

SOS – AN ANDROID APPLICATION FOR EMERGENCIES

By

AKASH SURYAWANSHI

B.E., SHRI GOVINDRAM SEKSARIA INSTITUTE OF TECH & SCIENCE, INDIA, 2010

A REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computing and Information Sciences
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Dr. Daniel A. Andresen

Abstract

The aim of the project is to develop an Android application that lets its users to send notifications in case of an emergency or a panic situation. The users can send multiple text messages and emails on the press of a single button. The phone numbers, email ids and the contents of the text and email messages can be set from within the application. The text messages and emails sent, along with the content, also have the last known location of the user. This is very helpful in tracking the whereabouts of the person. The user can also call 911 directly from within the application, if the nature of the situation demands it.

Additionally the user of the application may allow the app to track their location. If this option is selected, the application fetches the device's location at about every 15 minutes and stores it in a database. This information is very useful and can be used in a variety of ways. One such use of the location data is from within the Android app where the user can view a map that shows their location history over a period of time for a particular day.

Table of Contents

List of Figures	v
List of Tables	vi
Acknowledgements	vii
Chapter 1 - Introduction	1
1.1 Motivation	1
1.2 Project Description.....	1
Chapter 2 - Background.....	3
2.1 Android	3
2.1.1 Android Architecture	3
2.2 Google Maps Android API v2	5
2.3 PHP.....	6
2.4 MySQL	6
2.5 JSON.....	7
Chapter 3 - Requirement Analysis	8
3.1 Requirements Gathering	8
3.2 Requirement Specifications	9
3.2.1 Software Requirements	9
3.2.2 Hardware Requirements.....	10
Chapter 4 - Architecture & Design	11
4.1 System Architecture	11
4.2 Design Diagrams	13
4.2.1 Use case diagrams	13
Chapter 5 - Android Framework Components.....	15
5.1 AndroidManifest.xml	15
5.2 Activities	18
5.3 Intents.....	19
Chapter 6 - Implementation.....	20
6.1 Graphical User Interface.....	20
6.1.1 Login	20

6.1.2 Register.....	21
6.1.3 Registered	22
6.1.4 Reset password.....	23
6.1.5 Main screen.....	24
6.1.6 Personal Setting page	25
6.1.7 Change password.....	26
6.1.8 Contacts setting	27
6.1.9 Set SMS Contacts.....	28
6.1.10 Set SMS Message	29
6.1.11 Set Email contacts	30
6.1.12 Set Email message	31
6.1.13 Location records	32
6.1.14 Date Picker.....	33
6.1.15 Time Picker	34
6.1.16 Map	35
Chapter 7 - Testing	36
7.1 Unit Testing.....	36
7.1.1 Login Screen test cases.....	36
7.1.2 Register screen test cases.....	37
7.1.3 Main screen test cases.....	37
7.1.4 Personal settings screen test cases	38
7.1.5 Contacts setting screen test cases.....	39
7.1.6 Location records screen test cases	40
7.1.7 Map test cases	41
7.2 Integration testing	41
7.3 Performance testing.....	43
Chapter 8 - Future Work	45
Chapter 9 - Conclusion	46
Chapter 10 - Bibliography.....	47

List of Figures

Figure 2-1 Android Architecture[2]	3
Figure 4-1 System Architecture diagram.....	11
Figure 4-2 Use case diagram - 1	13
Figure 4-3 Use case diagram - 2	14
Figure 4-4 Use case diagram – 3	14
Figure 5-1 Activity lifecycle[5]	18
Figure 6-1 Login Screen	20
Figure 6-2 Register screen	21
Figure 6-3 Registered screen.....	22
Figure 6-4 Password reset screen.....	23
Figure 6-5 Main screen	24
Figure 6-6 Personal setting screen.....	25
Figure 6-7 Change password screen	26
Figure 6-8 Contacts setting screen.....	27
Figure 6-9 SMS contacts screen.....	28
Figure 6-10 SMS message screen.....	29
Figure 6-11 Email contacts screen.....	30
Figure 6-12 Email message screen	31
Figure 6-13 Location records screen.....	32
Figure 6-14 Date picker screen.....	33
Figure 6-15 Time picker screen.....	34

List of Tables

Table 6-1 Lines of Code (LOC).....	20
Table 7-1 Unit test cases - 1.....	37
Table 7-2 Unit test cases - 2.....	37
Table 7-3 Unit test cases - 3.....	38
Table 7-4 Unit test cases - 4.....	39
Table 7-5 Unit test cases - 5.....	40
Table 7-6 Unit test cases - 6.....	40
Table 7-7 Unit test cases - 7.....	41
Table 7-8 Integration test cases	43
Table 7-9 Performance testing.....	44

Acknowledgements

This project would not have been possible without the support and guidance of my Major Professor Dr. Daniel A. Andresen. 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.

