



## SOS – AN ANDROID APPLICATION FOR EMERGENCIES

By

Nalini IAP19110010409

Vivek K IAP19110010413

Vamsi Krishna J IAP19110010366

Srilekha J IAP19110010397

SRM UNIVERSITY, Amaravati, AP.

A REPORT

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF TECHNOLOGY

Department of Computer Sciences and Engineering

College of Engineering

Approved by:

Assistant Professor

Dr. Sriramulu Bojjagani

## **Abstract**

The project's goal is to create an Android app that allows users to send notifications in the event of an emergency or a panic situation. Users can send multiple text messages with the press of a single button. The phone numbers and text content can be customized from within the application. Along with the content, the text messages sent include the user's last known location. This is extremely useful in tracking down the person's whereabouts. If the nature of the situation requires it, the user can also dial 100/102/1091 directly from within the application.

In addition, the app's user may grant the app permission to track their location. Whether this option is chosen, the application retrieves the device's location every 15 minutes and stores it in a database. This information is extremely advantageous and can be applied in a wide variety of ways. One such application of location data is from within the Android app, where the user can view a map that shows their location history over time for a specific day.

## Table of Contents

Abstract.....	2
Acknowledgements .....	5
Chapter 1 - Introduction.....	5
1.1 Motivation.....	6
1.2 Project Description.....	6
Chapter 2 -Literature view.....	7
2.1Domain research.....	7
2.2 Need for mobile technology .....	7
2.3 Need for Android technology .....	8
2.4GSM, GPRS,WIFI.....	8
2.5Visibility.....	8
2.6 Feedback and constraints .....	8
Chapter 3 -Background.....	8
3.1Android.....	8
3.2Android architecture.....	9
Chapter 4 - Requirement Analysis.....	10
4.1 Requirement Gathering.....	10
4.2 Requirement Specifications.....	11
4.3 Software Requirements.....	11
4.4 Hardware Requirements .....	12
Chapter 5 - Architecture & Design .....	12
5.1 System Architecture .....	12
5.2 Design Diagrams.....	13
Chapter 6 - Android Framework Components.....	15
6.1 AndroidManifest.xml .....	15
6.2 Activities.....	18
6.3 Intents .....	18

Chapter 7 - Implementation.....	19
7.1 Graphical User Interface.....	20
7.1.1 Login .....	20
7.1.2 Register.....	21
7.1.3 Reset password.....	22
7.1.4 Main screen.....	22
7.1.5 Change password.....	24
7.1.6 Set SMS messgae.....	24
7.1.7 Location.....	25
Chapter 8 Testing .....	26
8.1 Unit Testing.....	27
8.1.1 Login Screen test cases.....	27
8.1.2 Register screen test cases.....	27
8.1.3 Main screen test cases.....	28
8.2 Integration testing .....	29
8.3 Performance testing.....	32
Chapter 9 Future Work .....	33
Chapter 10 Conclusion .....	33
Chapter11- Bibliography.....	34

## List of Figures

Figure 3-1 Android Architecture .....	9
Figure 3-2 System Architecture diagram.....	11
Figure 4-2.1 Use case diagram - 1 .....	13
Figure 4-2.2 Use case diagram - 2 .....	14
Figure 4-2.3 Use case diagram – 3 .....	15
Figure 6-1 Activity lifecycle.....	18

Figure 7.1.1 Login Screen .....	20
Figure 7.1.2 Register screen .....	21
Figure 7.1.3 Password reset screen.....	22
Figure 7.1.4 Main screen.....	23
Figure 7.1.5 Change password screen.....	24
Figure 7.1.6 SMS message screen.....	25
Figure 7.1.7 Location .....	26

## List of Tables

Table 6-1 Lines of Code (LOC).....	20
Table 7-1 Unit test cases - 1.....	28
Table 7-2 Unit test cases - 2.....	29
Table 7-3 Unit test cases - 3.....	30

## Acknowledgements

This project would not have been possible without the support and guidance of my Sriramulu Bojjagani . I would like to extend my sincere gratitude to him for trusting in my abilities and providing me with an opportunity to work with him. He has been a source of immense knowledge, encouragement and provoked me to think innovatively.

Finally, I would like to thank my family and friends for their endless support and motivation.

## Chapter 1 - Introduction

The system's task is defined by its name: "SOS"(save our souls) denotes a system that will handle emergency situations and provide home remedies for various ailments. Anyone in a difficult situation, from home users to business users, who requires assistance or guidance, whether stranded in an emergency situation or looking for home remedies for common diseases. In the event of an emergency, people who want to send an emergency message. In an emergency, a person who wants to find the nearest hospital, police station, and taxis. What are the specifications for this application? Consider the following points to gain a better understanding of the problem and the need for this application: I was unable to contact anyone during an emergency. I am unable to reach the nearest hospital in the event of an emergency. You are unable

to locate a hospital or police station in the event of an emergency. Some critical locations and routes were unable to be saved. You won't be able to send multiple texts in an emergency. Inability to find home remedies for minor ailments.

## **Motivation**

As much as we would like to get rid of them, panic or emergency situations are unavoidable and usually unexpected. The nature and consequences of these situations can vary significantly and in worst cases also be life threatening. Therefore it would be really nice to have some mechanism by which we can notify certain people about such circumstances and increase the chances of receiving help as soon as possible.

The need for such a mechanism increases even more as in this era of technology, platforms exist to support them. One such platform and a very common one is a Smartphone. Almost everyone today carries a Smartphone with them as they become more and more affordable and easily available. In terms of market share, Android is also the undisputed leader in the Smartphone market. According to one report, Android-powered smartphones accounted for 78.1 percent of all Smartphones sold in 2013. As a result, creating an Android application is an obvious choice.

## **Project Description**

SOS is an application designed for Android devices, primarily smartphones, but also tablets that support Cellular Service. The application's primary functions and features are as follows:

- i. The first time a user opens the app on his device, he must login by entering a username and password. He then remains logged into the application until he explicitly logs out.
- ii. If the user does not have an account, he can register on the login screen.
- iii. The user can also choose the password reset option in case he does not remember his password. A new password is set for the user and a mail containing this new password is sent to the registered email id.
- iv. Once logged in, the user is directed to the main screen of the application. This is the screen that would open up when the user opens the application. The user can press the panic button to send text messages and emails to the contacts set up, he can also send an 'I am OK' message to these contacts by clicking on the OK button. The user can also call 100/102/1091 directly from within the application by pressing the police/ambulance/women's helpline button. In order to avoid unnecessary and accidental

press of these buttons, the user has the option to enable and disable these buttons.

- v. The user will also see his current location on the map screen. This way he would know his exact location and refer to it in case he makes a call to 100. This location is also sent as a part of the text..
- vi. The user can set the contacts to send the text message within the app. He can either select the contact from the contact book or can enter one manually. He can also set the text message that would be sent.
- vii. The user can enable the option to start location tracking. If this option is selected, the application fetches the location of the device (about every 15 minutes) and stores it in an external database.
- viii. If the permission to track the location was granted, he can see the latitude and longitude of your present location.

## **2. Literature View**

### **2.1 Domain Research**

As this project is a website as well as an android application. The development of this needs great research work. With research work done properly one can make out the success or failure of the project, as it provides complete exposure of knowledge, business, human networking, better insights and understanding of the required area. Domain research will deal with the whole method of a Website building. Then the developer will include the study of XML, Android, SQL Queries. To make the research better, developer has divided it in the following domains:

### **2.2 Need for Mobile Technology**

At the end of 2011, there were 6 billion mobile subscriptions, estimates The International Telecommunication Union (2011). That is equivalent to 87 percent of the world population. And is a huge increase from 5.4 billion in 2010 and 4.7 billion mobile subscriptions in 2009. From the given facts one can easily understand why we should develop a mobile application. Also in the problem description area I have mentioned many reasons why we should not opt for a file system. This is the main reason behind choosing mobile technology.

### **2.3 Need for Android Mobile Technology**

48.5% of people in the US own an android device. Out of 51.5 all other mobile companies are fighting for their existence. At the 2012 Mobile World Congress, Google released a startling statistic: There are now around 850,000 Android activations made each day.

