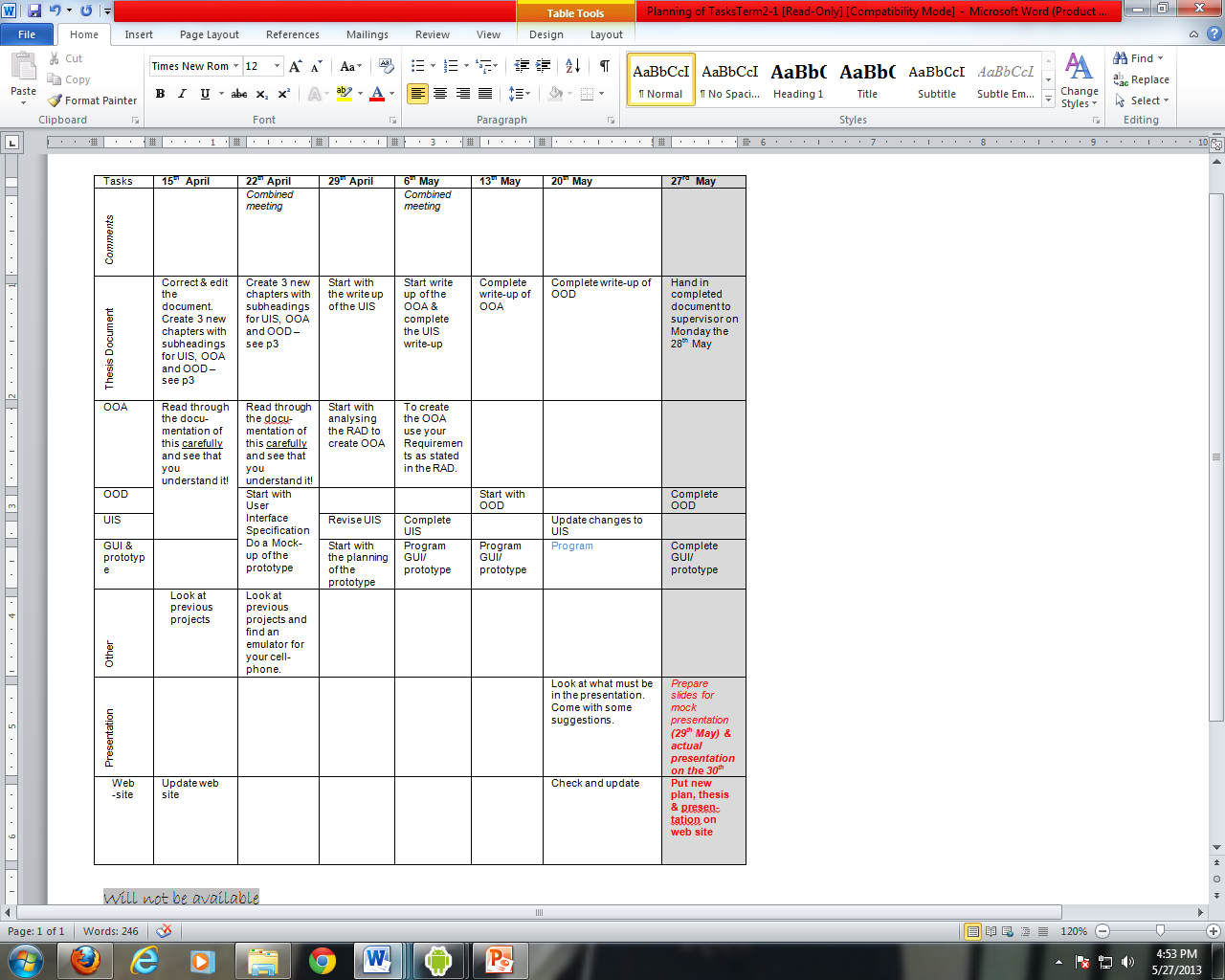
# Appendix C

Table 5 Project Plan term3



# Appendix D

Table 6 Project Plan 2



# Appendix E

Table 7 Project plan

|  |  |  |
| --- | --- | --- |
| **Term** | **Goal** | **Finishing** |
| 1 | Requirements and analysis   * Research possible project * Literature Review * Identify users to interview * Literature Review * User requirements gathering * User requirements document write up * Literature review * User requirements documentation * Start analyzing user requirements * Literature review * Requirements analysis document * Create website * Literature review * Complete document | 2nd April |
| 2 | Design and Development | End of term 2 |
| 3 | Implementation | End of term 3 |
| 4 | Test and Debug | End of term 4 |

# APPENDIX F

# Questionnaire

How old are you?

