University of the Western Cape

Computer Science Honours Project

Video notification for SignSupport

Author:
Vuyisa Phindiso

Supervisor:
Prof. Bill Tucker

August 28, 2015

Abstract

SignSupport is a mobile application created to assist the communication between a Deaf and a hearing person; SignSupport application is in a pattern form and it is for predictable communication in various scenarios. The current version of SignSupport focuses on a communication between a pharmacist and a Deaf patient. The pharmacist types in and makes selections on the available buttons. When the pharmacist is done with medical dispending, the Deaf user also enters the settings information, which are the times the user takes daily meals. After all that information has been entered, the video alarm then automatically reads all that information and sets the alarm. The system then matches the selected options with the pre-recorded videos and populates or displays them to the Deaf patient who uses the South African Sign Language (SASL). SignSupport works offline because it is stored in the memory SD card of the mobile phone. With this application, Deaf patients can later on review the medication instructions. Because Deaf patients also forget to take their medication as instructed by the pharmacist like hearing people do. Therefore they need a medication reminder to remind them when it is time to take their medication by following the instructions given by the pharmacist. This paper proposes a medication reminder in the form of which video notification will be added on SignSupport mobile application. The video notification gives a reminder to Deaf patients. A patient can feel the intense vibration when there is a reminder. The video notification shows an image of the medication to be taken on the mobile phone screen; the patient can then click on the image to view its instructions regarding when and how to take the medication. These instructions will be presented by a video in SASL.

Contents

1 Glossary

Memory SD card $\rightarrow$ A small storage medium used to store data such as text, pictures and audio

SASL $\rightarrow$ South African Sign Language

XML $\rightarrow$ Extensible Markup Language

Android $\rightarrow$ It is a mobile operating system

2 Introduction

The video notification will be added on the existing SignSupport mobile application. The video alarm will use SASL videos to collect the necessary information from the Deaf users. The reason for using the SASL videos is that there are very few pharmacists that know how to sign [1], which makes it difficult for the Deaf users to grasp the information given by the pharmacist; this creates a communication barrier (see Figure 1). The pharmacist will input all the required information on how and when to take the medication, then the Deaf user will be given the phone to input his/her eating times of the day(breakfast, lunch and supper). All that information will be then used to automatically set the alarm.

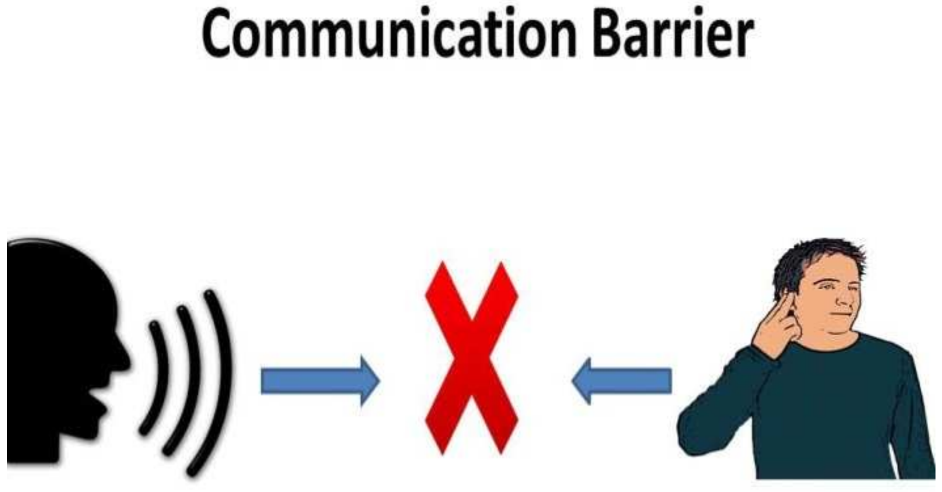


Figure 1: Communication barrier

2.1 Problem

Deaf patients also forget to take their medication as instructed by the pharmacist like hearing people do. Therefore they need a medication reminder to remind them when it is time to take their medication by following the instructions given by the pharmacist. But now the problem is that the existing reminders are either audio, English text or they use the combination of audio and text.

2.2 Possible Solution

The majority of people in South Africa own mobile phones and there are Who many people own smart phones that allow video play back, thus a video notification reminder is the solution to the problem currently faced by Deaf people. The aim of this project is to help Deaf people in a medication situation; in particular it will aid the Deaf patient to understand how and when to take the medication. As a solution to the problem, the video notification will be added to the SignSupport mobile application. This video notification will aid as a reminder to the Deaf patient on when and how to take the given medication. A pharmacist will enter medication instructions on the phone, when it is time for the Deaf person to take the medication the phone will vibrate with no sound. Then an image of the medication will be displayed on the screen, when the Deaf patient click or taps on that image, a SASL video will be played which will explain how to take the medication. The user will also be able to replay the pause, playback and fast forward the video.

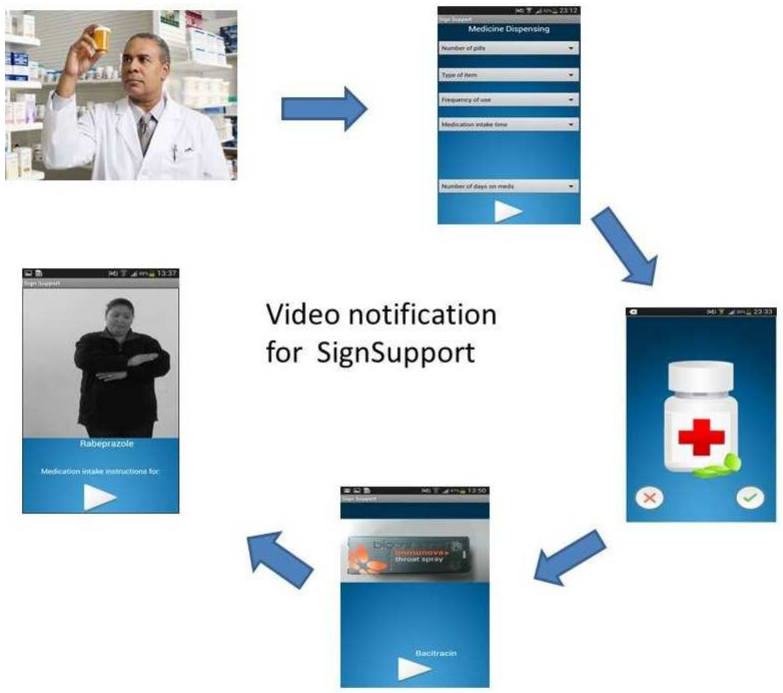


Figure 2: Solution summary

3 SignSupport Overview

SignSupport is an existing mobile application that assists a communication between a Deaf person and a pharmacist when a Deaf person is visiting a public hospital pharmacy (see Figure 3). SignSupport was designed by a Design Engineering student from Delft University of Technology (Prangnat Chininthorn)[6] and then implemented by one of the University of the Western Cape's former Masters Computer Science student Mr Michael Mothlabi. Video notification is an add-on feature in the SignSupport mobile application.

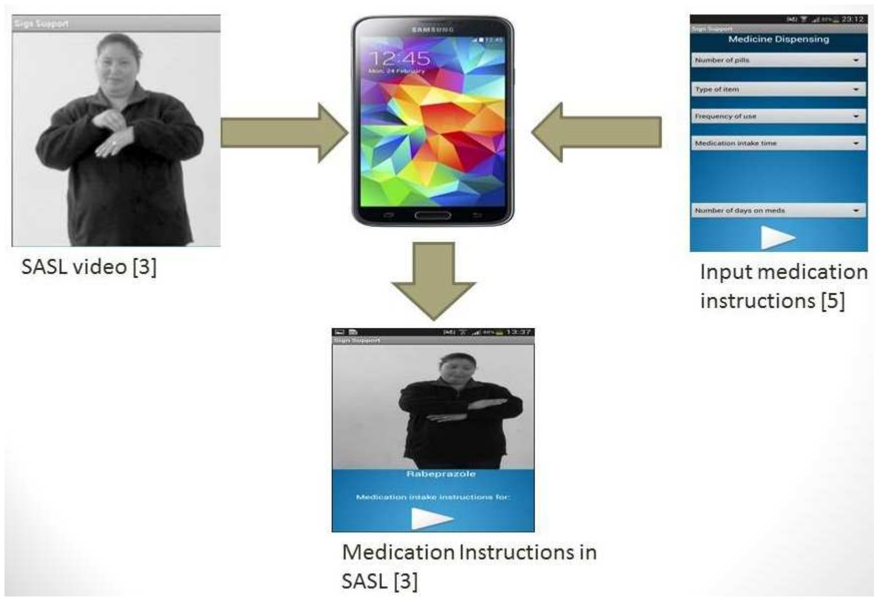


Figure 3: SignSupport Overview

4 User Requirements

4.1 Specifications

- Video notification must work on Android smart phones
- Suggested size of the phone is medium to large
- Phone must be able to view full screen
- Phone must have intense vibration
- The video instructions must be univocal
- Video notification must be network independent

4.2 Expectations of the Software Solution

Since there are two different end users of the proposed application, the following words indicate who is going to use that certain part of the application and the word in brackets indicates where expectation refers to.