When close to one million phones are being turned on every day, you're doing something right. Yet, Google and its mobile operating system, Android, still get a bad rap because of different versions coming out. Still it is going to play a vital role in the future mobile era. This is why we have chosen android technology to implement this system.

## **2.4 GSM, GPRS, WIFI**

The proposed system will provide the services across the countries also as the users often continue to use their mobile phones when they travel to other countries. The proposed system can access the internet by GPRS and the data usage must be less than 56 Kbps so that all users can effectively use the services. The proposed system can access the internet by Wi-Fi and the data usage must be less than 80 Kbps so that all users can effectively use the services

## **2.5 Visibility**

Visibility is one of the most important design principles and what it means is that, as and when the user looks on the system screen he/she may feel the possibility for action. The developer will keep an eye on this principle in order to provide better visibility.

## **2.6 Feedback and Constraints**

Feedback is the response to the user of the action performed Constraints are some universally accepted conventions which notify some specific actions.

# **Chapter 3- Background**

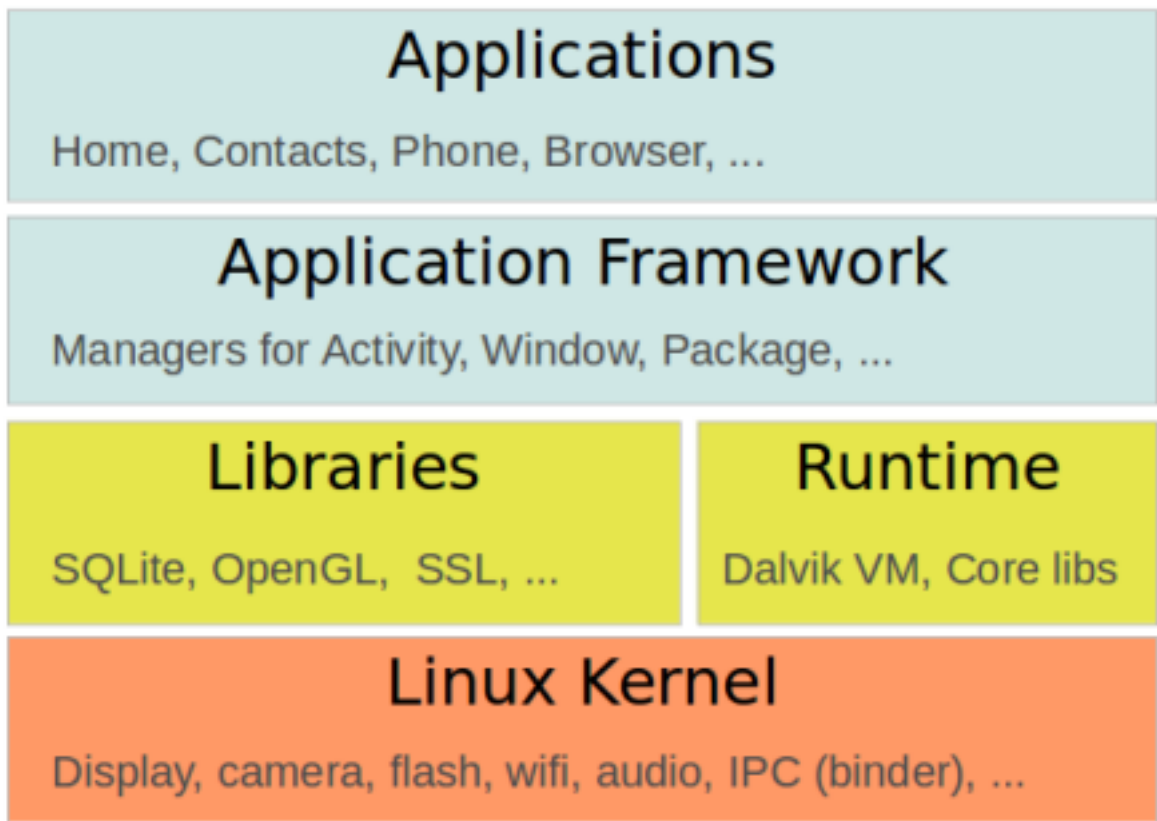
## **Android**

Android is one of the most widely used Mobile Operating System today. Android OS specific programming, as it is better for customization Android is well supported with JSP Android Support with Google-Map and GPS technology Android has Multi-threading, Handler, Exception Handling, Encapsulation which can be used in the system for various functions like fetching friends data while showing their location on Google-Map.

## ***Android Architecture***

The Android operating system is implemented as a stack of different layers of software. The following image depicts these different layers:





**Android Architecture [2]**

**Linux Kernel** – This is the layer at the very bottom of the Android architecture. All other layers run on top of the Linux kernel and rely on this kernel to interact with the hardware. This layer contains all the essential hardware drivers which help to control and communicate with the hardware. It provides the basic functionality like Process Management, Memory Management and Device Management like Camera, Display, Flash etc.

**Libraries** – This is a set of common functions of the application framework that enables the device to handle different types of data. Some of the most important sets of libraries that are included are – Web kit which is the browser engine to display HTML, OpenGL used to render 2-D or 3-D graphics onto the screen, SQLite which is a useful repository for storing and sharing of application data.

**Android Runtime** – The Android runtime mainly consists of the Dalvik Virtual Machine (DVM). DVM is very much like the standard Java Virtual Machine (JVM) except that it is optimized for mobile devices that have low processing power and low memory. DVM generates a .dex file from the .class file at compile time and provides higher efficiency in low resources devices. Each application has its own process and an instance of DVM. Android runtime also provides core libraries that enable the Android developers to create applications using the Java language.

**Application Framework-** These are some standard class files that are available to the developer for use. An application can directly interact with them and make use of them. The application framework provides the most basic functionality of the phone like Location Manager, Content Providers etc.

**Applications** – This is the topmost layer in the architecture and the layer where the application that we develop fits in. This layer provides several pre-installed applications that are default for certain things like Contacts Books, Browser etc.

## **Chapter 4 - Requirement Analysis**

### **Requirements Gathering**

Requirements gathering is one of the most important phases of a software development life cycle. It is the phase that tells us what the system is supposed to do and drives the other phases in the life cycle.

Requirement gathering for the SOS app started with brainstorming and discussion with other students as to what features are the most essential in a panic situation. This led to the most basic and initial draft of requirements for the application. Requirements were also collected by looking at other devices like personal locator beacons and satellite messengers that are commercially available. A brief study of the functionality of the devices helped me to refine and narrow down the requirements even further. One important thing to learn for these devices was the simplicity of their design. This helped me to design an effective and simple UI design for my application. The next step for requirements understanding was to look for existing solutions and similar applications in the Android market. A careful study of these applications, adding other important features and removing unnecessary features was done.

I met with my major professor Dr. Daniel A. Andresen regularly and he helped me to refine the requirements and user interface even further to set a clear set of functional requirements for the application.

The major functional requirements for the SOS app are –

- i. The user of the application should be asked to log in only the first time he uses the application on his device. The user must see the main page of the app (with the buttons to send notifications) for every other time he opens the app.
- ii. The user shall be able to send notifications with the tap of a single button. Separate buttons should be available for sending panic messages, I am OK and making a call to 911.
- iii. An option must be provided to enable/disable these buttons to avoid pressing them by accident.

- iv. The user shall be able to see their current location.
- v. The user shall be able to set the contacts to send text messages from within the application.  
The user must also be able to set the contents of the messages. Also the user may select these contacts from the contact book or enter them manually.
- vi. They must also be able to see their location history from within the application.

Other non-functional requirements for the application are –

- i. Providing a simple and elegant UI for the main screen. This is necessary as the user would usually come on to this screen in case of a panic or emergency and hence each button should be clearly visible and easily pressed.
- ii. In case the option to track location is selected and there is no internet connectivity on the device (both wireless and Cellular data), the application should be able to store the locations offline and send them to be stored in the database once the internet connectivity is up again.
- iii. Providing a tab based view to display the different settings for the application and location history for the user.
- iv. Enabling swipe gestures for the tabbed view.
- v. Displaying user friendly dialogs for picking the date, time, entering the contacts to send text and email messages to and to enter the contents of the text and email messages.

