SOS Emergency Application

End Of Semester One Report

End Of Semester One Report

Name: Justin Wu
ID:3974847
UPI: ywu126

University Of Auckland

TABLE OF CONTENTS

1. INTRODUCTION **4**

2. COMPANY **4**

3. GOAL **4**

4. ANDROID MOBILE **5**

4.1. Introduction on Android 5

4.2. Android Operating System 5

4.3. Writing an Android Application 6

5. IOS FOR IPHONE **7**

5.1. Introduction on iOS 7

5.2. iPhone Operation System 7

5.3. Writing an iPhone Application (iOS) 8

6. COMPRASION FOR IPHONE AND ANDROID **9**

6.1. Closed System vs. Open System 9

6.2. Security 9

6.3. Vender Lock in 9

6.4. User Control 10

6.5. Variations of Mobile Device 10

6.6. Battery Life 10

7. PROS AND CONS FOR ANDROID AND IPHONE PLATFORM **11**

8. RESEARCH WORK **12**

8.1. Falling Accident 12

8.2. Car Accident 13

9. RESEARCH ON RELATED APPLICATION **15**

10. DESIGNS AND USER INTERFACE **16**

10.1. First View 16

10.2. Setting View 17

11. IMPLEMENTATION FOR THE APPLICATION **19**

11.1. Setting Up the Initial and “Setting” View 19

11.2. Setting Up to the Button to Change Between the Two Views 20

11.3. Sending Text Message 21

11.4. Setting Up and Display GPS Information 22

11.5. Setting Up and Display 23

12. COMPLEXCATIONS **24**

13. FUTURE WORK **24**

14. CONCLUSION **24**

15. REFERENCES **25**

1. INTRODUCTION

We are stepping our feet into a millennium, an era where mobile phones are not only a important gadget for our daily lives but an essential tool that is needed among all walks of life. This paper proposes my project of designing an application that detects when the user has encountered in an accident, consequently our mobile application will send out an emergency SMS to our selected contact.

Step by step I will explain my discoveries and findings. Along with the way I am implementing and researching.

All in all I'm very excited about making this happen. Not only to prove that mobile applications are not only fun games for the young but they can also serve an important function in your world. You'll never know when it could just save your's or your loved ones life.

2. COMPANY

For this project, I am going to be working with a company called the BLACKHAWK. The BLACKHAWK is a New Zealand owned company, established by two flatmates Andrew Radcliffe and Andrew Sharp. The idea came to them when Andrew Radcliffe's car was stolen and they could not find a suitable anti-theft device to prevent this from happening again.

The BlackHawk is now a recognized company operating in New Zealand and Australia, with potential to become world leading. They provide anti-theft and tracking system for both personal users and business owners to monitor and track their vehicles.

3. GOAL

The goal of this project is to create a SOS emergency application of different mobile operating systems. The application will need to be able to sense collision accidents and send an emergency message to the contact the user has assigned to. The application will need to be able to determine what kind of transporting method the users are currently using and to sense different kind of collisions accordingly. Transporting method include walking and driving by car or motorbike. The application will need to use functions provided in the mobile device to determine the collisions.

4. ANDROID MOBILE

4.1. Introduction on Android

The Android operating system is an open-source platform for mobile devices based on Linux, it is developed by Google and the Open Handset Alliance (OHA). This alliance contains major mobile operators, software and hardware manufactures. Their aim is to introduce an open source software into the current mobile market to allow different companies to cooperate faster to match the emerging user needs.

4.2. Android Operating System

The figure 1 below shows the major components in Android Operation Systems.