Deaf → The Deaf patient
Pharm → The pharmacist
Both → Both, the pharmacist and the Deaf user

- The software should be able to interrupt whatever the user is busy with (Deaf).
- The software should have a nice and clear instructions for input (Pharm)
- User should be able to insert new instructions without disturbing the existing instructions (Pharm).
- User should be able to cancel or delete the instructions in case of mistakes made (Pharm).
- User should be able to take a picture of the medication using a camera (Pharm).
- Medication pictures should be able stored in the memory SD card (Both).

- User's phone must vibrate when it is time to take medication (Deaf)
- Picture of the medication must be displayed on the screen when the phone vibrates (Deaf).
- User should be able to display a SASL video or show instructions in SASL by clicking on the picture (Deaf).
- User should be able to playback, pause and fast forward the video instructions (Deaf).
- The software should be able to handle multiple notifications

5 Requirements Analysis

5.1 High level design of the solution

The Deaf patients will interact with the video notification system by using a mobile application that will be pre-installed as an add-on within the SignSupport mobile application. The SASL video will have an image of the medication and two buttons, one for watching the medication instructions and the other for dismissing the alarm. As soon as the user clicks on the yes button a video that is stored in the memory SD card and corresponds with that image will be retrieved and played. The Deaf patient will then watch the instructions given in the SASL video. The alarm will be automatically set using the information from the pharmacist and the Deaf user.

5.2 Deep Analysis of the solution

The video notification will be implemented on an Android mobile phone as an add-on feature for the SignSupport mobile application. The pharmacist types in and makes selections on the available buttons. After entering all the required information, the pharmacist will hand the phone back to the Deaf patient. The patient will then enter his/her daily eating times (breakfast, lunch, and supper). All that information will be written to a text file, then the video alarm will be automatically set using those text file. When it is time for the Deaf user to take the medication, the phone will vibrate and the back light will flash. The phone will then display the picture of the medicine with two buttons, one to continue and take the medication, and another one to dismiss the alarm. The phone has to have a big screen resolution with big clear icons, the capability to playback pause and fast forward videos with no internet access required. The phone has to have a 2 megapixels camera to take clear pictures of the medication. The user needs have a memory SD card of at least 512 Megabytes which will be used to store the different video with their images. When it is time to take the medication an image of the medication will appear on the screen of the phone, if the user clicks on the image, a SASL video will play on the phone and the user will have the options of pressing the pause button, playback button or fast forward button. The SASL video with instructions will be pre-installed in the memory SD card.

After playing the first video, the user can continue watching other SASL videos using the SignSupport user friendly buttons.

5.3 Related work

There are many other notification solutions that can act as a reminder; however the problem with these solutions is that most mobile phone manufactures produce audio and text based notifications and these solutions cater for literate people with basic mobile phone experience. The main aim of this video alarm is to focus on SASL videos; therefore this video alarm is based on vibration and video notification. Deaf and functionally illiterate people are not catered at all with these existing solution systems. MediSafe is an attractive, visual well designed app. Its interface is attractive and has an easy-to-use pill reminder [3]. MediSafe (see Figure 4) is a medication management application that is very close to SignSupport in a sense that it also reminds the patient to take the medication by showing the image of the medicine on the screen. MediSafe also allows the user to set the alarm, which is not the case on the SignSupport video alarm; the video alarm is automatically set. The problem with MediSafe is that it is built for hearing and literate people [3]. Furthermore, MediSafe allows the user help other family members with their medication. Another problem with the existing mobile notifications is the fact that the notification must be set by the users themselves, that means the users have to remember all the instructions given by the pharmacist and also remember to set the notifications which is not possible for the Deaf users, and this video alarm focuses on Deaf users. automatically set using the information from the pharmacist and the Deaf user.

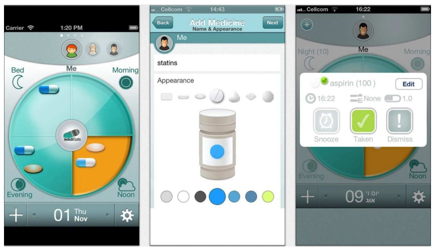


Figure 4: MediSafe

6 User Interface Specification

This section describes the User Interface in detail, it describes the design of the interface and how the user will interact with the software. Since the User Interface Specification focuses on the user interface itself, the description of the UIS will include the screen shots of the software that the user will interact with.

6.1 Description of the user interface

The SignSupport mobile application consists of two Graphical User Interfaces (GUIs), one for the pharmacist and the other for the Deaf user. The interface for the pharmacist appears with concise text instructions, a type-in box, multiple-choice buttons, or drop-down buttons. Whereby the interface for the Deaf user appears mainly in SASL videos, graphical buttons, and minimal text captions.
The pharmacist interface prompts the pharmacist to check the patient profile summary before dispensing any prescribed medication. At the end of the medication dispensing process, the pharmacist will hand back the phone to the Deaf user. The user will then enter his/her daily eating times. All that information will then be used to set the video alarm which will help remind the user when and how to take the given medication. The Deaf user interface also prompts the user to view the medication instructions videos. With the Video Alarm, the Deaf user will be reminded by the alarm that will vibrate on the phone showing the medicine picture and allow the user to review the video of how to take the medicine or dismiss the alarm. The two interfaces are designed to be clear, simple and easy to understand for both users.

6.2 User interface appearance to the user

The existing SignSupport application has different GUIs. The pharmacist will enter the instructions on how and when to take the medication (see Figure 5, left). The Frequency of use and Number of days on meds will be used to set the video alarm.

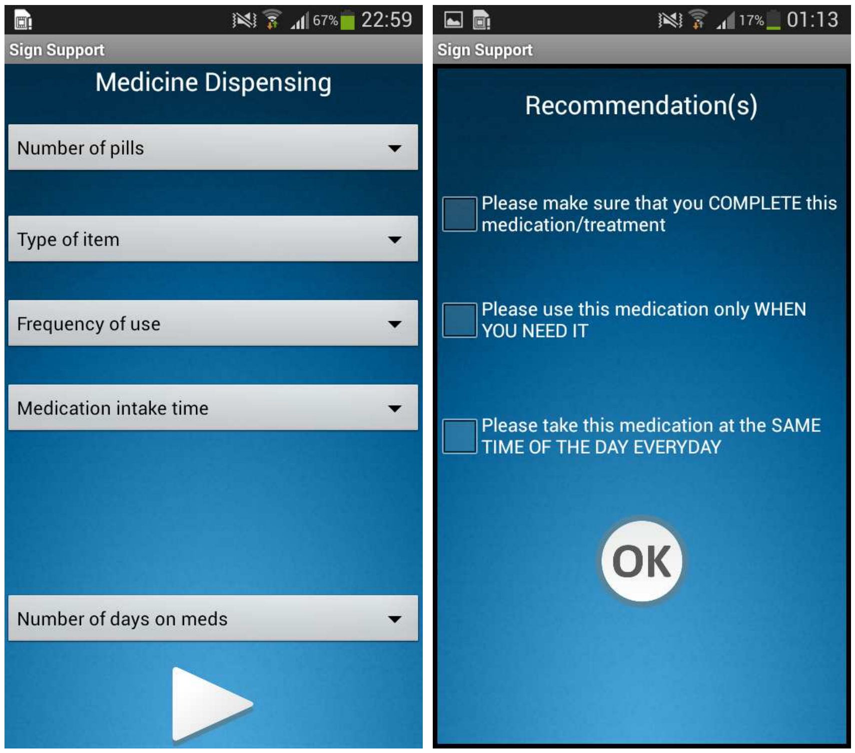

67% 22:59
Sign Support
Medicine Dispensing
Number of pills
Type of item
Frequency of use
Medication intake time
Number of days on meds
17% 01:13
Sign Support
Recommendation(s)
Please make sure that you COMPLETE this medication/treatment
Please use this medication only WHEN YOU NEED IT
Please take this medication at the SAME TIME OF THE DAY EVERYDAY
OK