## **Requirement Specifications**

### *Software Requirements*

These requirements are separated based on whether you are developing the app or running the app on a device.

#### **For development:**

Operating System: Windows XP or higher/Mac OS X 15.8 or later/Linux

Platform: Android SDK Framework 10 or higher

Tools: Eclipse SDK 3.5, ADT plug-in for eclipse

Technologies used: Java, SQLite, Android, Google maps v2 API

Debugger: Android Dalvik Debug Monitor Service (DDMS)

Android Emulator: API level 14 or higher

#### **For running on a device:**

Operating System: Android 3.0 or higher

Cellular capabilities for SMS messages

## ***Hardware Requirements***

### **For development:**

Processor: Intel Pentium IV or higher

RAM:256MB

Space on disk: 250 MB (at the least)

### **For running on a device:**

Device: Phone or tablet running Android 3.0 or

higher Disk space: 6 MB (at the least).

## **Chapter 5- Architecture & Design**

### **System Architecture**

The different components in the architecture are –

- i. User – This is the person who installs the application on his Android device. The user provides various inputs like username, password, and contact numbers etc. and triggers various events on the application.
- ii. Front End – This is the part of the application that is visible to the user. A screen presented to the user is usually an Activity, Fragment or a Dialog Box. They contain various elements like text box or buttons to take inputs from and provide outputs to the user.
- iii. Logic – These are the java files that contain the logic of the application. They contain various methods and classes that meet the functional requirements of the application. These files also contain code to communicate with other components in the application.

For example, a file called Map.java will make use of android .gms.location request to connect the Android app with Google Maps Engine to render location.

- iv. Services – This is the component of the application that is typically used to perform long background tasks that do not have a user interface. For example – a service is used to track the location of the device at every fixed interval of time.
- v. Receivers – This is the component of the application that typically listens for some events or responses from other services. For example – A receiver is used to fetch the location co-ordinates from the location service and then add this location to the database for future references.
- vii. Location Manager – It is used to fetch the location of the Android device. The app uses both the GPS provider and the network provider to find the location for the device. GPS

provides more accurate data about the location but usually takes sufficient time to start up after the connection is relinquished. Network provider on the other hand are quicker but the accuracy is lesser than GPS.

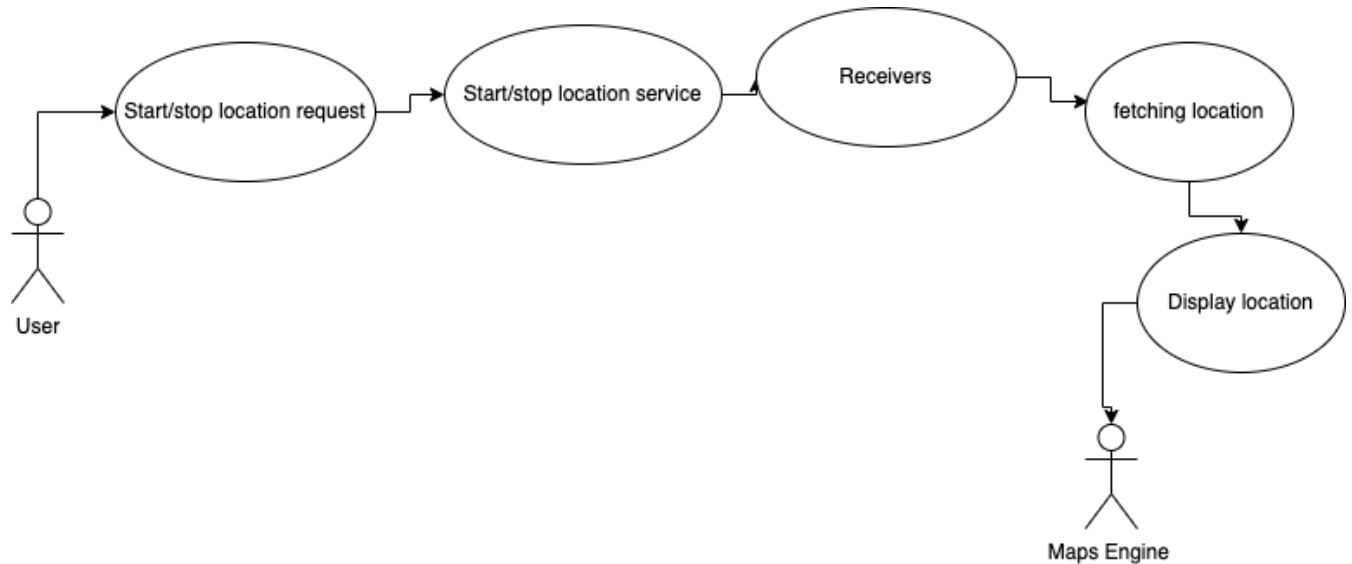
- viii. **Contacts Manager** – A system service that provides the contact to use so that the user can select a contact that is already present in the contact book. When the user clicks on the number text box to enter a number it opens up the contact book application. If the user selects a contact and if that contact has a number associated, it is sent to the SOS application and displayed in these text boxes.

## Design Diagrams

### Use case diagrams

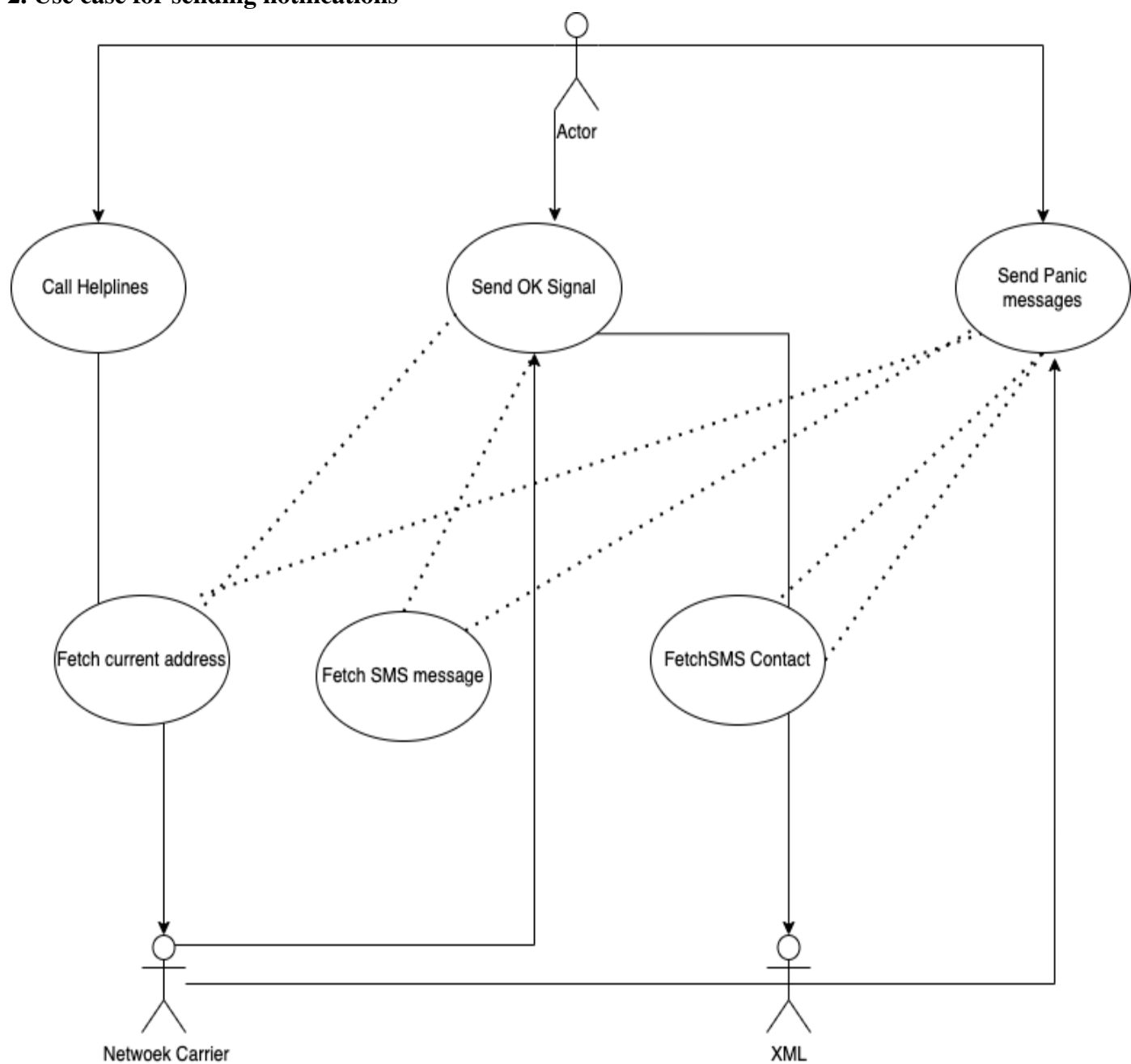
A use case diagram is used to specify the functionality of the system from the point of view of a user. Each use case describes a logical task that may be performed by a user. It mainly shows the interaction between the system and the outside world.