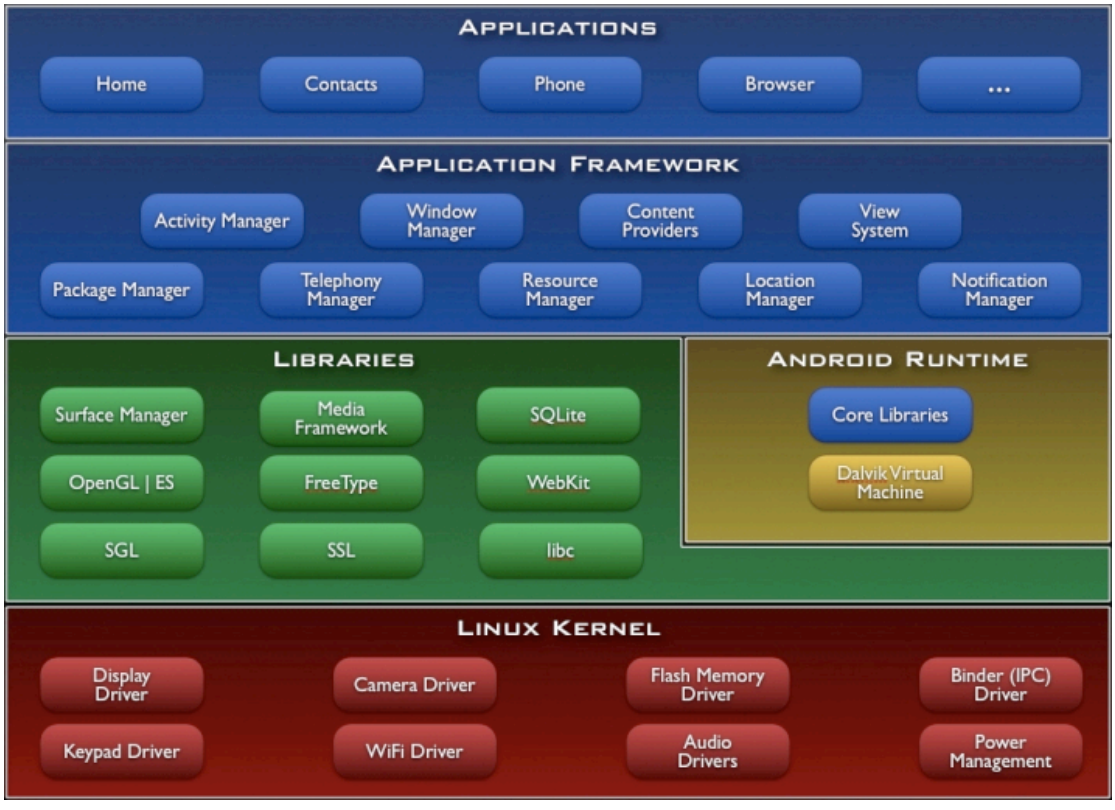

Figure 1. Android Architecture

As presented above, Android architecture is composed of five layers:

- Application
- Application Framework
- Native Libraries
- Android Runtime
- Linux kernel

The Linux Kernel is the layer lying between the mobile hardware and the Android platform, it provide the core service like security, memory management, process management, network stack and driver model

The set of libraries supports several standard mechanisms, such SSL, openGL, SGL, system C library or multimedia codes. The Android runtime is the core of the Android platform, managing all applications and services running. This layer manages both the lifecycle of the processes (like applications and services) and the corresponding resources (e.g. memory and processing capacity). It has two main components; the core libraries, which are java applications, program interfaces (APIs) to manage the data structure or network access and the Dalvik Virtual Machine, which runs Android applications compiled by a Java language compiler.

The application framework consists of several services that allow managing the system resources, such as window or notification management. Some services are internal to the system, while others provide and API to the applications, like GPS or Wi-Fi.

The final layer is the applications layer. These are the applications that are visible to the users. These applications run over the Dalvik virtual machine and use the application framework to access the services and resources of the system. Each application runs over a single instance of the virtual machine, so the processers are independent of each other.

4.3. Writing an Android Application

To begin programming Android applications, tools and Android SDK need to be installed firstly on to the computer. I will be using Eclipse as my developing tool. The three steps shows below show how to set up eclipse with Android SDK (assuming Eclipse is already installed in the computer):

1. Download the SDK package from http://developer.android.com/sdk/installing.html and either unpack it (if it is zip or tgz file) or install it (if it is exe file) to a safe location on your machine.
2. Install the Android Development Tool (ADT) plug-in for Eclipse. By following the instruction in http://developer.android.com/sdk/eclipse-adt.html#installing.
3. Setting up the SDK using the Android SDK and AVD that will now be available in the top menu of the Eclipse by selecting Window > Android SDK and AVD Manager. The Android SDK and AVD Manager will allow you to installs components in your SDK environment. (e.g. Android Platforms and SDK Tools). After installing the prefer Android platform you will be working with, you now can create a Android Project in Eclipse.

5. IOS FOR IPHONE

5.1. Introduction on iOS

Formally known as iPhone OS, iSO is the Apple's mobile operating system that is used to run a number of their hand held devices. For example iPhone, IPad and IPod touch. With over 300,000 iOS applications available in the apple market, it is the most popular app store than any other mobile devices. iOS uses a multi-touched interface, so that simple movement like swiping fingers across screen to move pages can used to operate the device.

5.2. iPhone Operating System.

The figure 2 below shows the major components in iOS.

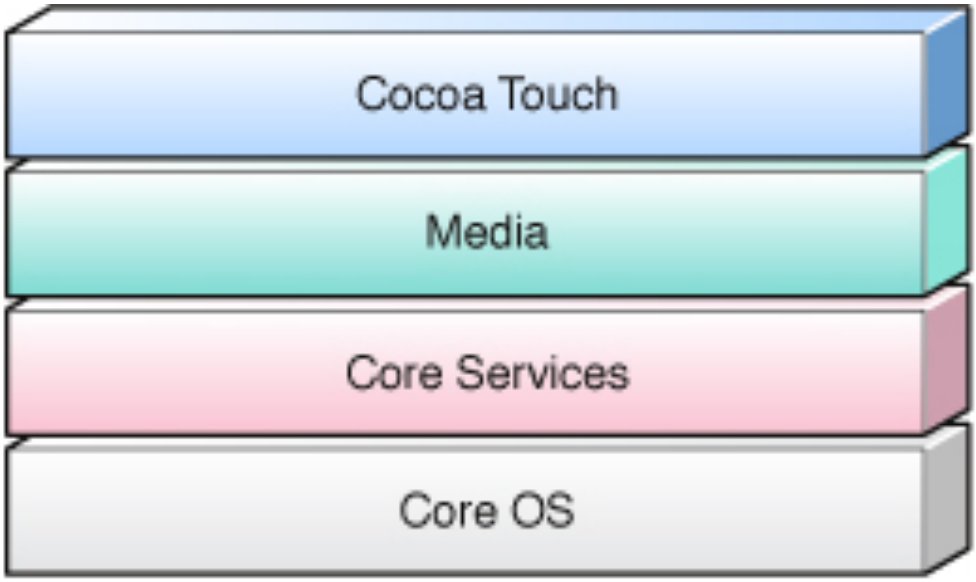

Figure 2. iOS components in layers

iOS acts as an intermediary software between the hardware and applications show on the screen of iPhone. The applications do not have contact with the hardware directly, but instead applications communicate with the hardware through a set of system interfaces that protects the applications from hardware change.