Figure 5: SignSupport Dispencing and Recoomendations screen

After dispensing the medicine, the pharmacist will give Recommendations of the medicine use (see Figure 5, right).
After completing all the above steps, the information given by the pharmacist is written to text files. The pharmacist will hand over the phone to the user; the user will then enter his/her daily eating times (see Figure 6). When the user is done entering the eating times, all that information will be written to text files. The video alarm will be then automatically set using those text files.

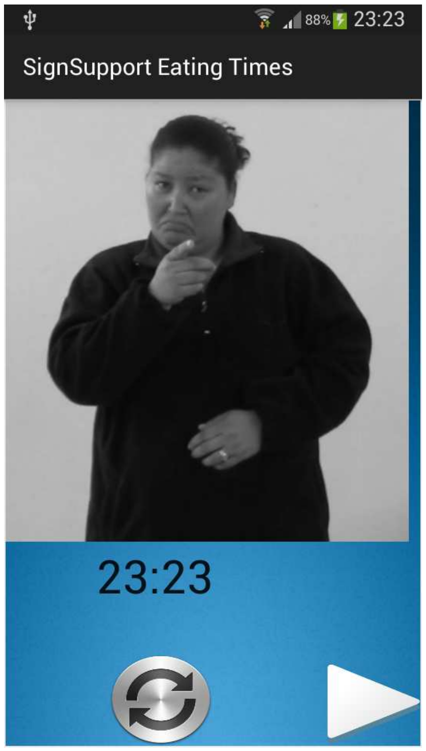


Figure 6: Video alarm settings screen

When it is time to take the medication, the mobile phone of the patient will vibrate and flashing light at the same time. When the patient attends the phone he/she will be able to view the instructions (in SASL) on how and when to take the medication by clicking on the green tick button, or dismiss the alarm by clicking on the red X button (see Figure 7).

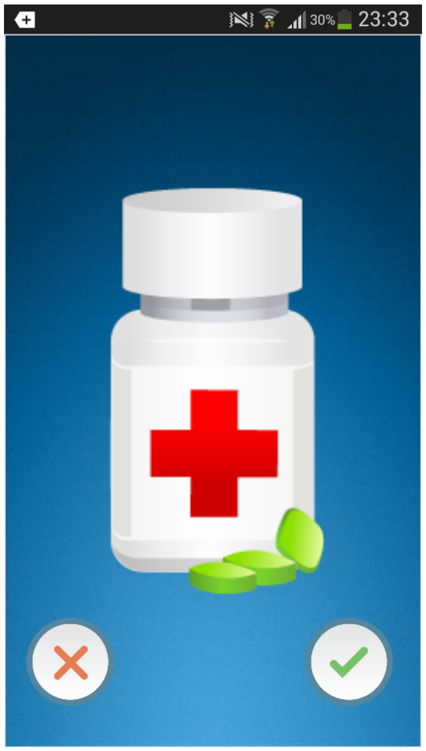


Figure 7: Video alarm screen

When the user clicks the green tick, the user will firstly see the image of the medicine to be taken, then followed by the medicine taking instructions (see Figure 8).

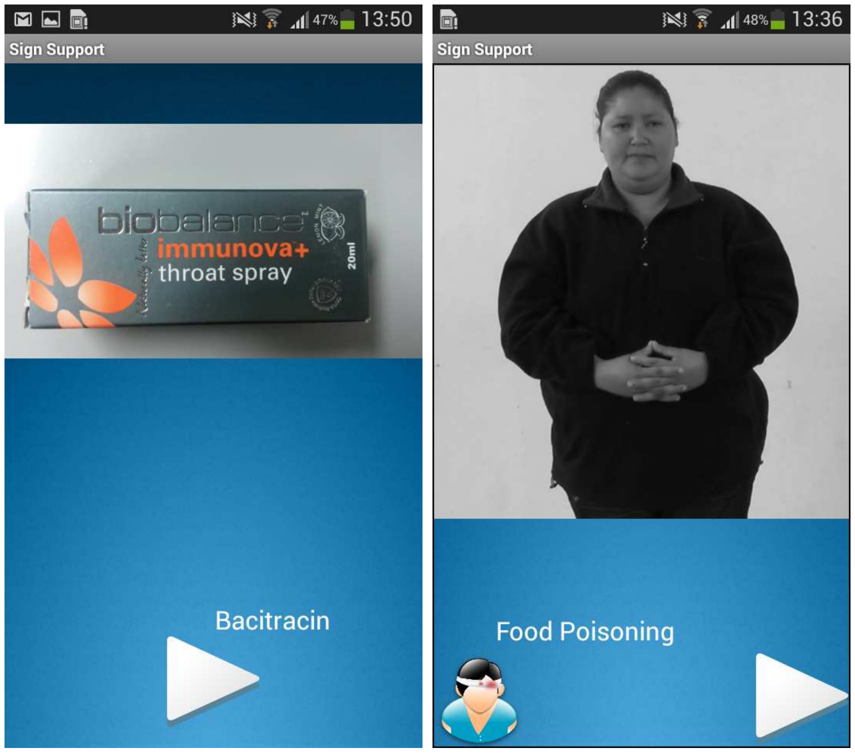


Figure 8: SignSupport condition screens

After playing the video of instructions, the patient will view the recommendations about taking the medicine (see Figure 9).