I would also like to express my special gratitude and thanks to Dr. Torben Amtoft and Dr. Mitchell L. Neilsen for serving on my committee and for their kind assistance and constant guidance.

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

Chapter 1 - Introduction

SOS (which stands for Save Our Souls or Save Our Ships) has primarily been used as an International Morse code distress signal. It is commonly used in navigation by Sailors when under attack by Pirates or when they need help of some kind. But the signal is not limited to navigation and is used in a more general sense whenever a notification has to be sent about a situation that requires immediate attention.

1.1 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 in that is a Smartphone. Almost everyone today carry a Smartphone with them as they become more and more affordable and easily available. Also within the Smartphone market Android is the clear leader in terms of market share. According to one report, 78.1 % of the total Smartphones sold in 2013 were the Smartphones that run on Android Operating System [1]. Hence developing an Android application becomes an obvious choice.

1.2 Project Description

SOS is an application that is meant to run on Android devices mainly smartphones but also tablets that support Cellular Service. The main functions and features of the application are –

- i. The user of the application has to login by entering a username and password the first time he opens the app on his device. He then remains logged into the application until he logs out explicitly.
- 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 911 directly from within the application by pressing the 911 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 main screen. This way he would know his exact location and refer to it in case he makes a call to 911. This location is also sent as a part of the text and email messages.
- vi. The user can set the contacts to send the text message and emails 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 and the email 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, the user can at a later point see the various locations he had been to for a time interval on a particular day. He would be presented with a map that display these locations. The user can see the address and the time he was at that location by clicking on the marker for a location.

Chapter 2 - Background

2.1 Android

Android is one of the most widely used Mobile Operating System today. It is a software bunch comprising not only of the Operating System but also middleware and key applications. Some of the most important features of an Android operating system is that it enables reuse and replacement of components, it is optimized for mobile devices and tablets, it is based on the open source Web kit engine and supports 2-D and 3-D graphics using OpenGL-ES standard.