#### 1. Use case for location tracking and fetching location history



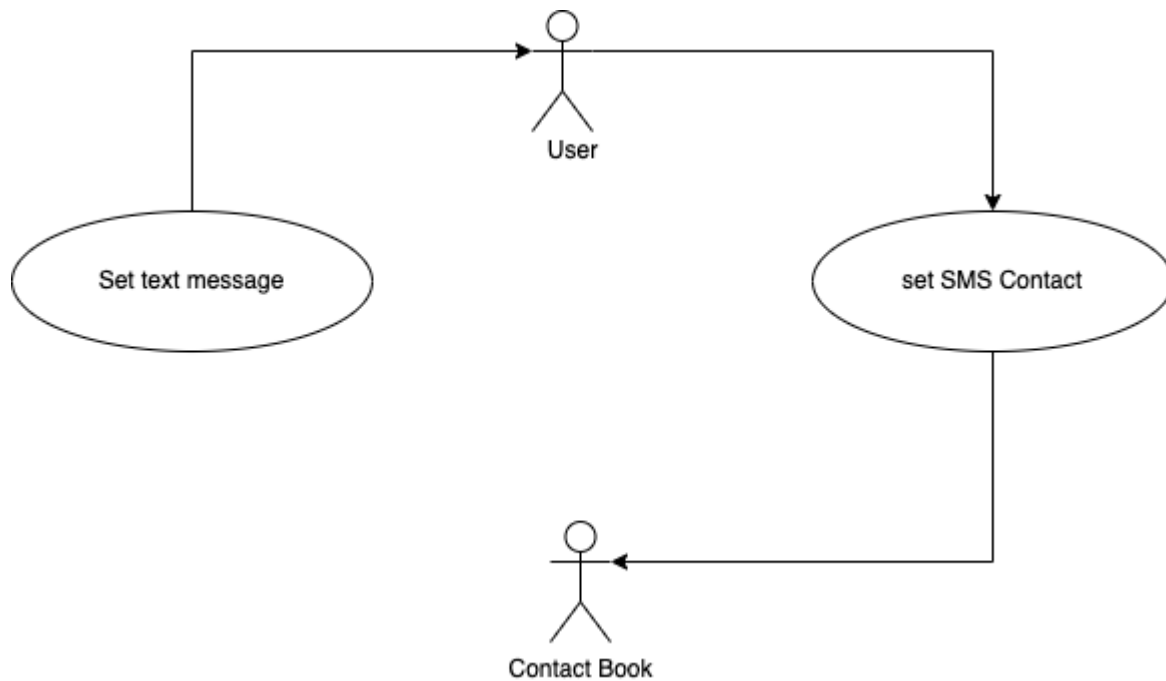
Use case diagram - 1

## 2. Use case for sending notifications



Use case diagram - 2

### 3. Use case for setting contacts



Use case diagram – 3

## Chapter 6 - Android Framework Components

Android applications are written in Java. There are different integrated environments (IDEs) that can be used to develop Android apps. SOS app is developed in eclipse using Android software development kit (SDK). SDK tools create an Android package (.apk) that contain all the necessary resources to install and run the app. Each app runs as a separate process in the underlying Linux kernel and behaves like a separate user. Files within an app can be run only by the specific user id assigned to the app. Each app also has its own instance of the Dalvik Virtual Machine (DVM). In order for the apps to share data with other apps like system services we have to assign permissions to the app during install time. This is done by adding the required permissions in the Manifest file.

### AndroidManifest.xml

The AndroidManifest.xml (Manifest) file provides important information to the Android system to run the app. All the components have to be declared in the Manifest file for the Android system to be able to instantiate them. The Manifest file also contains the various permissions needed by the application, API libraries that the app is linked with like Google maps Android API v2, other hardware and software features that the app uses and also the minimum API Level supported by the app. The Manifest file for the SOS app is as below:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shuraksha">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.SEND_SMS" />
    <uses-permission android:name="android.permission.RECEIVE_SMS" />
    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Shuraksha"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Shuraksha">
        <activity
            android:name=".helpline"
            android:exported="false" />
        <activity
            android:name=".maps_page_2"
            android:exported="false" />
        <activity
            android:name=".MapsPage"
            android:exported="false" />
        <activity

```



```

        android:name=".criminate"
        android:exported="false" />
    <activity
        android:name=".messages"
        android:exported="false" />
    <activity android:name=".register" />
    <activity android:name=".homeact" />
    <activity android:name=".login">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <meta-data
        android:name="com.google.android.gms.version"
        android:value="12451000" />
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyBhm_51C8X0rulfCY10sFaqDo5NFTyDi0Q" />
</application>