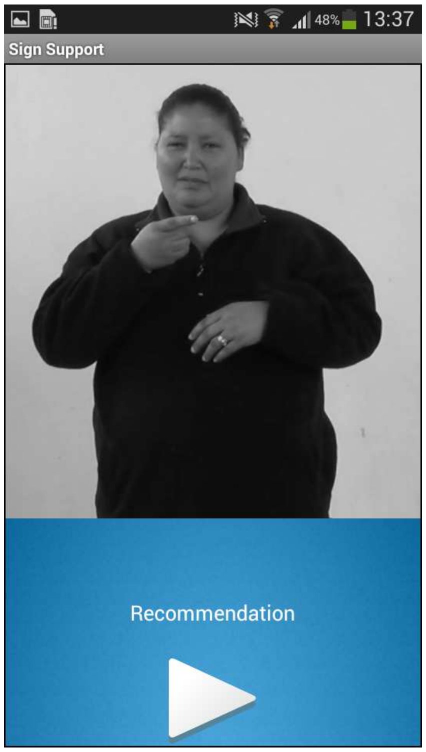


Figure 9: Recommendations screen

6.3 User interface behaviour

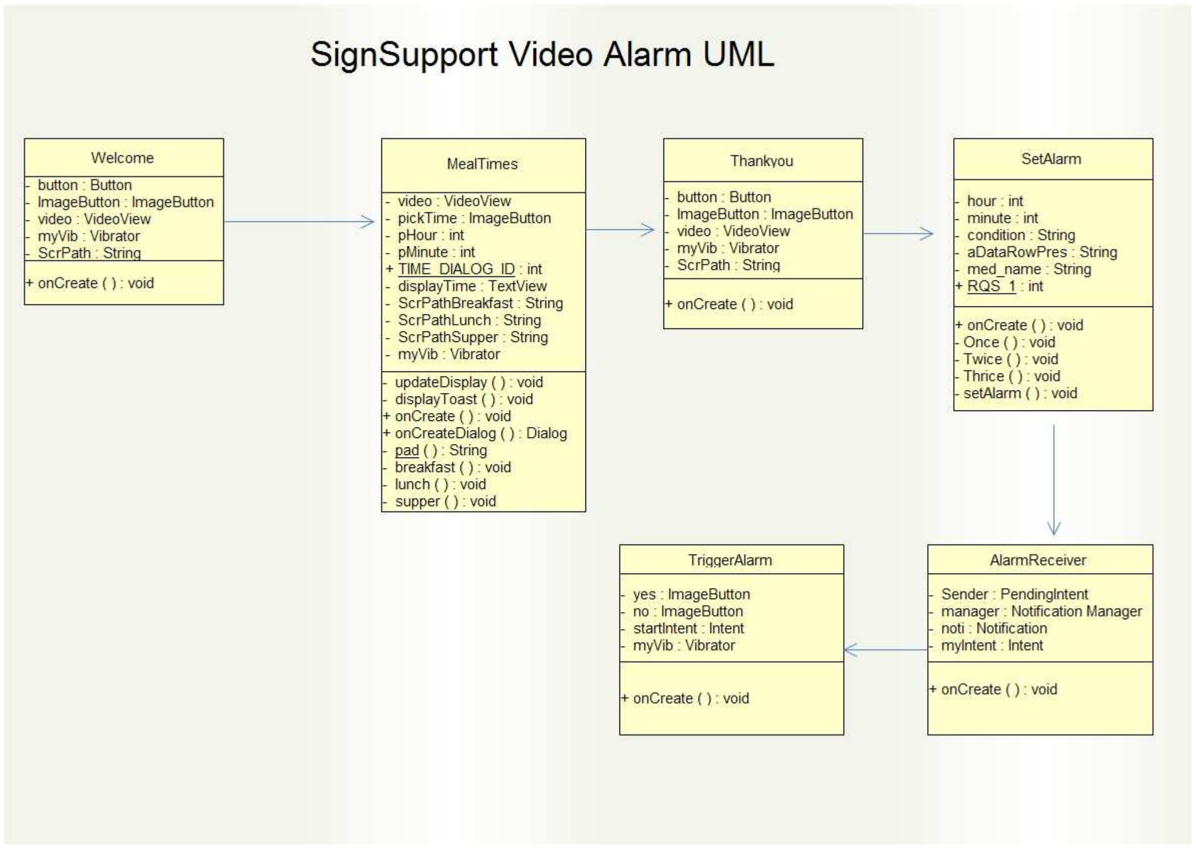
Video alarm is automatically set using the information from the pharmacist and the user, that information is written in the text files. Therefore the video alarm reads those files to automatically set the alarm. On the patient side, the mobile phone will vibrate and flash LED light at the same time. When the user attends the phone, the phone will show a picture of the medicine followed with two buttons bellow the picture. The left button will be a red X button for dismissing the alarm, and the green tick button will be for playing the SASL video for medicine taking instructions. As the SASL video is playing, the user can choose rewind the video when there is something not clear. Then user can also pause and fast-forward the video. After playing the instructions video, the user can also play the warnings and recommendations videos if need be.

6.4 User interaction with system

The user system on the pharmacist side allows the pharmacist to input medication instructions, warnings, and recommendations. While on the Deaf patient side it allows the patient to enter the daily eating times and play, pause and repeat the videos.

7 Object Oriented Analysis (High Level Design)

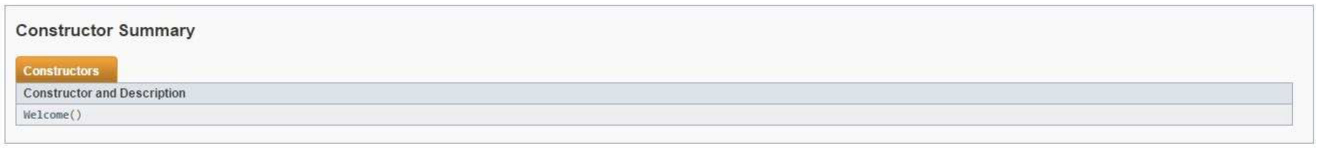
This section will focus on analysing the problem in terms of objects. A detailed breakdown of the technical solution will be discussed. The following UML diagram (see Figure 10.0) explains the flow of information of the video alarm. The Deaf user enters the breakfast, lunch, and supper eating times (MealTimes class). After that, the video alarm automatically reads the meal times information from the text files and set the alarm (SetAlarm class). If the user only takes the medication once a day, the method Once () is a called, if twice a day, the method Twice () is called, if its more than two times a day, then the method Thrice () is called.
When it time for the patient to take the medication, the class AlarmReceiver calls the class TriggerAlarm using intents. Then the TriggerAlarm class displays the alarm and vibrates, the user has an option to continue and view the medication instructions (yes button) or dismiss he alarm (no button).



8 Object Oriented Design (Low Level Design)

This section focuses on the classes that make up the video alarm and the settings. A brief description of each method in each class will be shown. Each class will be explained how it works. The brief description will be of a Javadoc HTML file automatically created by Android Studio.

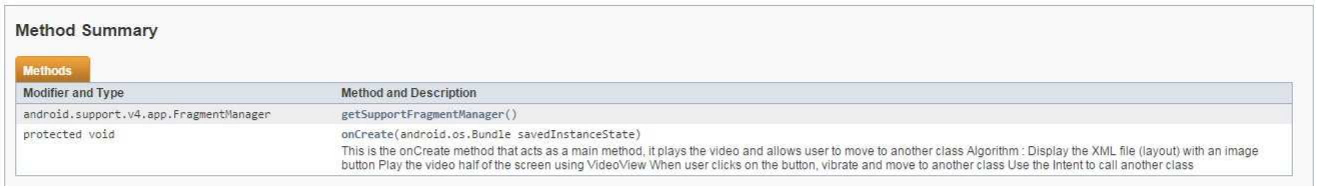
This is the Weclome class, which welcomes the Deaf user with a SASL video. This class has only one method, the onCreate method which acts as a main method. This is the welcome class, it plays the SASL welcoming video. It has a button to allow the user to continue to another video for setting the daily meals times. This class extends the ActionbarActivity android built-in class.



Constructor Summary

Constructors

Constructor and Description
`Welcome()`

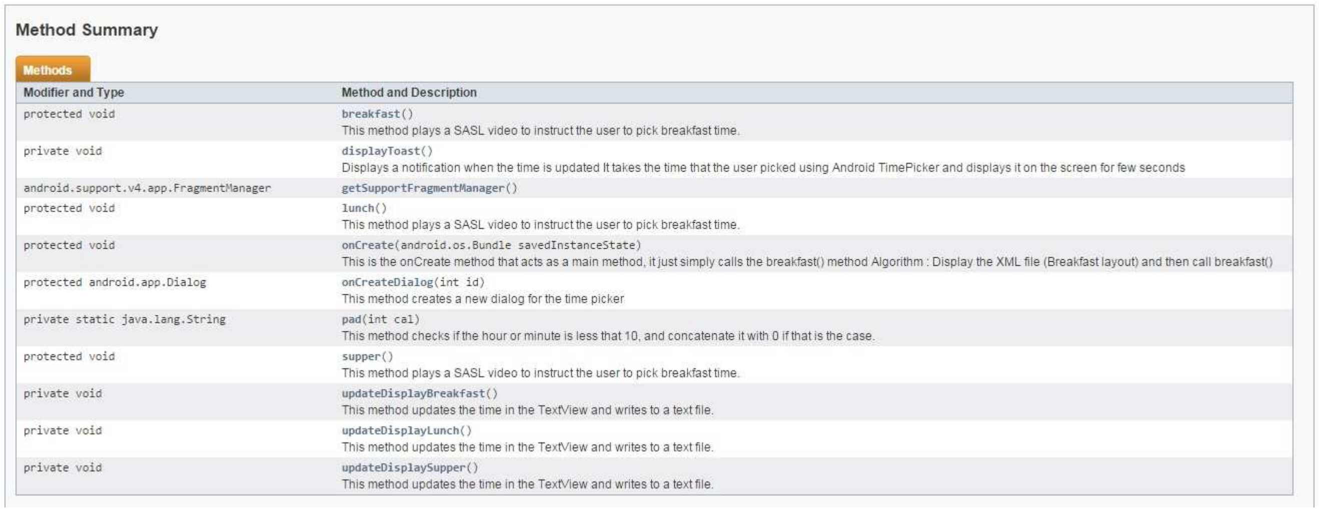


Method Summary

Methods

Modifier and Type	Method and Description
`android.support.v4.app.FragmentManager`	`getSupportFragmentManager()`
`protected void`	`onCreate(android.os.Bundle savedInstanceState)` This is the onCreate method that acts as a main method, it plays the video and allows user to move to another class Algorithm : Display the XML file (layout) with an image button Play the video half of the screen using VideoView When user clicks on the button, vibrate and move to another class Use the Intent to call another class

This is the meal times class where the Deaf user enters the times he/she eats the daily meals(breakfast, lunch, and supper. This class is for getting the user's daily eating times The three methods, breakfast, lunch, and supper write the eating times to three different text files. These three methods call each other, then the last method (supper) calls the following class (SetAlarm).