The technologies used in iOS can be viewed as a series of layers, which is shown above. The composed layers shown are:

- Cocoa Touch
- Media
- Core Service
- Core OS

The top layer Cocoa Touch contains the key framework to build iOS applications, they includes the Foundation Kit Framework and UIKit Framework. This layer is primary written in Object-C languages. The layer also support for technologies like core animation, multitasking and gesture recognizers.

Media layer is located just underneath the Cocoa Touch layer. This layer contains the media frameworks to control the graphics, audio and video technologies that are used to create multimedia on iPhone. Frameworks like AV Foundation Framework in this layer are used to manage complex sound and video playback and editing. iPhone can create complex graphics, play video and audio thanks to the frameworks in the Media layer.

Third layer Core Services provide the fundamental system services that all applications used. This layer is used to access lower-level operating services, like networking and file access. It contains frameworks like, Core Data and CFNetwork Frameworks. Even if your applications do not use theses services directly, many part of system are built upon it.

The last bottom layer is the Core OS Layer. This layer is made up by the lowest- level services in the iOS. These features include complex math, hardware accessories and cryptography. The frameworks in this layer is only used in rare circumstances, but even if you don't use the technologies directly from this layer, they are likely to be used by other frameworks.

5.3. Writing an iPhone Application (iOS).

To begin writing an iPhone application, there are certain requirements that must be meet. They are:

1. Have access to a mac computer.
2. Register as an Apple Developer to be able to download the needed iOS SDK.
3. Download the iOS SDK and Xcode tool. (Xcode comes free with new purchase of mac laptops or desktops)

After installing both of the Xcode and iOS SDK on a mac computer, you can simply start writing an iPhone application by opening Xcode and start a new project.

6. COMPARISION BETWEEN IPHONE AND ANDROID

IPhone and Android mobile have become two of the most used mobile OS in this age of time. However, they have very internal and external differences. Below are some of the differences I found most relevant to me after trying both of the mobile OS out.

6.1. Closed System vs Open System

iPhone is a closed system application, it means that the users can only use the resources Apple gives to them. Extra services, if Apple does not want to provide, users will not be able to access them. For example, Apple does not allow Adobe Flash to be used on iPhone. Also applications cannot just be installed and ran on iPhones without registration.

On the other hand, Android is an open system application. Therefore, the users are free to modify it any way they wish. Applications can be downloaded, installed and run on any Android OS, even if they are not in the Google's official Android Market.

6.2. Security

Due to the fact Android applications can be build by anyone and install onto any Android mobiles, therefore it is more likely for it to have security holes in those applications.

On the other hand, with Apple's tightly controlling the applications that are being released, it seems to contain less security holes comparing with Android.

6.3. Vender Lock In

One obviouse different in buying an Android or iPhone, is the fact that you can only buy iPhone from ether Apple directly or a registered Apple store, there are limited providers. While on the other hand, Android has more vendors selling them.

In New Zealand, Android mobiles can run in both Telecom and Vodafone, while iPhone is only available for Vodafone. This means Android phone will have more choice for their wireless carriers comparing to IPhone. This also seems to be the case for many other countries as well.

6.4. User Control

The truth is, if the user like the interface for iPhone, then Apple have great engineering design to make sure the interfaces are smooth and working all together. On the other hand, even though the interface on Android does not run as smoothly as interface on Apple, but the users have more control over how widgets and interfaces appearances.

6.5. Variations Of Mobile Devices

One clear differences between iPhone and Android mobile in turns of hardware, it the fact that there are many mobile devices out there running Android and just one kind of mobile running iOS. This will provide the user chooses on the look and internal hardware that they wishes more, while on the other hand people who want to use iOS have only one chose of mobile.

6.6. Battery Life

IPhone seems to have a longer lasting battery life to me comparing to Android mobiles. However, IPhone's battery cannot be replaced yourself at home but Andriod can. Thus the suitability of the mobile depends on the users specific needs and preferences.

7. PROS AND CONS FOR ANDROID AND IPHONE PLATFORMS

Here are a list of some of the pros and cons I found on researching the two mobile platforms.

PROS

Android OS	IPhone OS
Wider SP model availability	Consistent UI.
More carriers available	Easy OTA OS upgrade for every User.
Low to high-end mobile	Applications management is easy to control
Freedom to create services	Consistent mobile architectures
Freedom to customize UI layer	
Freedom to customize middleware	

CONS

Android OS	IPhone OS
App incompatible between different versions of Android OS.	Third-party API banned.
UI not consistence due to the freedom for customization.	Tightly controlled by Apple.
OS evolving too fast, hard to foresee problems that might occur for app compatibility.	Lack of flash.
Complex and frequent OS upgrade	

8. RESEARCH WORK

For this project I am to implement a mobile application that can detect force caused accidents and send out emergency SOS messages. The two kinds of emergencies that the application will need to be able to detect are falling and car crash accidents. Here are some research work conducted for the two kind of accidents.