</manifest>

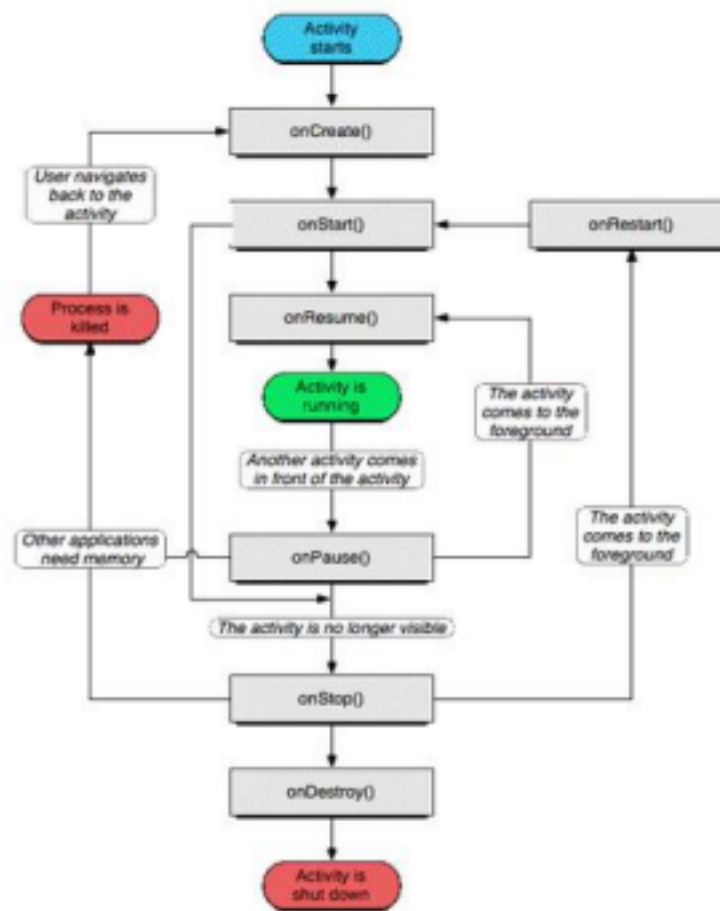
```

The minimum SDK version required for the app is 14 which corresponds with Android 4.0 (Ice cream sandwich). The reason for this is that the user interface of the application uses fragmentation and google maps which are better supported in Ice cream sandwich and above. The Manifest file also declares the various activities like Login, Register, Registered, Main, Settings, Reset password, Change password and Map that are used in the SOS app. Other components declared in the Manifest file are the services and the receivers used in the SOS app. The Manifest file also declares the Intent filters for some receivers and activities. Like the “Android.Intent.action.MAIN” and “Android.Intent.category.LAUNCHER” filters specify an Activity to be the main Activity that starts up when the app icon is clicked. Various permissions like Internet, read contacts, send SMS, call phone, Wake lock, Write external storage are provided in this Manifest file.

## Activities

An Activity is the component of an Android app that is presented to the user and

responsible for interacting with them [4]. The Activity may cover the entire screen of the device or may only cover a part of the screen displaying on top of another Activity. An Android app is a collection of loosely coupled activities along with other resources where one Activity can call another Activity at any point usually using Intents. When an Activity starts another Activity the previous Activity is pushed on to a back stack. The back stack is a Last in – First out (LIFO) structure maintained by the Android system for every app. When the user starts a new Activity it is pushed on the top of the stack and displayed to the user, when the user is done with the Activity and presses the back button, this Activity is popped from the stack and the user sees the previous Activity. An Activity implements a number of callback methods that are invoked by different events during the lifecycle of an Activity. The following diagram shows the various callback methods for an Activity:



**Activity lifecycle[5]**

## Intents

Intents are objects that are used to exchange messages between different app components[6]. They are typically used for the following purposes:

- i. To start an Activity by calling the `startActivity()` method, if the calling Activity expects a result from the Activity being called the Activity should be started with

startActivityForResult() method.

- ii. To start a service by calling the startService() method. Services are typically used to perform long background tasks that do not require a front end.
- iii. To deliver a broadcast message to various components within the same or different app that have the corresponding Intent filter declared.

An Intent can be of two types:

- i. Explicit – These are Intents that specify the name of the app component to call. Such Intents are typically used to call components within your app. The Android system finds the component with the specified name and immediately starts passing it any additional information that may have been provided in the Intent.

For example –