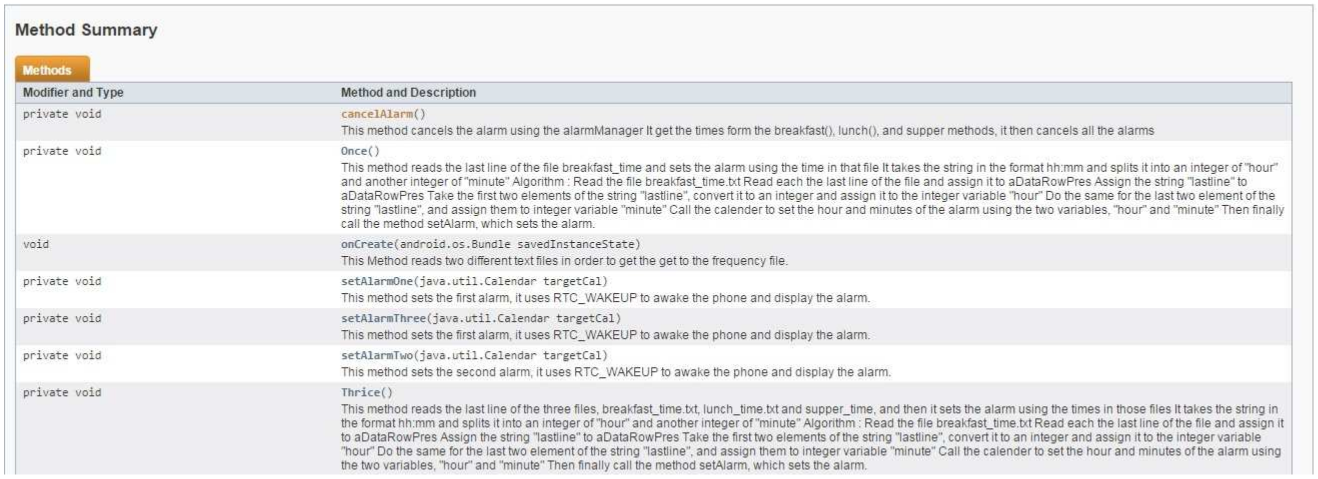
Method Summary

Methods

Modifier and Type	Method and Description
`protected void`	`breakfast()` This method plays a SASL video to instruct the user to pick breakfast time.
`private void`	`displayToast()` Displays a notification when the time is updated It takes the time that the user picked using Android TimePicker and displays it on the screen for few seconds
`android.support.v4.app.FragmentManager`	`getSupportFragmentManager()`
`protected void`	`lunch()` This method plays a SASL video to instruct the user to pick breakfast time.
`protected void`	`onCreate(android.os.Bundle savedInstanceState)` This is the onCreate method that acts as a main method, it just simply calls the breakfast() method Algorithm : Display the XML file (Breakfast layout) and then call breakfast()
`protected android.app.Dialog`	`onCreateDialog(int id)` This method creates a new dialog for the time picker
`private static java.lang.String`	`pad(int cal)` This method checks if the hour or minute is less that 10, and concatenate it with 0 if that is the case.
`protected void`	`supper()` This method plays a SASL video to instruct the user to pick breakfast time.
`private void`	`updateDisplayBreakfast()` This method updates the time in the TextView and writes to a text file.
`private void`	`updateDisplayLunch()` This method updates the time in the TextView and writes to a text file.
`private void`	`updateDisplaySupper()` This method updates the time in the TextView and writes to a text file.

This is the SetAlarm class that does the setting of the video alarm. This class is used to set the alarm, it has three methods that are responsible for

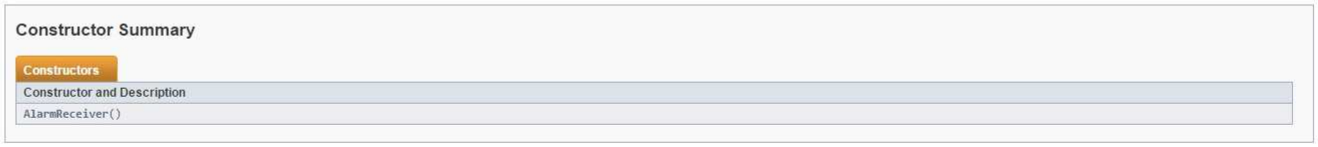
firing the alarm. breakfast(), lunch(), And supper()
When it is time for the alarm to vibrate, this class calls the AlarmReceive class, which also calls another class to display the alarm.


Method Summary

Methods

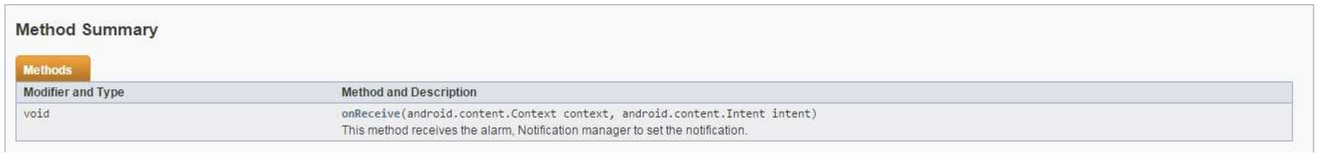
Modifier and Type	Method and Description
`private void`	`cancelAlarm()` This method cancels the alarm using the alarmManager It get the times form the breakfast(), lunch(), and supper methods, it then cancels all the alarms
`private void`	`Once()` This method reads the last line of the file breakfast_time and sets the alarm using the time in that file It takes the string in the format hh:mm and splits it into an integer of "hour" and another integer of "minute" Algorithm : Read the file breakfast_time.txt Read each the last line of the file and assign it to aDataRowPres Assign the string "lastline" to aDataRowPres Take the first two elements of the string "lastline", convert it to an integer and assign it to the integer variable "hour" Do the same for the last two element of the string "lastline", and assign them to integer variable "minute" Call the calender to set the hour and minutes of the alarm using the two variables, "hour" and "minute" Then finally call the method setAlarm, which sets the alarm.
`void`	`onCreate(android.os.Bundle savedInstanceState)` This Method reads two different text files in order to get the get to the frequency file.
`private void`	`setAlarmOne(java.util.Calendar targetCal)` This method sets the first alarm, it uses RTC_WAKEUP to awake the phone and display the alarm.
`private void`	`setAlarmThree(java.util.Calendar targetCal)` This method sets the first alarm, it uses RTC_WAKEUP to awake the phone and display the alarm.
`private void`	`setAlarmTwo(java.util.Calendar targetCal)` This method sets the second alarm, it uses RTC_WAKEUP to awake the phone and display the alarm.
`private void`	`Thrice()` This method reads the last line of the three files, breakfast_time.txt, lunch_time.txt and supper_time, and then it sets the alarm using the times in those files It takes the string in the format hh:mm and splits it into an integer of "hour" and another integer of "minute" Algorithm : Read the file breakfast_time.txt Read each the last line of the file and assign it to aDataRowPres Assign the string "lastline" to aDataRowPres Take the first two elements of the string "lastline", convert it to an integer and assign it to the integer variable "hour" Do the same for the last two element of the string "lastline", and assign them to integer variable "minute" Call the calender to set the hour and minutes of the alarm using the two variables, "hour" and "minute" Then finally call the method setAlarm, which sets the alarm.