2.1.1 Android Architecture

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

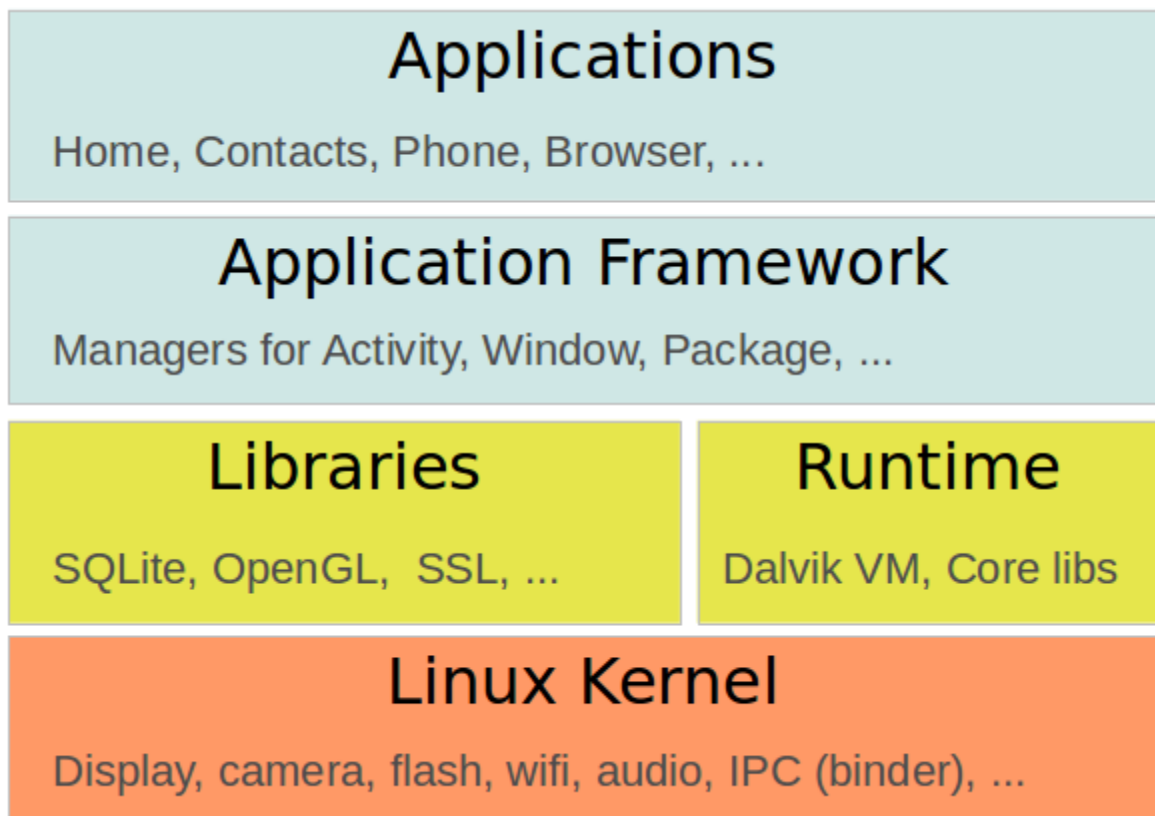


Figure 2-1 Android Architecture [2]

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 set 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 on to the screen, SQLite which is a useful repository for storing and sharing of application data.

Android Runtime – The Android runtime mainly consist 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.

2.2 Google Maps Android API v2

Google provides a very nice, comprehensive API for developers working with Android and who want to use google maps in their application. Using this API one can easily add maps to their apps and the API automatically handles access to google maps server, data downloading, map display and response to gestures. Additionally the API can be used to add the following on to the maps:

- i. Markers that are used to show specific position on the map.
- ii. Line segments (Polylines)
- iii. Enclosed segments (Polygons)
- iv. Various images that are shown on the map (overlays) like zoom control, compass etc.

To get started using the google maps Android API v2 one has to first obtain an API key specific to your application by using the apps signing certificate from google's API console. One should also add Google play services SDK as a project on the local disk and add it as an external library in the app.[3]

The next step is to add the google play services version to the apps Manifest file by including

```
<meta-data
    Android:name="com.google.Android.gms.version"
    Android:value="@integer/google_play_services_version"/>
```

Once an API key is obtained specific to the app add the following to Manifest file

```
<meta-data
    Android:name="com.google.Android.maps.v2.API_KEY"
    Android:value="Your_API_KEY"/>
```

One also needs to add the following permission in the Manifest file

```
<uses-permission Android:name="Android.permission.INTERNET" />
<uses-permission Android:name="Android.permission.ACCESS_FINE_LOCATION" />
<uses-permission Android:name="Android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
    Android:name="com.google.Android.providers.gsf.permission.READ_GSERVICES"/>
```

In order to add a google map you must add a fragment to your Activity using

```
<fragment xmlns:Android="http://schemas.Android.com/apk/res/Android"
    Android:id="@+id/map"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"
    Android:name="com.google.Android.gms.maps.MapFragment"/>
```

To get an instance of google map you should call the `getMap()` function on a `MapFragment` object by

```
GoogleMap map = ((MapFragment)
getFragmentManager().findFragmentById(R.id.map)).getMap();
```

You can then add markers to this map by calling the `addMarker()` method on the map

```
map.addMarker(new MarkerOptions()
    .title(address)
    .position(new LatLng(latitude, longitude))
    .snippet(time));
```

2.3 PHP

PHP is one of the most popular server side scripting language. Php scripts can be embedded into the HTML pages. The start and the end tags for a PHP script are `<? Php` and `?>` respectively. PHP pages can contain text, HTML, CSS, JavaScript and more. They have the `.php` extension and are executed on the server.

PHP can be help you get the following things done –

- i. Create dynamic web pages by generating data based on user requests.
- ii. Collect and process data from forms submitted by the user.
- iii. Work with files on the server
- iv. Manipulate some database
- v. Send and receive cookies etc.

Apart from the above reasons PHP can run on various platforms and is compatible with almost all server in use. It is free and has a big support community.