8.1. Falling Accidents

The number of Fall caused accidents yearly are in fact quite significant. Falling is a major cause for personal injuries, especially for elderly people. According to the Center for Disease Control and Prevention (CDC), over one million Americans suffers from slip, trip and falling injuries every year, and out of those people 17, 000 people dies. 15 percent of all job related injuries in America are due to slip, trip or fall.

CDC categorizes falls into two types: elevated falls and same-level falls. Same-level falls are accident cause by slipping and tripping, while falling from great heights causes elevated falls accidents. Even though some-level falls are more common, but elevated falls are usually more dangerous and can cause more serious injuries. For elevated falls accidents over 60 percents are from a height of less than 3.048 meters.

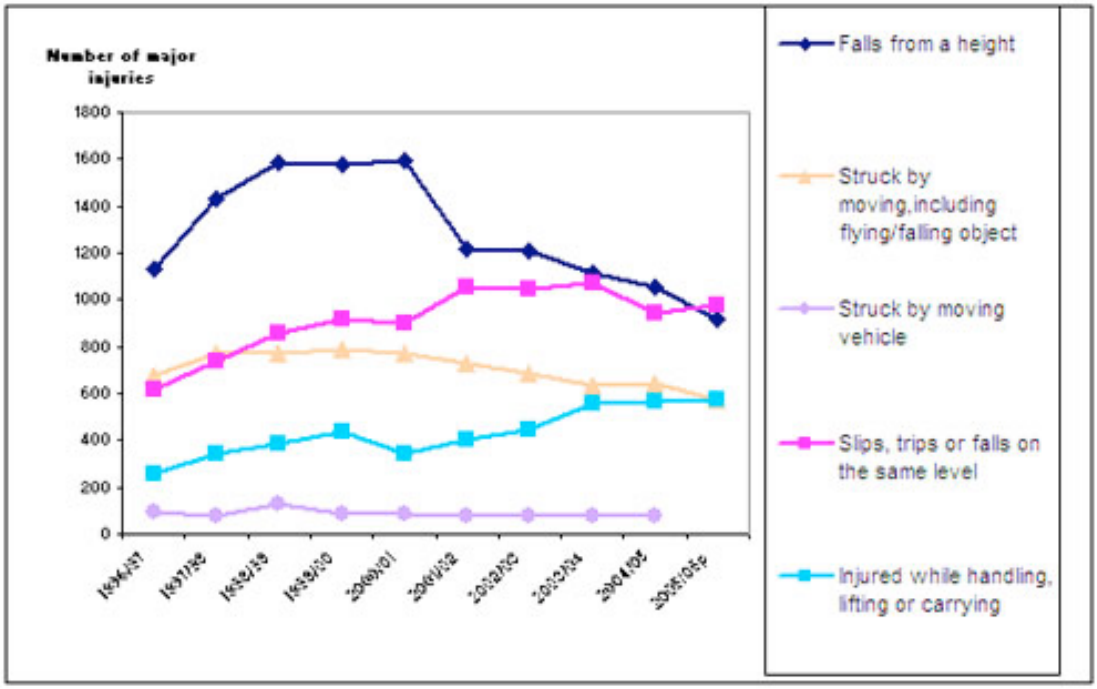

Figure 3. Causes for Fatalities for a person. [5]

The above graph at figure 2 shows the statistic for injuries caused to a person from 1987 to 2006 in United Kingdoms by range of accidents. From the graph, it is clear even though the number for “Fall from a height” injuries has decrease, but falling accidents are still the major causes for injuries.

8.2. Car Accident

Automobile accidents are one of the most common cause accidents in the world. In high population countries like Taiwan, it is the most common cause for both death and accidents. There are many causes for automobile accidents and they can usually be divided into the five categories listed below:

- Substance abuse
- Speeding
- Road layout and weather condition
- Vehicle failure
- Driver's fault

Substance abuse accidents are caused by consuming legal or illegal substances, these can include alcohol or drugs and are the major causes for car accidents.

Speeding accidents are cause by driving over the speed limit. According to the Insurance Institute for Highway Safely (IIHS) states, when speed increases from 40 mph to 60 mph, the energy released in a crash is more than double, this not only increases the risk of crashing, but also make the accident more severe.

Road layout and weather condition can affect the driver's ability to control and view the environment. Many roads have blind spots', where drivers cannot see each other.

Vehicle failure accidents are cause by mechanical breakdown in the vehicle, most commonly are due to damaged tyres, break etc. Lastly accidents can occur due to the driver's fault, where drivers mis-judge or distractions causes the accidents.

The project research focuses on the accident cause by speeding, as it can be determined by the change of speed and force that we can use the mobile to detect.