```
Intent i = new Intent(context, Map.class);
i.putExtra("key", json_string);
startActivity(i);
```

The above Intent is for the Map Activity in the app. It also contains extra data with key “key” and value “json\_string”. When the startActivity() method is called, the Map Activity is started along with the extra information.

- ii. Implicit – These Intents do not specify the name of a component but rather contain an action that they would like to be performed. The Android system then finds a component that can perform the specified action from other apps by matching the action against the Intent Filter for the components.

## Chapter 7 - Implementation

The SOS app is a collection of Activities and Fragments that are presented to the user. These Activities and Fragments have associated XML files (Layouts) declared in the layout folder which determine the graphical interface for these components. The SOS app also contains other Service and Broadcast Receivers along with the declarations and necessary permissions in the Manifest file in the root directory of the project.

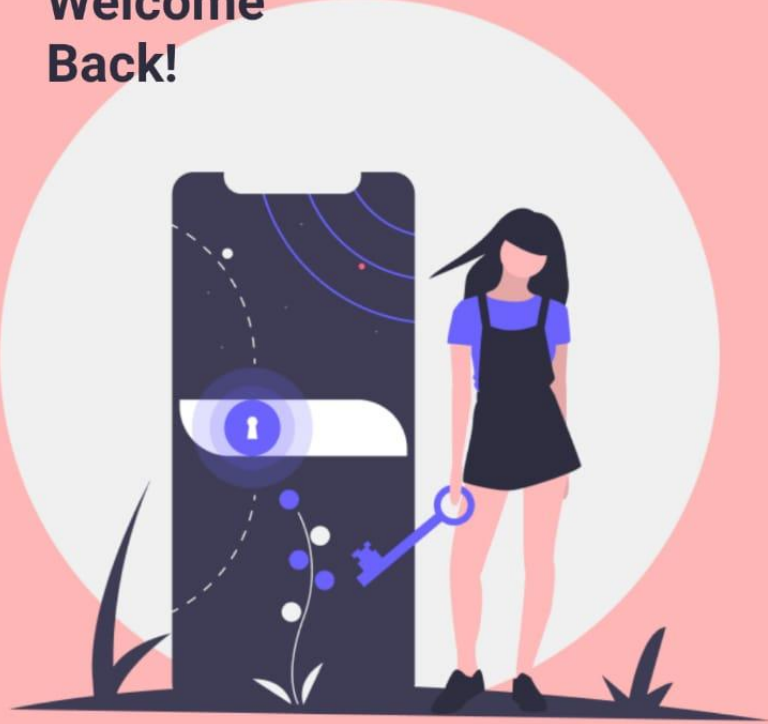
Language	Lines
JAVA	1388
XML	1460

**Lines of Code (LOC)**

**Graphical User Interface**

## *Login*

**Welcome  
Back!**



Email

Password

[Forgot Password?](#)

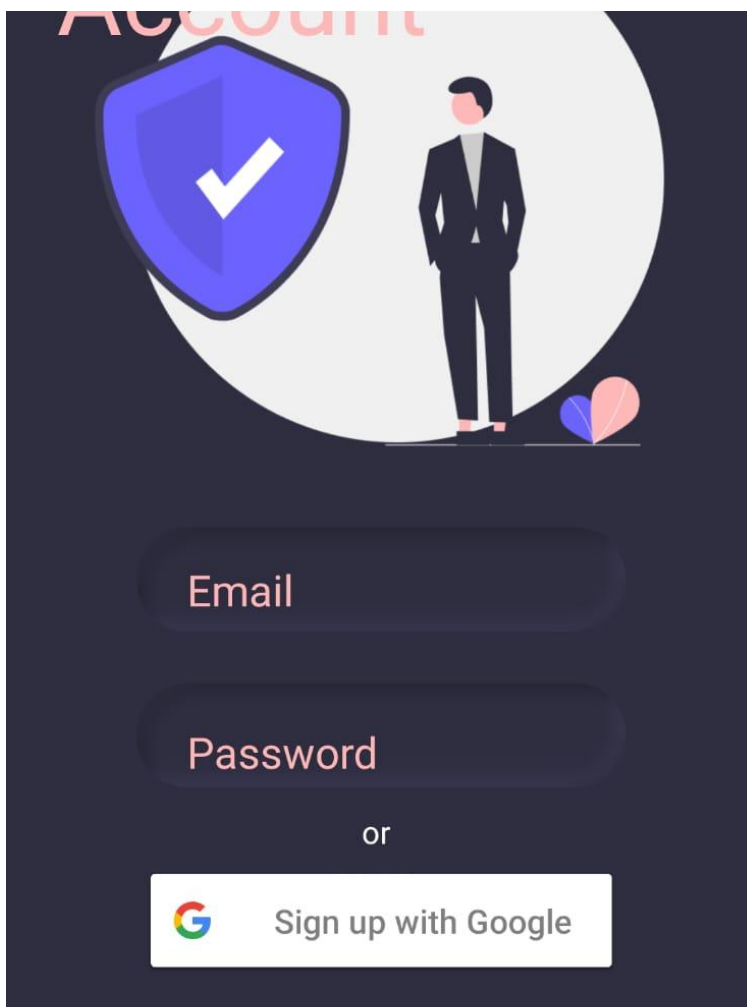
**SIGNUP** **LOGIN**

**Login Screen**

The user should log in to the SOS app using the above interface. If the user is registered,

he should enter the email id and password to log in. The user can also click on the Register button to register for the SOS app. The user may also click on the Forgot password if he does not remember his password and wants to reset it.

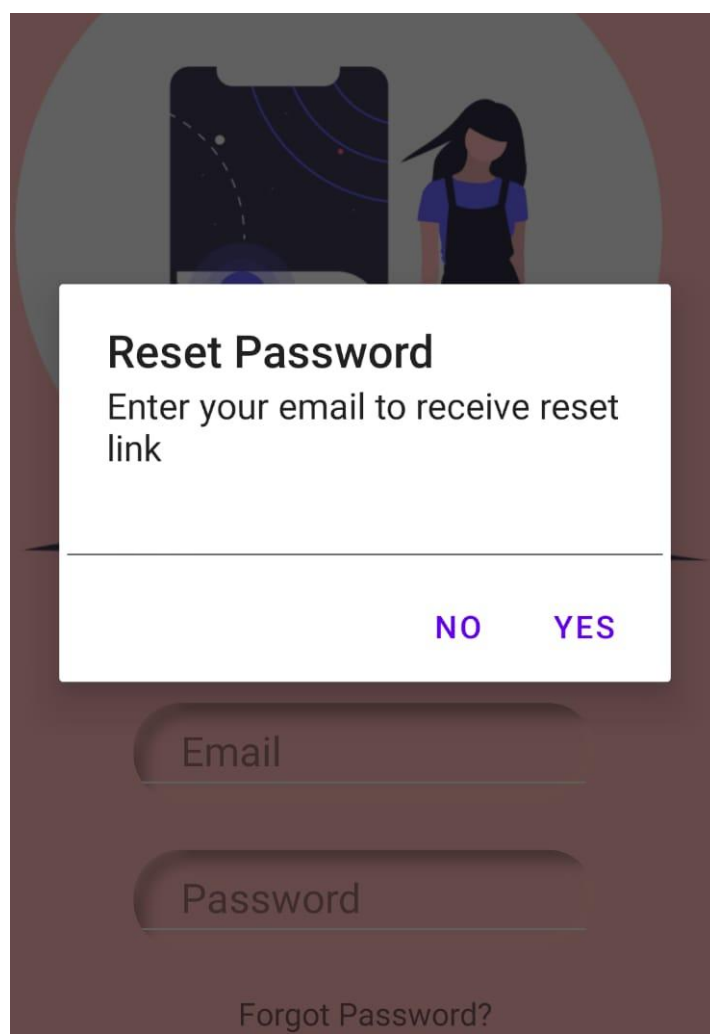
### *Register*

The image shows a mobile app registration screen with a dark blue background. At the top, the word "Account" is partially visible in a light pink font. Below it is a large circular graphic containing a blue shield with a white checkmark on the left and a stylized person in a dark suit on the right. Two small balloons, one blue and one pink, are at the bottom right of the circle. Below the graphic are two rounded rectangular input fields: the top one is labeled "Email" and the bottom one is labeled "Password", both in a light pink font. Below these fields is the word "or" in a small, light pink font. At the bottom is a white rectangular button with the Google "G" logo on the left and the text "Sign up with Google" on the right.

**Register screen**

If the user is not already registered, he can register for an account on the SOS app using the above interface. The user should provide a first name, last name, Email id, and a desired username and should create a password. If the email id is already registered with the SOS app a notification is shown to the user and he is not registered

### *Reset password*



The image shows a mobile application screen for password reset. At the top, there is a header with a stylized illustration of a person and a smartphone. Below this, a white modal dialog box is centered, containing the text "Reset Password" and "Enter your email to receive reset link". Below the dialog, there are two input fields labeled "Email" and "Password". At the bottom, there is a button labeled "Forgot Password?".

**Reset Password**  
Enter your email to receive reset link

NO YES

Email

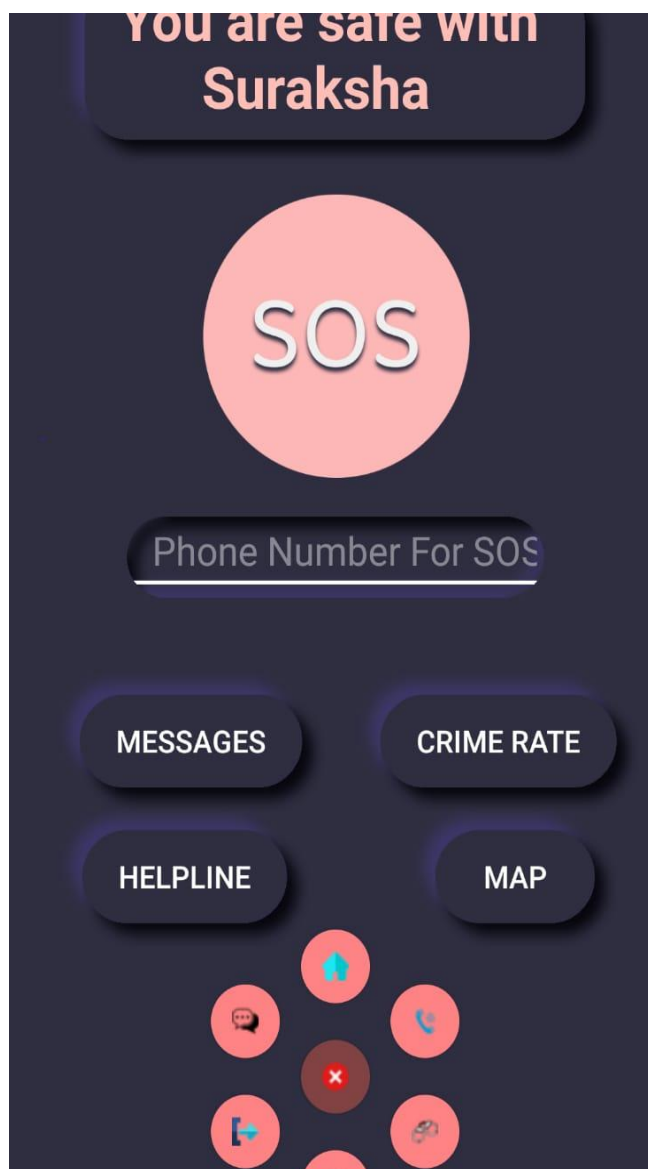
Password

Forgot Password?

**Password reset screen**

In case the user has forgotten his password, he can reset it by clicking on the forgot password button on the Login screen. He is then sent to reset the password page as above. The user can provide his email id and click on the reset button. A recovery email containing a temporary password is sent to the user on his email id.

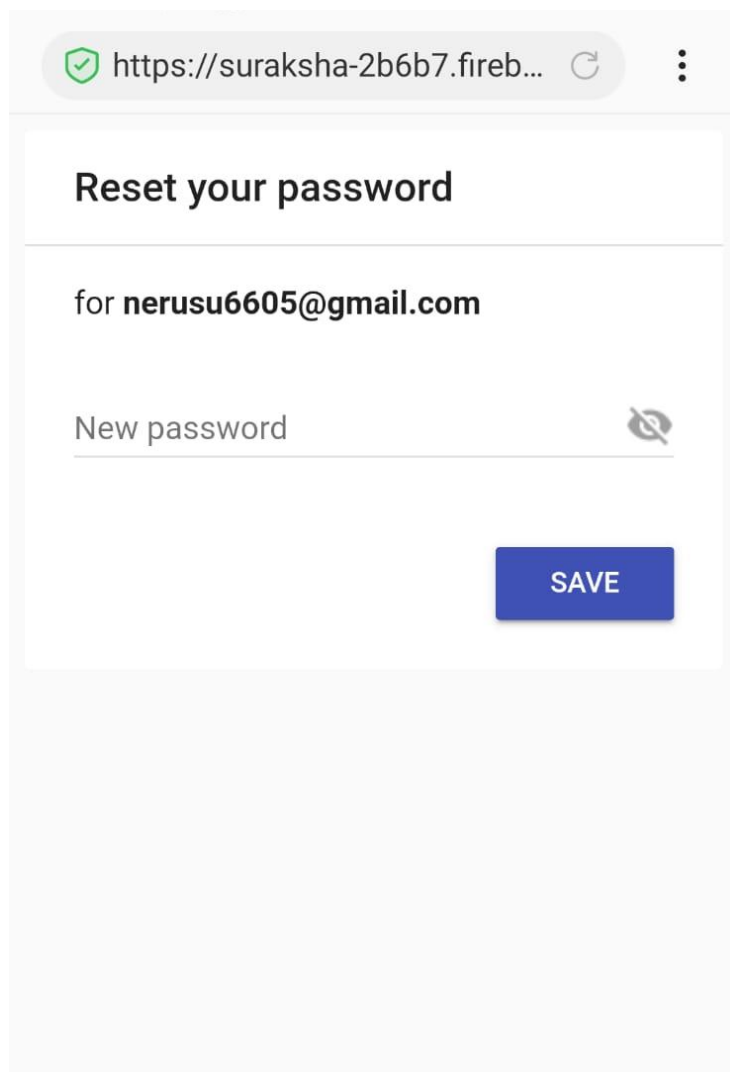
***Main screen***



**Main screen**


This is the main screen of the application. Once the user has logged in successfully, every time he opens the SOS app, this is the first screen presented to him. The screen contains a slider switch to enable/disable the buttons. The user can send panic text messages by clicking on the panic button. He can also send “I am OK” notifications using the OK button. Also the user may click on the 100/102/911 button to call police directly from within the app. The user also sees his current location at the bottom of this screen.

### *Change password*



Reset your password

for nerusu6605@gmail.com

New password 

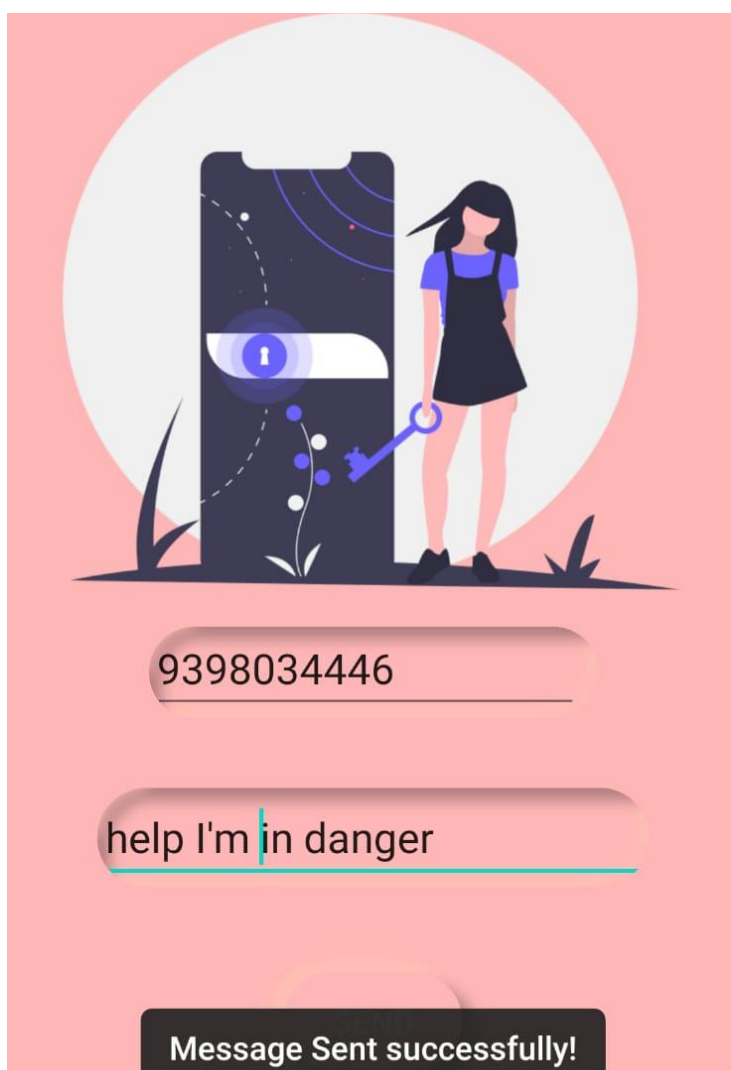
SAVE

### **Change password screen**

The user can change his password from the above screen. An email notification is sent on the registered email id regarding the change of password.

### *Set SMS Message*

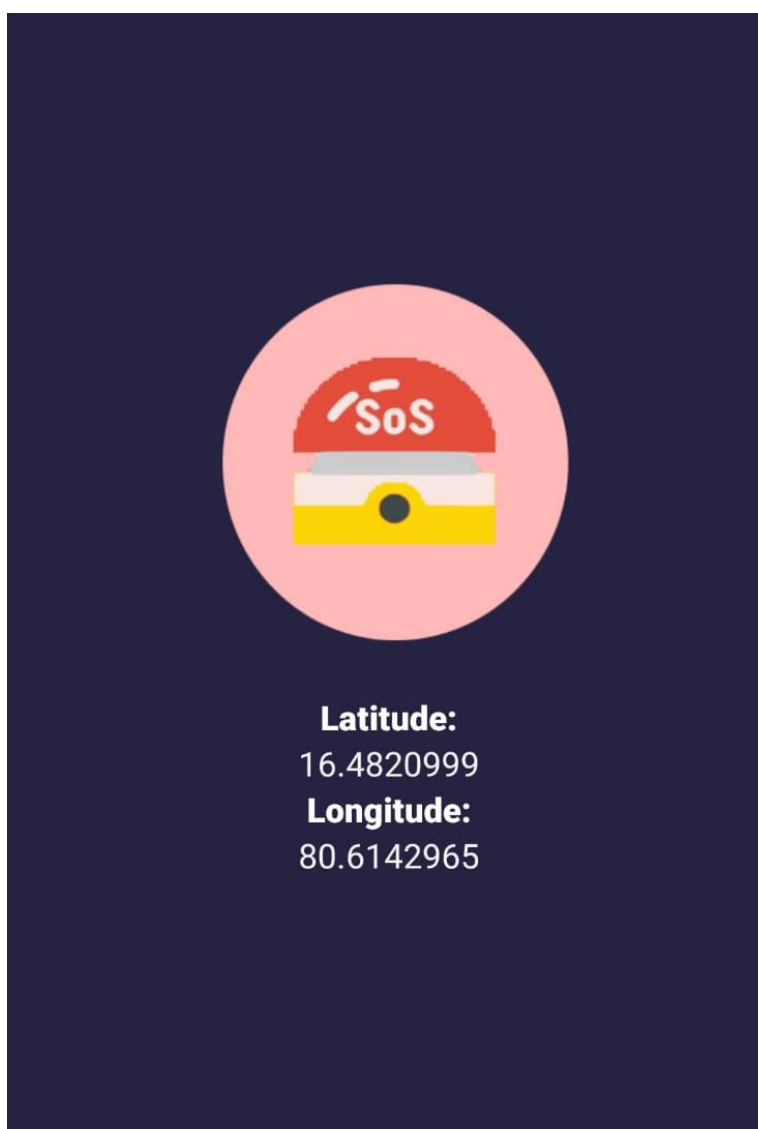




**SMS message screen**

The user is presented with the above dialog box to enter the message that he would like to send as a part of the SMS messages sent in case the panic button is pressed.

### ***Location***



**Location records screen**

The user can see his location in the above screen.

## **Chapter 8 - Testing**

Software testing is an essential phase in the development life cycle of an application. Testing ensures that the developed system meets its functional and non-functional requirements. Two important terms in software testing are Verification and Validation. Verification is the process of evaluating work-products like requirement specs, design specs and test cases etc. of different development phases to make sure that they meet the requirements for that phase. It ensures that the system is built in the right way. Whereas Validation is the process of evaluating the software at the end of the development phase to make sure that it meets the business requirements. It is used to make sure that the product fulfills its intended use and that the end product is built right. In this chapter we mainly validate the SOS app to make sure it meets the