2.4 MySQL

It is one of the most widely used Relational database management system (RDBMS). It is open source and available for free. It scales very well for large quantities of data, is fast, reliable and easy to use. Like most other RDBMS it uses structured query language (SQL) for accessing and manipulating data.

2.5 JSON

JSON stands for JavaScript Object Notation. It is a light weight format for storing and exchanging objects that contain name/value pairs, array and other objects. It is language independent i.e. although it uses JavaScript syntax, it is still language and platform independent and many different languages support JSON parsers and libraries. The two main advantages of JSON over other ways of exchanging data like XML are –

- i. JSON is usually smaller than the corresponding XML.
- ii. JSON is easier to parse and faster.

For example –

```
{
  "products":[
    {"id": 1,
     "name":"car"},
    {"id":2,
     "name":"truck"}
  ]
}
```

The above JSON has two products (JSONArray) where each product is an Object (JSONObject) with keys id and name and their corresponding values.

Chapter 3 - Requirement Analysis

3.1 Requirements Gathering

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

Requirement gathering for the SOS app started with brain storming 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 signal 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 message and email 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. The user shall be able to start/stop location tracking. 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 setting 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.

3.2 Requirement Specifications

3.2.1 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

3.2.2 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 4 - Architecture & Design

4.1 System Architecture

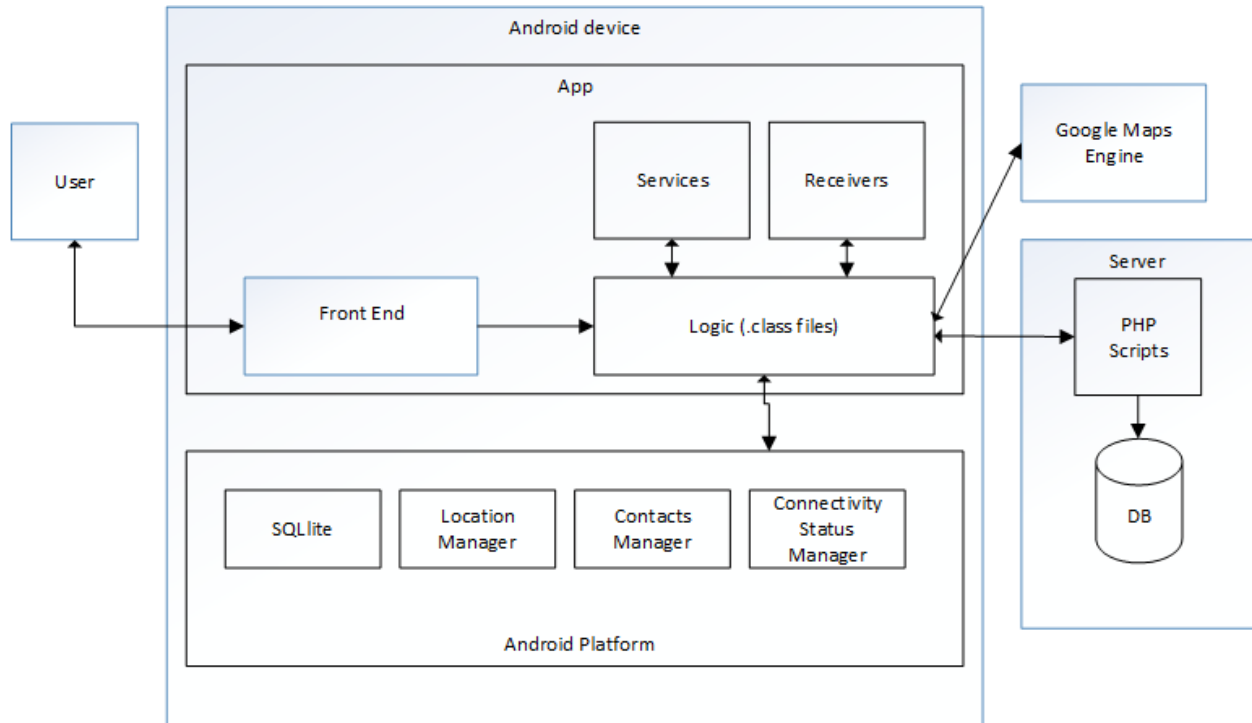


Figure 4-1 System Architecture diagram

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 Google maps Android API v2 to connect the Android app with Google Maps Engine to render map and markers of them.
- 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.
 - vi. SQLite – Android platform provides libraries for SQLite database. A SQLite database is a relational database that is local to an Android device. It requires no configuration and is available to use for an app developer. For example – SQLite is used in the app to store various information about a user, his last known location or in case there is no internet connectivity SQLite database stores the location until the internet connection is back up and the records are sent to the database.
 - 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 send to the SOS application and displayed in these text boxes.
 - ix. Connectivity status manager- This system service tells the SOS application about changes in the connectivity status for the device. The application uses it to make sure that if there is no active internet connection on the device at the time of sending the fetched location to the database, it needs to store the location in the SQLite database. The application should also listen to the connectivity manager to make sure that once the internet is up

again all the locations that are stored in the SQLite database are sent over to the database on the server and then the SQLite database is cleared.

- x. Google Maps Engine – SOS app uses google maps Android API v2 to work with maps. When this API is used, calls are made to the google maps engine to fetch the map or place various markers on it.
- xi. PHP Scripts – The SOS app sends JSON objects to various PHP scripts using HTTP POST methods. These scripts interact with the database on the server to give a response to the app. For example- When the user logs into the application a JSON object is created that contains the values that the user entered into the app. This JSON is sent to a PHP script login.php on the server that queries the DB to validate the user. If the user credentials are valid a success response is sent back to the SOS app.
- xii. Database- This is the MySQL database on the server. It is used to permanently store the data for the SOS app. It stores information of various user, location history etc.