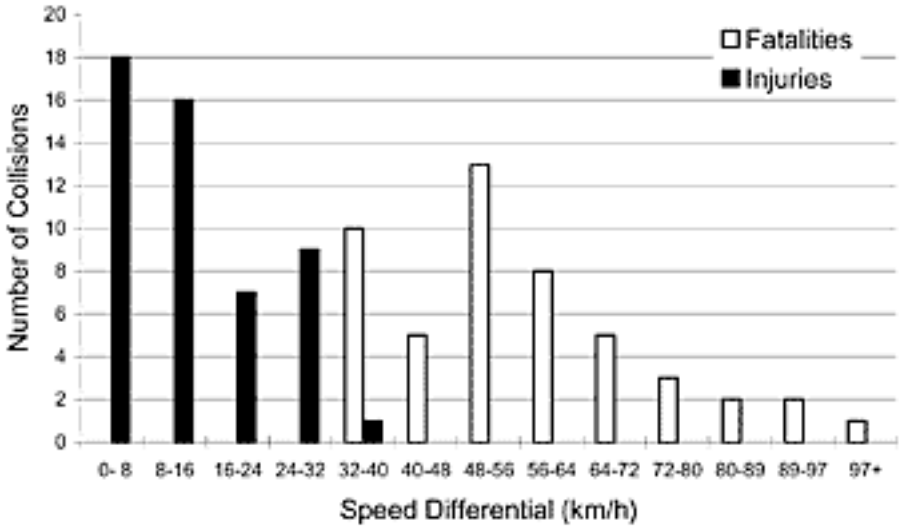

Figure 4. Number and fatalness of truck collisions according to speed. [9]

As you can see from the graph show in figure 4 above, it is clear that when an automobile is driving above the speed 32 km/h, there is an increasing chance to cause fatalities than driving under 32 km/h.

The force impacted on the driver when the car crash is dependent on the distance it took the car to stop, weight of the car and the speed.

$$F_{avg} = \frac{-\frac{1}{2}mv^2}{d}$$

Figure 5. Formula to calculate force. [8]

The formula show in figure 5 above can be used to calculate the force impacted when a car crashes, where m= the total weight for car and person, v=speed of the vehicle and d= distance it took before the car stops.

9. RESEARCH ON RELATED APPLICATIONS

ICE: Emergency contact.

This is an alert application designed for Android. It alerts contacts and rescue workers with one single click. The alert contact list is based from your own contact list in your mobile. [9]

Features:

Send to all the people in your ICE contact list
Access contact list from your contact
Get the GPS coordinate to help your contact to find you.

Emergency Distress Beacon

This is a simple application, that sends out a distress beacon with your current location to rescuers when active. [10]

Features:

Send by email rather than txt message.

Emergency QuickDial Widget

This widget for Android let you place an emergency call depending on your country and send a SMS alert to your chosen contact with your current position. Information such as: blood type, allergies are saved. [11]

Features:

Selection of countries emergency numbers
ICE information about your self.

Emergency Alert

This application, keeps track of your travel journey and if in the event of a major accident, this will send out a SMS message to a number, providing notice of the accident and your coordinates to send help. [12]

Features:

Have different sensor modes for different vehicle.

10. DESIGNS AND USER INTERFACE

Since this application is used when there is an emergency, the user interface will need to be simple and clear at the same time. The plan is to have just two different views. On one of them, the user can select the mode of transport that he/she needs monitoring; it will also display some information about the current status. The other view is used to set up detail information about your self and the emergency contact that you wish your SOS message will be sent to.

10.1 First View

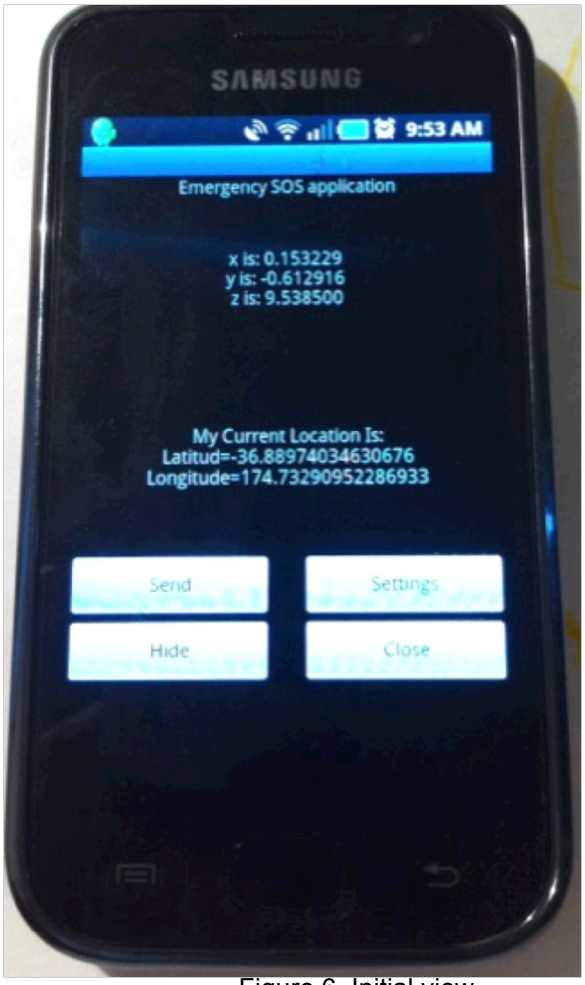

Figure 6. Initial view.

Figure 6 shows the initial view that will show up once the application has been activated from the menu. This view contains the following information:

- Title of the application
- Accelerometer information shows X, Y, Z position
- GPS position shows in longitude and latitude, statues shows in enable and disable.
- A “Send” Button
- A “Setting” Button
- A “Close” Button
- A “Hide” Button

The initial title for the application is shown on top of the first view, this is called “Emergency SOS Message”. Underneath the title it will display the information on accelerometer position in x, y, z-axis.

Information on the GPS will be display next. “GPS enable” will be displayed when the GPS is enabled for the mobile, before the position is determined. “GPS disabled” will be shown, if GPS is ether unavailable for the device or inactivated. Once the position has been found, it will be shown in longitude and latitude.