What is the highest grade you passed?

How often do you consult a Doctor?

What problems do you encounter when consulting a Doctor?

With whom do you discuss your state of your illness with?

If you were given an opportunity to view information about the state of your illness what information would you view

From where do you access internet?

What kind of a mobile phone do you use?

Does it have all the features that suit your needs?

Do you prefer using video or text to communicate with people?

bibliography

Akritidis, L., Katsaros, D., & Bozanis, P. (2008). effective ranking fusion methods for personalized MetaSrarch.

April L, Winningham, & Wyatt. (2003). Deaf and Hard of Hearing: HIV Prevention Needs.

Aslam, A., & Montague, M. (2001). Models for Metasearch.

Blake. (2008). *id-book*. Retrieved from id-book website: http://www.id-book.com/casestudy\_11-1\_2\_3.php

Buttussi, F. (2010). Using Mobile Devices to Support Communication between. 7-16.

Carmen Rodríguez (TID), M., (EUD), J. C., & Kampen, A. V. (2011). Report about the study of the new communication bridge between signers and hearing Community. *Sign Speak*, 1-52.

Collin-thome, D. (2008). *Why do you keep missing me .* Britain: Sign Health.

Folkins, A., & Sadler, G. (2005). Improvieng The Deaf Community 's access to prostate and testicular cancer. *BMC Public Health*.

Glaser, M., & Lorenzo, T. (n.d.). Developing Literacy with Deaf Adults. *Disa bility and Social change :South Africa Agenda*, 193.

Gulli, A., & Signorini, A. (2005). Building an Open Source Meta.

JADIDOLESLEMY, H. (July 2012). Search Result Merging and Ranking Strategies in Meta-Search. *IJCSI International Journal of Computer Science*.

Joachims, T. (n.d.). Optimizing Search Engines using Clickthrough Data. *Ithaca, NY 14853 USA*.

Kotze, N. (2012, 09 18). *Look Local*. Retrieved from Look local website: http://www.looklocal.co.za/looklocal/content/en/springs/springs-news-municipal?oid=6353649&sn=Detail&pid=490328&Access-to-information-for-deaf-people

Lezzoni, L., O'Day, B., killeen, M., & Harker, h. (2004). Communicating about health care:Observations from persons who are deaf or hard of hearing. *Ann Intern Med*.

Looijesteijn, K. (2009). The Design of a Deaf-to-Hearing communicationb Aid for South Africans. *Graduation project*, 11-25.

Macnair, D. T., & Hicks, D. R. (2013). *health*. (BCC) Retrieved from Deaf and hearing problem: http://www.bbc.co.uk/health/physical\_health/conditions/deafness1.shtml#aids\_to\_communication

Mutemwa, M. (2011). a mobile deaf to hearing communication aid for medical diagnoses.

Pay-Pal Integration Center. (n.d.).

Power Pos. (2011). Retrieved February 22, 2012, from Biometric Software Features: http://www.powerpos.co.za/PowerTill/Biometrics.aspx

S. M. Mahabhashyam, P. S. (2004). Tadpole: A MetaSearch Engine Evaluation of metasearch ranking strategies.

services, C. D. (n.d.). Directory of resourcesfor the Deaf. 145.

South African Government. (1997, 07 24). *Education White Paper 3: A Programme for the Transformation of Higher Education.* Retrieved 06 14, 2011, from Council on Higher Education: http://www.che.ac.za/documents/d000005/White\_Paper3.pdf

W. MENG. (2008.). Metasearch Engines.

Yiyao, L., Weiyi, M., Liangcai, S., Clement, Y., & King-Lup, L. (2003). Evaluation of Result Merging Strategies.

Index

A

Apache · 6

application) · 19

C

Class Diagram · i, iii, 25, 26

D

Deaf · i, v, 1, 2, 3, 5, 8, 9, 25, 32, 105, 106

H

HIV/AIDS · 3, 5

hypertension. · 5

M

MySQL · v

P

PHP · v

Pseudo-code · 28

S

SD card · vi, 6, 19, 20, 28, 32, 33

Software · 5

State diagram · i, iii, 27

U

URD · 5

Use case · iii, 21

Use case diagram · 21

User interface · i, 10, 11, 13, 19, 22, 25, 35, 51, 84

V

Video footage · 6