4.2 Design Diagrams

4.2.1 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

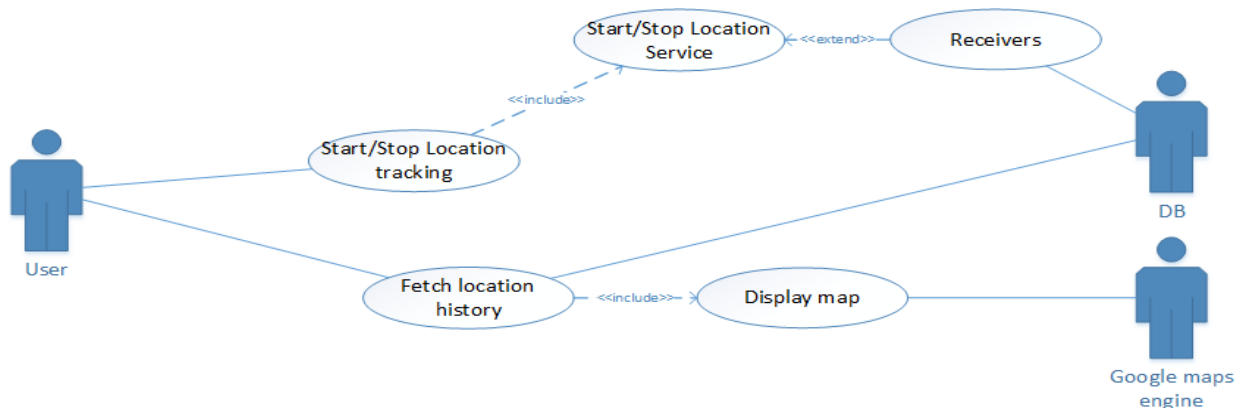


Figure 4-2 Use case diagram - 1

2. Use case for sending notifications

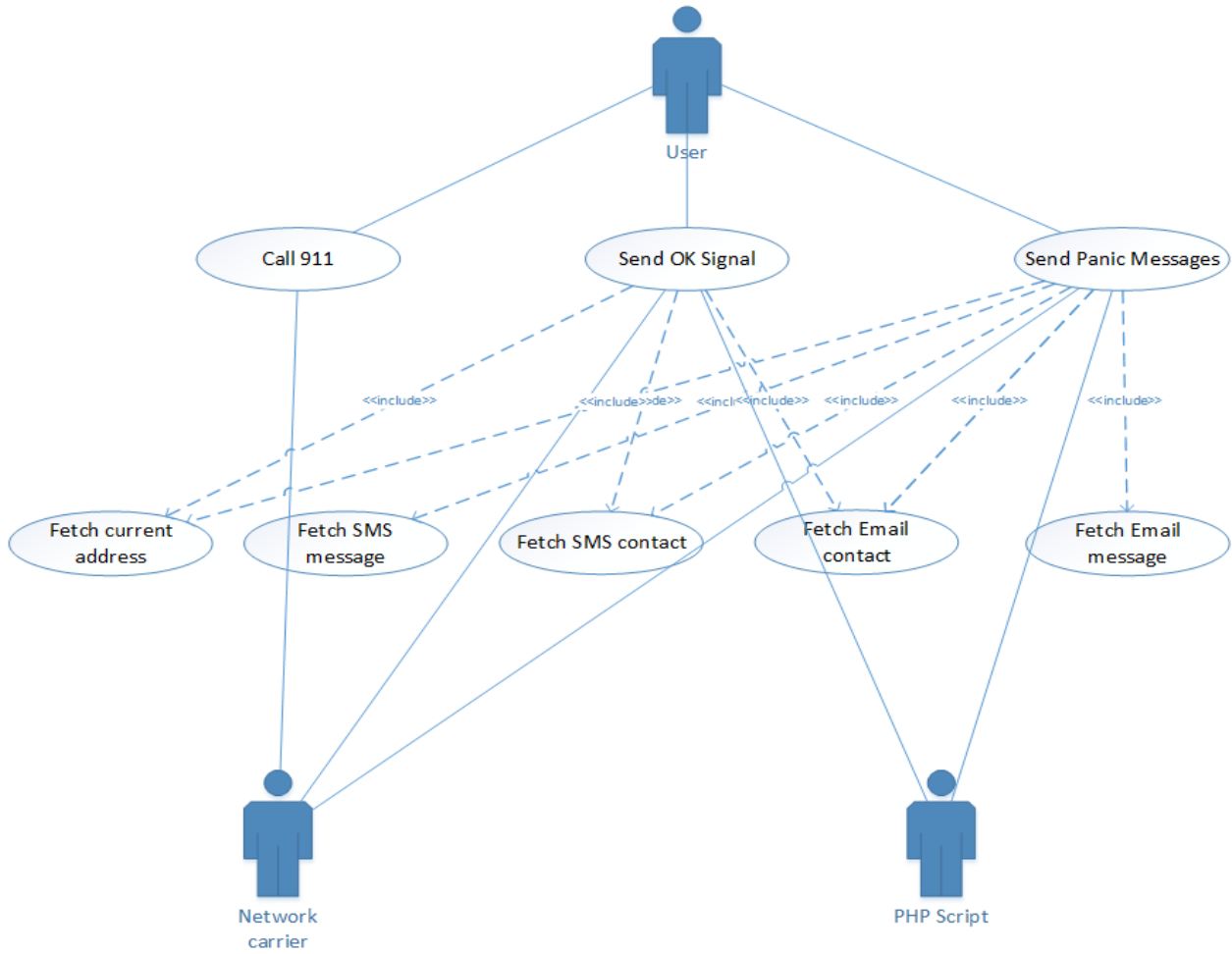