Four Buttons are located under the display of GPS information. The “Send” button will send an emergency SOS message with your information and GPS position (in longitude and latitude) to the contact that you have setup in setting. The “Setting” button will take you to the next view where you can setup your own and your emergency contact’s information. The “Close” button will end the application once clicked and the last “Hide” button will run the application in the background of the mobile. A little dialog box will pop up once the “Send” button is clicked, telling the user that it is sending the SOS message.

10.2. Setting View

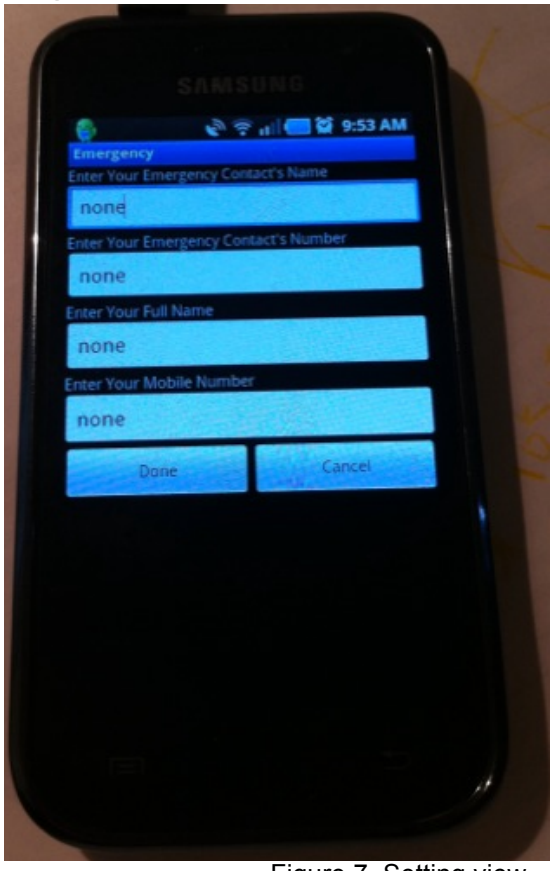

Figure 7. Setting view.

By clicking the “Setting” button in the initial view, it will bring the user to the “Setting” page. In this view it contains the information:

- A title and text editor box, asking you to enter your emergency contact’s name
- A title and text editor box, asking you to enter your emergency contact’s number
- A title and text editor box, asking you to enter your own full name
- A title and text editor box, asking you to enter your own mobile number
- A “Save” button
- A “Cancel” button

Initially, when user first enters into the “Setting” view, the text editor box will display “none” as there are no information saved in the application. Once the user enter the information accordingly, he/she can ether press the button “Save” to save the information that he/she entered or “Cancel” if he/she don't want to save the information and returns to the initial page. If there are information saved in the mobile, when the user enters the “Setting” page, instead of displaying “none” in the text editor box, it will display the current saved information in them. Just like the “Send” button in initial view, when ether the “Save” or “Cancel” button has been pressed, a short dialog will pop up telling the user what action he/she has done. The view is show in figure 7.

11. IMPLEMENTATION FOR THE APPLICATION

11.1. Setting Up the Initial and "Setting" View

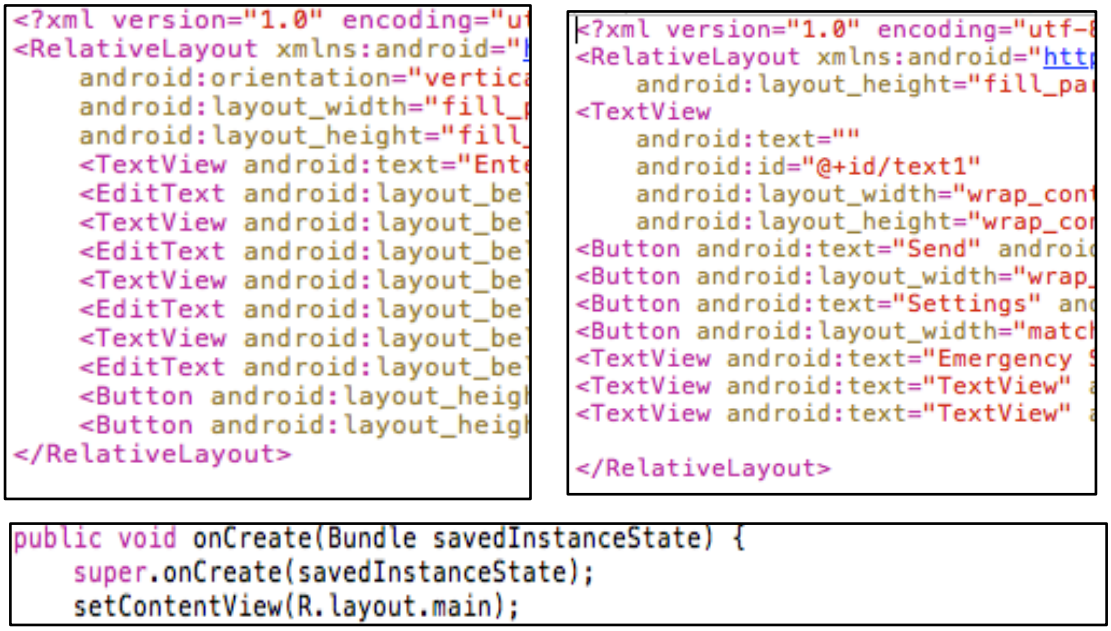

Figure 8. Top left code shows "main.xml", top right code show "setting.xml" and bottom code shows "setContentView()" method for "Project.java".