This is the class that receives the alarm and calls another class. This class just receives the alarm, put a notification on the notification bar of the phone and then call the TriggerAlarm class


Constructor Summary

Constructors

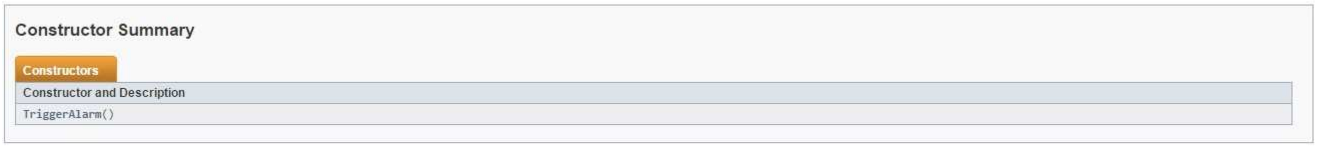
Constructor and Description
`AlarmReceiver()`


Method Summary

Methods

Modifier and Type	Method and Description
`void`	`onReceive(android.content.Context context, android.content.Intent intent)` This method receives the alarm, Notification manager to set the notification.

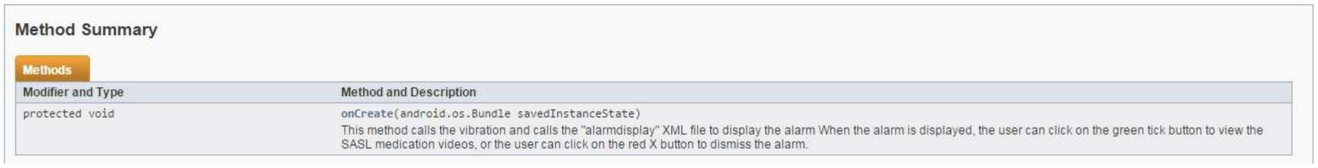
This is the trigger alarm that displays the alarm. This class displays the alarm with the picture of the medicine. The user can choose to continue and view the medication instructions, or dismiss the alarm using the two provided buttons.



Constructor Summary

Constructors

Constructor and Description
`TriggerAlarm()`



Method Summary

Methods

Modifier and Type	Method and Description
`protected void`	`onCreate(android.os.Bundle savedInstanceState)` This method calls the vibration and calls the "alarmdisplay" XML file to display the alarm When the alarm is displayed, the user can click on the green tick button to view the SASL medication videos, or the user can click on the red X button to dismiss the alarm.

9 Implementation

The target environment of SignSupport is Android Operating Systems. Since the video alarm is an add-on to SignSupport, it must also be developed for Android Operating System. The programming language used for developing the video alarm is Java with XML. XML is used to design and implement the interface of the application. Java acts as a back-end; it controls the functionality of the application. Java controls what must happen when a certain button or position of the interface is clicked, for example.

9.1 Tools Used

To develop the video alarm I used Android Studio IDE, because it supports the development of Android applications and it easy to use as compared to other IDEs such as Eclipse. I used the Samsung S4-mini for testing the progress of my project. The IDE was installed in a Desktop computer (i5). I also used Chrome web browser to view the java documentation.

9.2 Code Documentation

Android Studio

Android studio allows automatic generation of the Javadoc; it automatically generates the HTML file of the Javadoc with all the methods and parameters in each class.
The following are snippets of the video alarm Javadocs, for a detailed Javadoc please go to http://www.cs.uwc.ac.za/ vphindiso/Javadoc.rar

ImageButton

```
android.widget.ImageButton ImageButton
```

video

```
private android.widget.VideoView video
```

myVib

```
private android.os.Vibrator myVib
```

SrcPath

```
java.lang.String SrcPath
```

Constructor Detail

Welcome

```
public Welcome()
```

Method Detail

onCreate

```
protected void onCreate(android.os.Bundle savedInstanceState)
```

This is the onCreate method that acts as a main method, it plays the video and allows user to move to another class Algorithm : Display the XML file (layout) with an image button Play the video half of the screen using VideoView When user clicks on the button, vibrate and move to another class Use the Intent to call another class

Overrides:

`onCreate` in class `android.support.v7.app.ActionBarActivity`

Parameters:

`savedInstanceState` - Saving the activity state.

`ImageButton` - - click to go to new Breakfast.class

`myVib` - Vibrate

`myVideoView` - play the video

See Also:

```
Activity.onStart(), Activity.onSaveInstanceState(android.os.Bundle),
Activity.onRestoreInstanceState(android.os.Bundle),
Activity.onPostCreate(android.os.Bundle)
```

getSupportFragmentManager

```
public android.support.v4.app.FragmentManager getSupportFragmentManager()
```

Package Class Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

```
getClassLoader, getCodeCacheDir, getContentResolver, getDatabasePath, getDir,
getDisplayAdjustments, getExternalCacheDir, getExternalCacheDirs, getExternalFilesDir,
getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper,
getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath,
getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences,
getSharedPrefsFile, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight,
getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput,
openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper,
registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast,
removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast,
sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendOrderedBroadcast,
sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser,
sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser,
sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper,
startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser,
stopService, stopServiceAsUser, unbindService, unregisterReceiver
```

Methods inherited from class android.content.Context

```
getDrawable, getString, getString, getText, obtainStyledAttributes,
obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes,
registerComponentCallbacks, unregisterComponentCallbacks
```

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Field Detail

video

```
private android.widget.VideoView video
```

TIME_DIALOG_ID

```
static final int TIME_DIALOG_ID
```

See Also:

Constant Field Values

displayTime

```
android.widget.TextView displayTime
```

myVib

```
private android.os.Vibrator myVib
```

pickTime

```
android.widget.ImageButton pickTime
```

pHour

```
int pHour
```

pMinute

```
int pMinute
```

SrcPathBreakfast

```
java.lang.String SrcPathBreakfast
```

SrcPathLunch

```
java.lang.String SrcPathLunch
```

SrcPathSupper

```
java.lang.String SrcPathSupper
```

mTimeSetListener

```
private android.app.TimePickerDialog.OnTimeSetListener mTimeSetListener
```

Callback received when the user "picks" a time in the dialog

Constructor Detail

Breakfast

```
public Breakfast()
```

Method Detail

updateDisplayBreakfast

```
private void updateDisplayBreakfast()
```

This method updates the time in the TextView and writes to a text file. Algorithm : call the text viewer displayTime to display the time. Concatenate the three strings, pHour, : , and pMinute to output good looking form of time. Create a new text file (breakfast_time.txt) and append the time on it. If : if the file is empty append time on the same line else : append time on a new line try and catch, in case the file path ic not available, display the error message

Parameters:

`time` - String time that is writeen on the text file (breakfast_time.txt)

updateDisplayLunch

```
private void updateDisplayLunch()
```

This method updates the time in the TextView and writes to a text file. Algorithm : call the text viewer displayTime to display the time. Concatenate the three strings, pHour, : , and pMinute to output good looking form of time. Create a new text file (lunch_time.txt) and append the time on it. If : if the file is empty append time on the same line else : append time on a new line try and catch, in case the file path ic not available, display the error message

Parameters:

`time` - String time that is written on the text file (lunch_time.txt)

updateDisplaySupper

```
private void updateDisplaySupper()
```

This method updates the time in the TextView and writes to a text file. Algorithm : call the text viewer displayTime to display the time. Concatenate the three strings, pHour, : , and pMinute to output good looking form of time. Create a new text file (supper_time.txt) and append the time on it. If : if the file is empty append time on the same line else : append time on a new line try and catch, in case the file path ic not available, display the error message

Parameters:

`time` - String time that is written on the text file (supper_time.txt)

displayToast

```
private void displayToast()
```

Displays a notification when the time is updated It takes the time that the user picked using Android TimePicker and displays it on the screen for few seconds

onCreate

```
protected void onCreate(android.os.Bundle savedInstanceState)
```

This is the onCreate method that acts as a main method, it just simply calls the breakfast() method Algorithm : Display the XML file (Breakfast layout) and then call breakfast()

Overrides:

`onCreate` in class `android.support.v7.app.ActionBarActivity`

Parameters:

`savedInstanceState` - Saving the activity state.