requirements set initially.

One of the most important tools to test and debug an Android app is the Dalvik debug monitor server (DDMS) that is part of the Android framework. DDMS helps you to debug your code as it prints errors, warning and other information from your code. It also provide stack traces for exceptions on the Logcat output.

Various other testing strategies have been adopted to make sure the correctness of the SOS app. They are discussed in this chapter.

## Unit Testing

Unit testing is a strategy in software testing where individual components in a software are tested for correctness. In the SOS apps, these components are the Activities that are presented to the user as screens on the Android device, Fragments, Services and Receivers. Below is a list of test cases run on the SOS app, the test cases are categorized based on the target Activity:

### *Login Screen test cases*

S.No	Test Case	Pre-Condition	Post-Condition	Result
1.	On clicking the Login button	Email and password fields are empty	Show a notification to enter email and password	Pass
2.	On clicking the Login	Email and password fields	Show a notification	Pass

36

	button	contain data. Either the email or password is incorrect	about incorrect data and the user shall not get logged in	
--	--------	---	---	--

### Unit test cases - 1

### *Register screen test cases*

S.No	Test case	Pre-condition	Post-condition	Result
1.	On clicking the register button	One or more fields in the form is empty	A notification to enter all fields should be	Pass

			shown	
2.	On clicking the register button	All the fields are entered but the email is already registered with the SOS app	A notification that says email id is already registered should be shown	Pass
3.	On clicking the register button	All the fields are entered but the username is already registered with the SOS app	A notification that says username is already registered should be shown	Pass

### Unit test cases - 2

#### *Main screen test cases*

S.No	Test case	Pre-condition	Post-condition	Result
1.	On clicking either the panic, OK or the 911 button	Switch button is disabled	No notifications should be send	Pass
2.	On clicking the panic button	Switch button is enabled, SMS contacts or email contacts are not set	No messages must be sent. A notification to the user should be shown to add contacts	Pass
3.	On clicking the panic	SMS contacts and email	Message should be	Pass

	button	contacts are set from the contacts setting screen, Switch is enabled	sent. A notification after sending the message should be shown	
--	--------	--	--	--

4.	On clicking the OK button	SMS contacts and email contacts are set from the contacts setting screen, Switch is enabled	No messages must be send. A notification to the user should be shown to add contacts	Pass
5.	On clicking the OK button	SMS contacts and are set from the contacts setting screen, Switch is enabled	Message should be sent. A notification after sending the message should be shown	Pass
6.	On clicking the 100/102/1091 button	Switch is enabled	A call should be initiated to 100/102/1091	Pass
7.	On ending the call with 911	User must have clicked 911 button from the main page of SOS page	User should be sent back to the main page of SOS app	Pass
8.	Current location update	User must be on the main page	Address of the current location is shown at the bottom of the page	Pass

### Unit test cases - 3

### Integration testing

Integration testing is a strategy in software testing where different modules are combined and test to make sure they work together correctly. It is done when the components are unit test and the main objective is to test the interfaces between different components. Following are the integration test cases for the SOS app:

S.No	Test case	Pre-condition	Post-condition	Result
------	-----------	---------------	----------------	--------

1.	On clicking the Login button on login screen	Email and password fields contain data and both the fields are valid	User must get logged in and redirected to the main screen of the app	Pass
2.	On clicking the Register button on login screen	None	User must be redirected to the Register page	Pass
3.	On clicking the Reset	None	User must be redirected	Pass

	password button on login screen		to the Reset password page	
4.	On clicking the register button on the register screen	All fields are entered, email and username are not already registered with the SOS app	User should be registered and directed to the Registered page	Pass
5.	On clicking the back to login screen on the registered screen	User has registered.	User must be redirected to the Login screen	Pass
6.	On clicking the setting icon on the action bar on the main screen	None	User must be sent to the setting page with personal settings displayed	Pass

6.	Location updates	User must have checked the location tracking checkbox	Location tracking service must start and remain started as long as the check box is checked. A notification should be shown to the user about the start of location tracking	Pass
7.	On clicking the change password button on personal setting screen	None	User must be sent to the change password screen	Pass