To begin the implementing of the application, I first started by creating the two views that will be used in this application. This is done by setting up two-xml file in the layer fold in the project; one is call "main.xml" and the other call "setting.xml".

Both of the views are done in "RelativeLayout" rather than "LinearLayout" as the "RelativeLayout" allows the text view or buttons to line up next to each other and "LinearLayout" only allow them to layout one after each other. In the "main.xml" initially one text view is created showing the title "Emergency SOS application" with two buttons "Setting" and "Send" layout beside each other underneath it (Other text views and buttons are added in the implementation goes on). In the "setting.xml" four-text view, four edit text view and two buttons are created, for the user to enter the information required for this application.

"Project.java" and "Project2.java" file are then created. In each of the two java file one of the previousely created xml file is assigned to it by using the "setContentView(R.layout."name of the xml");" method in each of the java files. All of the above code is show at figure 8.

11.2. Setting Up to the Button to Change Between the Two Views

```
setting.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view){
        Toast.makeText(Project.this, "Settings", Toast.LENGTH_SHORT).show
        Intent myIntent = new Intent(view.getContext(),Project2.class);
        startActivityForResult(myIntent,0);
        finish();
            }
        });
```

```
done.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v){
        Project.eName=text1.getText().toString();
        Project.eContact=text2.getText().toString();
        Project.mName=text3.getText().toString();
        Project.mContact=text4.getText().toString();
        Toast.makeText(Project2.this, "Setting Saved", Toast
        Intent myIntent = new Intent(v.getContext(),Project.
        startActivityForResult(myIntent,0);
        finish();
```

Figure 9. Top code shows the setting for “Setting” button in “Project.java” and bottom view shows the “Done” button setting for “Project2.java”.

After linking up the appropriate xml file with the proper java file, the next implementation I did is by setting the linking up between the buttons created in the xml file to their java file. In both of the java file this is done by the command “ findViewById(R.id.”name of the button”);”. After each of the buttons have been linked a action is added to each of the buttons by the “setOnClickListener()” method with the action underneath with in the method.

For the “Setting” button in “Project.java” an action, which changes to the “Setting” View in “Project2.java” is conducted. This is done by creating a Intent object with the context set as the Project2.class and the “startActivityForResult()” method, which once it has activated will take to the user to the view setup in “Project2.java.

Similar stature is also implemented for the two buttons in “Project2.java” buttons, but instead of it returns to the “Project.java”. The “Save” button in “Project2.java” will also save the values that are entered into the edit text view.

For each of the buttons in the two-java file, a short dialog will be shown once it has been clicked. This is done by the creating a toast item and set the text to be display. Edit text view in “project2.java” is also linked by using the “findViewById()” method, so that the code will detected the new information that is typed on it. Implementation of code is show at figure 9.

11.3. Sending Text Message

```
<uses-permission android:name="android.permission.SEND_SMS">
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS">
</uses-permission>
```

```
sendB.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view){
        Toast.makeText(Project.this, "Sending", Toast.LENGTH_SHORT).show();
        sendSMS(eContact,eName+"\nThis an SOS message send by "+mName+"\n"+

            }
        });
```

Figure 10. Top view shows the code in “AndroidManifest.xml” and bottom view shows the “Send” button setting for “Project.java”.

To enable the ability to send a SMS message, I have to first allow this function to sent and receive SMS by implementing permission the code "android.permission.SEND_SMS" and "android.permission.RECEIVE_SMS" in the “AndroidManifest.xml” file.

After setting up the permission, in the “Project.java” file, a method called “sendSMS” is created. In this method, it takes in two string objects, one for the phone number and the other for the messages to be send, a SmsManager item is also created and with the combination of “sendTextMessage()” method, the user can now send SMS by calling this method. This newly created method is passed in underneath the actions for button “Send” with the emergency’s contact number and SOS message as the two strings. The implementation of the code is show in figure 10.

11.4. Setting Up and Display GPS information

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION">
</uses-permission>
```

```
LocationManager mlocManager = (LocationManager)getSystemService(Cont
LocationListener mlocListener = new MyLocationListener();
mlocManager.requestLocationUpdates( LocationManager.GPS_PROVIDER, 0,
```

```
public class MyLocationListener implements LocationListener {
    public void onLocationChanged(Location loc){
        loc.getLatitude();
        loc.getLongitude();
        locations="My Current Location Is:"+"\nLatitud="+loc.getLatitude()
        txtPerson.setText(locations);