See Also:

```
Activity.onStart(), Activity.onSaveInstanceState(android.os.Bundle),
Activity.onRestoreInstanceState(android.os.Bundle),
Activity.onPostCreate(android.os.Bundle)
```

breakfast

```
protected void breakfast()
```

This method plays a SASL video to instruct the user to pick breakfast time. The user clicks on the silver button to select the time and set it. After selecting the time, the selected time will be displayed on the screen. Then the user will simply press the white arrow button to go to the next class, Algorithm : Play the video using VideoView Pick the time you want to set - the button pickTime displays the Android Time picker Set the time. The time set by the user will be displayed on the screen Then click the button btn to move to the method lunch()

Parameters:

`btn` - click to go to lunch method

`myVib` - Vibrate

`myVideoView` - play the video

`displayTime` - display the time set by user

`pickTime` - ImageButton to view TimePicker and select the time

lunch

```
protected void lunch()
```

This method plays a SASL video to instruct the user to pick breakfast time. The user clicks on the silver button to select the time and set it. After selecting the time, the selected time will be displayed on the screen. Then the user will simply press the white arrow button to go to the next method. Algorithm : Play the video using VideoView Pick the time you want to set - the button pickTime displays the Android Time picker Set the time. The time set by the user will be displayed on the screen Then click the button btn to move to the method lunch()

Parameters:

`btn` - click to go to lunch method

`myVib` - Vibrate

`myVideoView` - play the video

`displayTime` - display the time set by user

`pickTime` - ImageButton to view TimePicker and select the time

supper

```
protected void supper()
```

This method plays a SASL video to instruct the user to pick breakfast time. The user clicks on the silver button to select the time and set it. After selecting the time, the selected time will be displayed on the screen. Then the user will simply press the white arrow button to go to the next class (SetAlarm). Algorithm : Play the video using VideoView Pick the time you want to set - the button pickTime displays the Android Time picker Set the time. The time set by the user will be displayed on the screen Then click the button btn to move to the class SetAlarm

Parameters:

`btn` - click to go to lunch method

`myVib` - Vibrate

`myVideoView` - play the video

`displayTime` - display the time set by user

`pickTime` - ImageButton to view TimePicker and select the time

onCreateDialog

```
protected android.app.Dialog onCreateDialog(int id)
```

This method creates a new dialog for the time picker

Overrides:

`onCreateDialog` in class `android.app.Activity`

Parameters:

`id` -

Returns:

null

pad

```
private static java.lang.String pad(int cal)
```

This method checks if the hour or minute is less that 10, and concatenate it with 0 if that is the case. Algorithm : Check if cal is less than 10 if cal >= 10: convert cal to string and return it else: convert cal to string, add "0" at the beggining of cal and return cal with "0"

Parameters:

`cal` - used to check the hour and minute

Returns:

"0" + String.valueOf(cal)

getSupportFragmentManager

```
public android.support.v4.app.FragmentManager getSupportFragmentManager()
```

```
sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper,
startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser,
stopService, stopServiceAsUser, unbindService, unregisterReceiver
```

Methods inherited from class android.content.Context

```
getDrawable, getString, getString, getText, obtainStyledAttributes,
obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes,
registerComponentCallbacks, unregisterComponentCallbacks
```

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Field Detail

myTimePicker

```
android.widget.TimePicker myTimePicker
```

buttonstartSetDialog

```
android.widget.Button buttonstartSetDialog
```

buttonCancelAlarm

```
android.widget.Button buttonCancelAlarm
```

textAlarmPrompt

```
android.widget.TextView textAlarmPrompt
```

timePicker

```
android.widget.TimePicker timePicker
```

optRepeat

```
android.widget.CheckBox optRepeat
```

hour

```
int hour
```

minute

```
int minute
```

hour2

```
int hour2
```

minute2

```
int minute2
```

RQS_1

```
static final int RQS_1
```

See Also:

Constant Field Values

Constructor Detail

SetAlarm

```
public SetAlarm()
```

Method Detail

onCreate

```
public void onCreate(android.os.Bundle savedInstanceState)
```

This Method reads two different text files in order to get the get to the frequency file. The first file is condition.txt, which holds the name of the user's condition. The following file is ned_name.txt, which holds the name of the medicine for the user's condition The las file is frequency.txt, which holds the number of times the user has to take the medication. Algorithm : read the file frequency.txt, checks the first element in the file, If the first element is "1" then the method Once() is called Else if first element is "2" then the method Twice() is called Else, the method Thrice() is called

Overrides:

`onCreate` in class `android.app.Activity`

Parameters:

`savedInstanceState` - Saving the activity state.

`aDataRowPres` - Empty String that reads each line of the file

See Also:

`Activity.onStart()`, `Activity.onSaveInstanceState(android.os.Bundle)`, `Activity.onRestoreInstanceState(android.os.Bundle)`, `Activity.onPostCreate(android.os.Bundle)`

Once

`private void Once()`

This method reads the last line of the file breakfast_time and sets the alarm using the time in that file It takes the string in the format hh:mm and splits it into an integer of "hour" and another integer of "minute" Algorithm : Read the file breakfast_time.txt Read each the last line of the file and assign it to aDataRowPres Assign the string "lastline" to aDataRowPres Take the first two elements of the string "lastline", convert it to an integer and assign it to the integer variable "hour" Do the same for the last two element of the string "lastline", and assign them to integer variable "minute" Call the calender to set the hour and minutes of the alarm using the two variables, "hour" and "minute" Then finally call the method setAlarm, which sets the alarm. Apply try and catch, in case the file path is not found.

Parameters:

`aDataRowPres` - This is the empty string that reads the file

`hour` - This is an integer that is read from the file, the last two elements of the string, this string is then parsed to integer

`minute` - This is an integer that is read from the file, the first two elements of the string, this string is then parsed to integer

Twice

`private void Twice()`

This method reads the last line of the two files, breakfast_time.txt and lunch_time.txt, and sets the alarm using the times in those files It takes the string in the format hh:mm and splits it into an integer of "hour" and another integer of "minute" Algorithm : Read the file breakfast_time.txt Read each the last line of the file and assign it to aDataRowPres Assign the string "lastline" to aDataRowPres Take the first two elements of the string "lastline", convert it to an integer and assign it to the integer variable "hour" Do the same for the last two element of the string "lastline", and assign them to integer variable "minute" Call the calender to set the hour and minutes of the alarm using the two variables, "hour" and "minute" Then finally call the method setAlarm, which sets the alarm. Apply try and catch, in case the file path is not found. Apply the same algorithm to both files.

Parameters:

`aDataRowPres` - This is the empty string that reads the file

`hour` - This is an integer that is read from the file, the last two elements of the string, this string is then parsed to integer

`minute` - This is an integer that is read from the file, the first two elements of the string, this string is then parsed to integer

Thrice

`private void Thrice()`

This method reads the last line of the three files, breakfast_time.txt, lunch_time.txt and supper_time, and then it sets the alarm using the times in those files It takes the string in the format hh:mm and splits it into an integer of "hour" and another integer of "minute" Algorithm : Read the file breakfast_time.txt Read each the last line of the file and assign it to aDataRowPres Assign the string "lastline" to aDataRowPres Take the first two elements of the string "lastline", convert it to an integer and assign it to the integer variable "hour" Do the same for the last two element of the string "lastline", and assign them to integer variable "minute" Call the calender to set the hour and minutes of the alarm using the two variables, "hour" and "minute" Then finally call the method setAlarm, which sets the alarm. Apply try and catch, in case the file path is not found. Apply the same algorithm to both files.