Figure 4-3 Use case diagram - 2

3. Use case for setting contacts

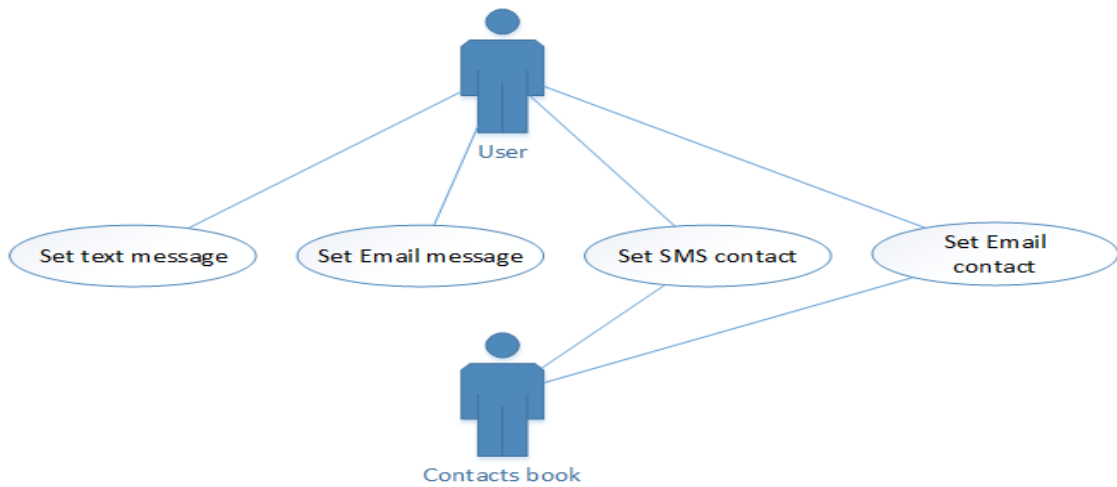


Figure 4-4 Use case diagram – 3

Chapter 5 - 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.