    }
    public void onProviderDisabled(String provider){
        //Toast.makeText(Project.this, "Gps Disabled", Toast.LENGTH_SHORT
        txtPerson.setText("GPS DISABLED");
    }
    public void onProviderEnabled(String provider){
        //Toast.makeText(Project.this, "Gps enabled", Toast.LENGTH_SHORT )
        txtPerson.setText("GPS ENABLED");
```

Figure 11. Top view shows the code in “AndroidManifest.xml”, middle view shows the initialization of “LocationManager” and “LocationListner” item in “Project.java” and bottom views shows the class “MyLocationListener” created in “Project.java”.

To enable the ability to read GPS, I also have to setup the permission in “AndroidManifest.xml” file by using the code "android.permission.ACCESS_FINE_LOCATION".

In the “Project.java” file, a “LocationManager” and a “LocationListener” item is created. “LocationManager” item is use to get the location updates, while “LocationListener” item listen to the change in location and pass it to “LocationManager” item. A new class “MyLocationListener” implements “LocationListener” calss is also created in the “Project.java” file. This class contains the methods “onLocationChange()”, “onProviderDisable()” and “onProviderEnable()” to detect the statues of GPS and if there is a location change or not. If there is a location change, the new location in latitude and longitude will be saved in to a String variable and passed into end of the sent SMS message. Implementation of the code is shown in figure 11 above.

A new text view is also added into the “main.xml” file and linked to “Project.java” to show the changing information for GPS.

11.5. Setting Up and Display Accelerometer information

```
SensorManager sensorManager;
Sensor sensor;
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensor = sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER).get(0);
result = (TextView) findViewById(R.id.textView2);
result.setText("No result yet");
```

```
private SensorEventListener accelerationListener = new SensorEventListener() {
    @Override
    public void onAccuracyChanged(Sensor sensor, int acc) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        x = event.values[0];
        y = event.values[1];
        z = event.values[2];
        refreshDisplay();
    }
};
```

```
private void refreshDisplay() {
    String output = String
            .format("x is: %f \ny is: %f \nz is: %f", x, y, z);
    result.setText(output);
}
```

Figure 12. Top view shows the initialization of “SensorManager” and “Sensor” item in “Project.java” , middle view shows class “SensorEventListener” created in “Project.java” and “bottom views shows the Method “refreshDisplay()” created in “Project.java”.

In the “Project.java” file, a “SensorManager” and a “Sensor” item is created. “SensorManager” item is use to get the accelerometer updates, while “Sensor” item is use to listen for accelerometer change. A “SensorEventListener” class with methods is created to override the initial method in the “SensorEventListener” class. A “onSensorChanged()” method is used to sense the change in x, y and z-axis for the accelerometer and store the value in float variable “x”, “y” and “z”. A method called “refreshDisply()” is also created to display the accelerometer’s position on the display. The implementation is show in figure 12 above.

A new text view is created in “main.xml” called “result”, which is linked to the “Project.java” and is used to display the accelerometer’s information on screen.

12. COMPLICATIONS

Some of the problems that I encountered during my implementations are:

1. Information for personal and emergency contact need to be reentered when application is shut down and restarted.
2. By enabling GPS, the application takes a large consumption of power from battery.
3. User interface is overly simple and not attractive at all.
4. Accelerometer's setting is too sensitive, adjustment needed.

13. FUTURE WORK

Some of the future works that will be conducted in the next a couple of weeks are:

1. Design and implement algorithm used to sense the collisions.
2. Better and clearer user interface
3. Notification in the top bar to display information when application is hidden in the background.
4. Ways to wake up the mobile from sleep mode when collision is detected.
5. Use GPS to calculate the speed.
6. Start implementation the application on iOS.

14. CONCLUSTION

Through out this first semester I have been working on creating a SOS emergency application for Android system mobile. This provided me an understanding on how Android's operating system works and knowledge on how to implement Android application. The Android SOS emergency application that I have implemented up to now uses the function GPS and accelerometer in the mobile to sense for collisions.

For the next few weeks, I will be working on solving the issues that I currently have in my Android application. For example, creating better user interface and collision algorithm. Also I will be starting on implementing on iOS for iPhone application.

15. REFRENCES

[1]. Android Developers. Retrieved from: http://developer.android.com/guide/basics/what-is-android.html

[2]. Develop for iOS. Retrieved from: http://developer.apple.com/technologies/ios/

[3]. Steven J.Vaughan-Nichols. (29/06/2010). iPhone vs. Android: Five Points of difference. Retrieved from:http://blogs.computerworld.com/16437/iphone_vs_android_five_points_of_difference

[4]. Slip and Fall Accidents Stats. Retrieved from: http://www.lawfirms.com/resources/personal-injury/slip-and-fall-accident/slip-and-fall-accidents.htm

[5]. (07/11/2006). Construction statistics 2005/06(p) – falls down, slips up. Retrieved from: http://www.hse.gov.uk/press/2006/e06109.htm

[6]. Diana Joseph. 5 Most Common Causes of Accidents. Retrieved from: http://ezinearticles.com/?5-Most-Common-Causes-of-Accidents&id=1466805

[7]. Killer Parked Trucks. Retrieved from: http://www.underridenetwork.org/CurrentIssues/TruckParking.aspx

[8]. Forces in Car Crashes. Retrieved from: http://hyperphysics.phy-astr.gsu.edu/hbase/carcr.html

[9]. (30/05/2011). ICE: Emergency Contact. Retrieved from: https://market.android.com/details?id=com.alexyu.android.ice&feature=search_result

[10]. Ian Cinnamon. (25/08/2008). Emergency Distress Beacon. Retrieved from: http://itunes.apple.com/us/app/emergency-distress-beacon/id288770664?mt=8

[11]. Emergency QuickDial Widget. Retrieved from: http://www.androlib.com/android.application.com-aportela-emergency-tECm.aspx

[12]. (14/10/2009). Emergency Alert (MarketVer). Retrieved from: https://market.android.com/details?id=org.firezenk.emergencyalert&feature=related_apps