8.	On clicking the logout Button on personal setting screen	None	User must be logged out and sent to the login screen	Pass
5.	On clicking the number text box on contacts setting screen	User must have clicked on Set SMS contact	Contact book app should be opened. User must be able to click on a contact and add his	Pass

			name and primary phone number in the app. If the contact does not have a number only the name must be entered and number must	
--	--	--	---	--

			display empty	
6.	On clicking the email text box on contacts setting screen	User must have clicked on Set Email contact	Contact book app should be opened. User must be able to click on a contact and add his name and primary email id in the app. If the contact does not have an email id only the name must be entered and email must display empty	Pass
6.	On clicking the select button on location history screen	All fields on the location history page contain valid data	User must be directed to a screen that contains a map	Pass

### Integration test cases

### Performance testing

Performance testing is a type of non-functional testing performed to determine how fast the system can perform under certain workload. In the SOS app performance testing is done to make sure that there are no significant lags in the user interface while using the application due to background tasks etc. Android SDK provide a graphical tool called Traceview [7] to profile the performance of the application.

The performance was tested on an android device “OPPO” running Android Kitkat (4.4.2).

### Performance testing

In addition to the above screens the time taken to submit the text messages. The actual time of delivery to the physical device of the recipient is a factor not in control of the SOS app



rather it is determined by the Carrier Service or the Internet Service Provider.

## **Chapter 9- Future Work**

The current work on the SOS app has a lot of essential features that would be used in case of an emergency situation like sending text messages , and making calls to 100/102/1091 from within the app on tap of a single button. An app for such a purpose has a lot of scope for enhancement. In the future the app may include features like –

- i. A home screen widget that can be used as a triggering point to send panic notifications. A user would then not have to open the app to send these panic notifications.
- ii. Initiating a call to a number set from within the application when the user presses the panic button.
- iii. The app can also listen to incoming messages from the set contacts. If these message have a pre-defined text like “UPDATE LOCATION” the app can reply with a text message containing the current location or for some other text like “AUDIO” in which case the app can record a short audio and send it as an email to the person. This is very helpful as you may have already pressed the panic button and may be in some trouble where you cannot reply. This way the person can track you constantly and also understand something about the nature of the emergency from the audio clip.
- iv. Setting up a password to stop the application.

## **Chapter 10 - Conclusion**

SOS is an essential app to have on a Smartphone. It is a personal security app that lets you send notifications to certain people via text messages and emails in case of emergencies. It also gives you the ability to call 911 on the tap of a single button. The app also keeps a track of your current location so that you always know the address of where you are. This can be very helpful if you would need to make a call to 100/102/1091. The text messages and email sent also have this location information.

The app can also track your location periodically and store it permanently enabling you to see your location history. You can for any particular day see the various locations that you have been to using the app.

SOS app was my first attempt at an Android application. It gave me very good exposure to the Android platform and mobile development in general.

## Chapter 11 - Bibliography

1. **IDC.** [Online] May 25, 2014. <http://www.idc.com/getdoc.jsp?containerId=prUS24676414>.
2. **Vogel, Lars.** Android System Architecture. [Online] May 27, 2014.  
<http://www.vogella.com/tutorials/Android/article.html>.
3. **Developers, Google.** Google Maps Android API V2. [Online] April 15, 2014.  
<https://developers.google.com/maps/documentation/Android/>.
4. **Developers, Android.** Activity. [Online] March 18, 2014.  
<http://developer.Android.com/training/basics/Activity-lifecycle/starting.html>.
5. **Sutcliffe, Geoff.** Activity Life Cycle. [Online] May 29, 2014.  
<http://www.cs.miami.edu/~geoff/Courses/CSC300-13S/Content/ActivityLifeCycle.html>.
6. **Developers, Android.** Intents. [Online] March 20, 2014.  
<http://developer.Android.com/guide/components/Intents-filters.html>.
7. **Developers,Android.** Traceview tool. [Online] May 29, 2014.  
<http://developer.Android.com/tools/debugging/debugging-tracing.html>.
8. **Developers,Android** Services. [Online] April 7, 2014.  
<http://developer.Android.com/guide/components/services.html>.
9. **Developers,Android** Broadcast Receivers. [Online] April 12, 2014.  
<http://developer.Android.com/reference/Android/content/BroadcastReceiver.html>.
10. **Developers,Android** Android basics. [Online] March 10, 2014.  
<http://developer.Android.com/training/index.html>.
11. **Android Apps for Sms:**  
<http://www.fileguru.com/Android-Apps-ForSMS/info>