5.1 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.sos"
    android:versionCode="1"
    android:versionName="1.0">

    <uses-sdk android:minSdkVersion="14"
        android:targetSdkVersion="17"/>

    <application android:label="@string/app_name"
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:theme="@android:style/Theme.Holo.Light.DarkActionBar">

        <Activity android:name="Login"
            android:label="@string/app_name">
            <Intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </Intent-filter>
        </Activity>
```

```

<Activity Android:name="Register"
    Android:label="@string/title_Activity_register"
    Android:parentActivityName="com.sos.Login">
    </Activity>

<Activity Android:name="Registered"
    Android:label="@string/title_Activity_registered"
    Android:parentActivityName="com.sos.Login">
    </Activity>

<Activity Android:name="Main"
    Android:label="@string/title_Activity_main">
</Activity>

<Activity Android:name="Settings"
    Android:label="@string/title_Activity_settings"
    Android:windowSoftInputMode="adjustPan|stateHidden"
    Android:parentActivityName="com.sos.Main">
</Activity>

<Activity Android:name="PasswordReset"
    Android:label="@string/title_Activity_passwordreset"
    Android:parentActivityName="com.sos.Login">
    </Activity>

<Activity Android:name="ChangePassword"
    Android:label="@string/title_Activity_passwordchange"
    Android:parentActivityName="com.sos.Settings">
    </Activity>

<Activity Android:name="Map"
    Android:label="@string/title_Activity_map"
    Android:parentActivityName="com.sos.Settings">
    </Activity>

<!-- Register the different receivers -->

<receiver
    Android:name="com.sos.Library.ConnectionStatusChange"
    Android:label="ConnectionStatusChange" >
    <Intent-filter>
        <action Android:name="Android.net.conn.CONNECTIVITY_CHANGE" />
        <action Android:name="Android.net.wifi.WIFI_STATE_CHANGED"/>
    </Intent-filter>
</receiver>
<receiver Android:name="com.sos.Library.LocationReceiver" />
<receiver Android:name="com.sos.Library.LocationReceiverForDB" />
<receiver Android:name="com.sos.Library.LocationPollerDB" />
<receiver Android:name="com.sos.Library.LocationPoller" />

<!-- List the service in the app here -->
<service Android:name="com.sos.Library.LocationPollerService" />
<service Android:name="com.sos.Library.LocationPollerServiceDB" />

```

```

    <meta-data
    Android:name="com.google.Android.gms.version"
    Android:value="@integer/google_play_services_version"/>

    <meta-data
    Android:name="com.google.Android.maps.v2.API_KEY"
    Android:value="AIzaSyDL8_XwvIfSu6KrcgkKaBf2Kg1omGIpwm8"/>

</application>

<!-- Give the required permissions to the app here-->

<uses-permission Android:name="Android.permission.INTERNET" />
<uses-permission Android:name="Android.permission.ACCESS_NETWORK_STATE" />
<uses-permission Android:name="Android.permission.READ_CONTACTS"/>
<uses-permission Android:name="Android.permission.SEND_SMS"/>
<uses-permission Android:name="Android.permission.CALL_PHONE"/>
<uses-permission Android:name="Android.permission.READ_PHONE_STATE"/>
<uses-permission Android:name="Android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission Android:name="Android.permission.WAKE_LOCK" />
<uses-permission Android:name="Android.permission.ACCESS_FINE_LOCATION" />
<uses-permission Android:name="Android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
Android:name="com.google.Android.providers.gsf.permission.READ_GSERVICES"/>
<uses-feature
    Android:gLESVersion="0x00020000"
    Android:required="true"/>

</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.