Parameters:

`aDataRowPres` - This is the empty string that reads the file

`hour` - This is an integer that is read from the file, the last two elements of the string, this string is then parsed to integer

`minute` - This is an integer that is read from the file, the first two elements of the string, this string is then parsed to integer

setAlarmOne

```
private void setAlarmOne(java.util.Calendar targetCal)
```

This method sets the first alarm, it uses RTC_WAKEUP to awake the phone and display the alarm. It calls the class Alarm.Receiver which displays the alarm This method allows the user to repeat the alarm every 24 hours (everyday) after it has started to ring or vibrate(i mean). LOL

Parameters:

`targetCal` - variable used to call the Android Calendar

`repeat` - Repeats the alarm after 24 hours (everyday)

setAlarmTwo

```
private void setAlarmTwo(java.util.Calendar targetCal)
```

This method sets the second alarm, it uses RTC_WAKEUP to awake the phone and display the alarm. It calls the class Alarm.Receiver which displays the alarm This method allows the user to repeat the alarm every 24 hours (everyday) after it has started to ring or vibrate(i mean). LOL

Parameters:

`targetCal` - variable used to call the Android Calendar

`repeat` - Repeats the alarm after 24 hours (everyday)

setAlarmThree

```
private void setAlarmThree(java.util.Calendar targetCal)
```

This method sets the first alarm, it uses RTC_WAKEUP to awake the phone and display the alarm. It calls the class Alarm.Receiver which displays the alarm This method allows the user to repeat the alarm every 24 hours (everyday) after it has started to ring or vibrate(i mean). LOL

Parameters:

`targetCal` - variable used to call the Android Calendar

`repeat` - Repeats the alarm after 24 hours (everyday)

cancelAlarm

```
private void cancelAlarm()
```

This method cancels the alarm using the alarmManager It get the times form the breakfast(), lunch(), and supper methods, it then cancels all the alarms

Constructor Detail

AlarmReceiver

```
public AlarmReceiver()
```

Method Detail

onReceive

```
public void onReceive(android.content.Context context,
             android.content.Intent intent)
```

This method receives the alarm, Notification manager to set the notification. Use PendingIntent to send the notification and then display the notification. Then use Intent to call the anther class (TriggerAlarm)

Specified by:

`onReceive` in class `android.content.BroadcastReceiver`

Parameters:

`context` - used to start the new activity

`intent` - used to call another class

```
getExternalFilesDirs, getExternalMediaDirs, getFilesDir, getFileStreamPath, getMainLooper,
getNoBackupFilesDir, getObbDir, getObbDirs, getOpPackageName, getPackageCodePath,
getPackageManager, getPackageName, getPackageResourcePath, getSharedPreferences,
getSharedPrefsFile, getUserId, getWallpaper, getWallpaperDesiredMinimumHeight,
getWallpaperDesiredMinimumWidth, grantUriPermission, isRestricted, openFileInput,
openFileOutput, openOrCreateDatabase, openOrCreateDatabase, peekWallpaper,
registerReceiver, registerReceiver, registerReceiverAsUser, removeStickyBroadcast,
removeStickyBroadcastAsUser, revokeUriPermission, sendBroadcast, sendBroadcast,
sendBroadcast, sendBroadcastAsUser, sendBroadcastAsUser, sendOrderedBroadcast,
sendOrderedBroadcast, sendOrderedBroadcast, sendOrderedBroadcastAsUser,
sendOrderedBroadcastAsUser, sendStickyBroadcast, sendStickyBroadcastAsUser,
sendStickyOrderedBroadcast, sendStickyOrderedBroadcastAsUser, setWallpaper, setWallpaper,
startActivitiesAsUser, startInstrumentation, startService, startServiceAsUser,
stopService, stopServiceAsUser, unbindService, unregisterReceiver
```

Methods inherited from class android.content.Context

```
getDrawable, getString, getString, getText, obtainStyledAttributes,
obtainStyledAttributes, obtainStyledAttributes, obtainStyledAttributes,
registerComponentCallbacks, unregisterComponentCallbacks
```

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait
```

Field Detail

myVib

```
private android.os.Vibrator myVib
```

LED_NOTIFICATION_ID

```
static final int LED_NOTIFICATION_ID
```

See Also:

Constant Field Values

Constructor Detail

TriggerAlarm

```
public TriggerAlarm()
```

Method Detail

onCreate

```
protected void onCreate(android.os.Bundle savedInstanceState)
```

This method calls the vibration and calls the "alarmdisplay" XML file to display the alarm When the alarm is displayed, the user can click on the green tick button to view the SASL medication videos, or the user can click on the red X button to dismiss the alarm. Algorithm : Display the alarm with medicine picture using setContentView Vibrate the phone three time for long strong vibration If user click on the red X button: call the home screen of the phone (exit the alarm) If user clicks on the green tick button: call the home class of SignSupport using Intent

Overrides:

`onCreate` in class `android.app.Activity`

Parameters:

`savedInstanceState` - Saving the activity state.

See Also:

`Activity.onStart()`, `Activity.onSaveInstanceState(android.os.Bundle)`, `Activity.onRestoreInstanceState(android.os.Bundle)`, `Activity.onPostCreate(android.os.Bundle)`

```
android.widget.ImageButton ImageButton
```

video

```
private android.widget.VideoView video
```

adapter

```
android.widget.ArrayAdapter<java.lang.CharSequence> adapter
```

myVib

```
private android.os.Vibrator myVib
```

SrcPath

```
java.lang.String SrcPath
```

Constructor Detail

Thankyou

```
public Thankyou()
```

Method Detail

onCreate

```
protected void onCreate(android.os.Bundle savedInstanceState)
```

This is the onCreate method that acts as a main method, it plays the video and allows user to go to SignSupport home page Algorithm : Display the XML file (layout) with an image button Play the video half of the screen using VideoView When user clicks on the button, vibrate and move to another class Use the Intent to call another class

Overrides:

`onCreate` in class `android.support.v7.app.ActionBarActivity`

Parameters:

`savedInstanceState` - Saving the activity state.

`ImageButton` - - click to go to new Breakfast.class

`myVib` - Vibrate

See Also:

`Activity.onStart()`, `Activity.onSaveInstanceState(android.os.Bundle)`, `Activity.onRestoreInstanceState(android.os.Bundle)`, `Activity.onPostCreate(android.os.Bundle)`

getSupportFragmentManager

```
public android.support.v4.app.FragmentManager getSupportFragmentManager()
```

Project Plan

Term / Week	Tasks
TERM 1 (Plan)	• Background reading SignSupport • Read previous video notification project documentation • Gather Video notification user requirements • Analyse the user requirements for the video notification
Term 2 (Model)	• Create a use-case diagram • Create a story board • Design the video notification application (prototype) • Verify the design against the user requirements • Produce a pseudo code for design • Write a report and prepare presentation
Term 3 (Construct): Week 1 : Week 2 : Week 3 : Week 4 : Week 5 : Week 6 : Week 7	• Design video alarm UML • Start video alarm implementation • Make changes to the code (combine classes) • Combine settings with the alarm • Test the alarm • Fix previous document and write a new one • Merge video alarm with SignSupport
Term 4 (Test)	• Analyse feedback from the previous stage's results • Fix Bugs • Modify the video notification to make it more efficient • Write documentation and a presentation of the final version the video notification

References

[1] June Isaacson Kailes, MSW, Associate Director Christie Mac Donald MPP (26-07-2009). *pharmacies and serving people with disabilities.*

[2] Koos Looijesteijn (21-08-2012). SignSupport blog, *The blog about the solution for Deaf and hearing people who want to talk to one another.*

[3] DeafSA(2003).*Information Booklet. South Africa*

[4] M. Glaser and W. Tucker(2004). Telecommunications bridging between deaf and hearing users in south africa.*In Proc. Conference and Workshop on Assistive Technologies for Vision and Hearing Impaired*

[5] M. Parker M. Mothlabi, M. Glaser and W. Tucker(2013). Signsupport : *A limited communication domain mobile aid for a deaf patient at the pharmacy. In Proc. SATNAC*

[6] Prangnat Chininthorn (September 2011). *Communication Tool Design for Deaf to Hearing in South Africa*

[7] Medisafe Project(2014). Medisafe meds and pill reminder. *https: // play. google. com/ store/ apps/ details? id= com. medisafe. android. client .*

[8] Michael B. Motlhabi,William D. Tucker, Mariam B. Parker, *Improving Usability and Correctness of a Mobile Tool to help a Deaf person with Pharmaceutical Instruction*