5.3 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 it 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 6 - Implementation

The SOS app is a collection of Activities and Fragments that are presented to the user. These Activities and Fragment have associate 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. The total lines for the app are 5846 which includes Java and XML files. Additionally there are 262 lines of PHP code to enable the app to interact with the server. The breakdown for the lines of code is as follows

Language	LOC
Java	5002
PHP	262
XML	844

Table 6-1 Lines of Code (LOC)

6.1 Graphical User Interface

6.1.1 Login

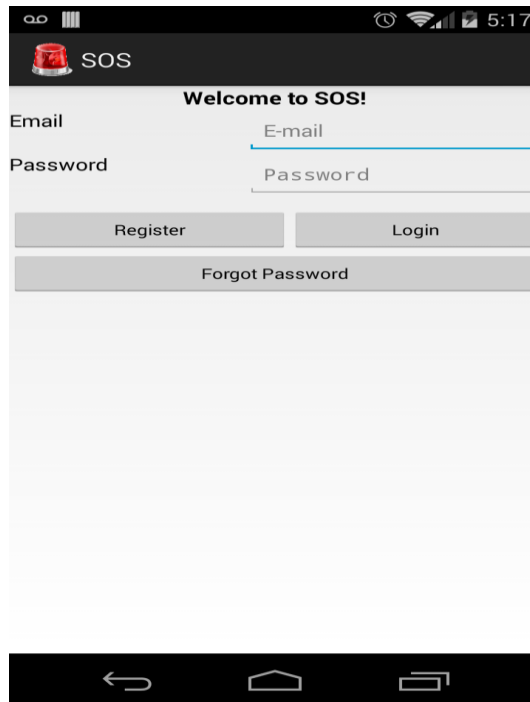
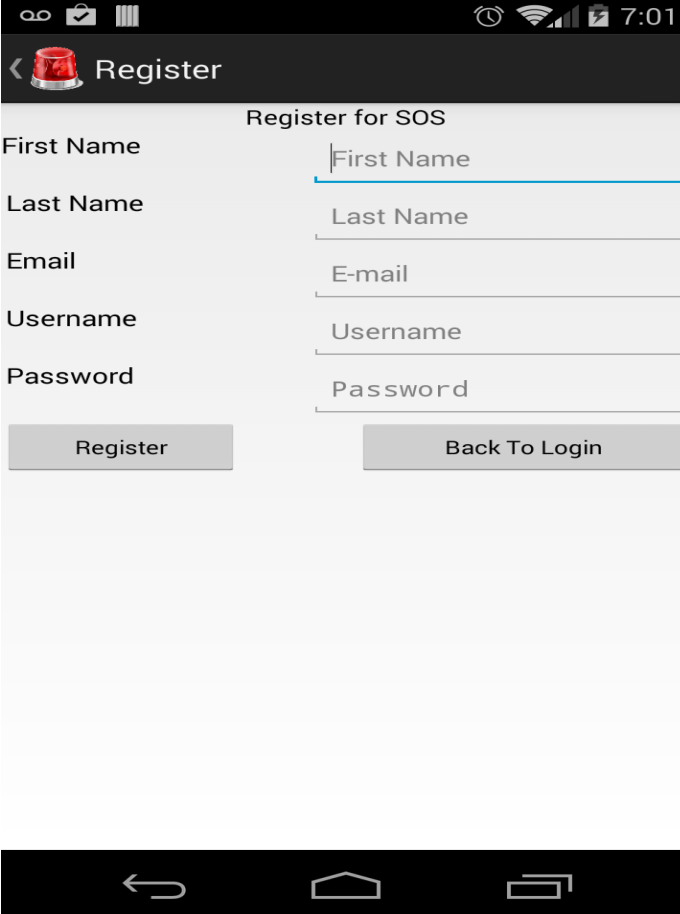


Figure 6-1 Login Screen

The user should log in to 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 Forgot password if he does not remember his password and wants to reset it.

6.1.2 Register



The screenshot shows the 'Register for SOS' screen. At the top, there is a navigation bar with a back arrow, a red alarm icon, and the title 'Register'. Below the navigation bar, the title 'Register for SOS' is displayed. The form consists of five input fields: 'First Name', 'Last Name', 'Email', 'Username', and 'Password'. Each field has a placeholder text corresponding to its label. Below the form, there are two buttons: 'Register' and 'Back To Login'. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Figure 6-2 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

6.1.3 Registered

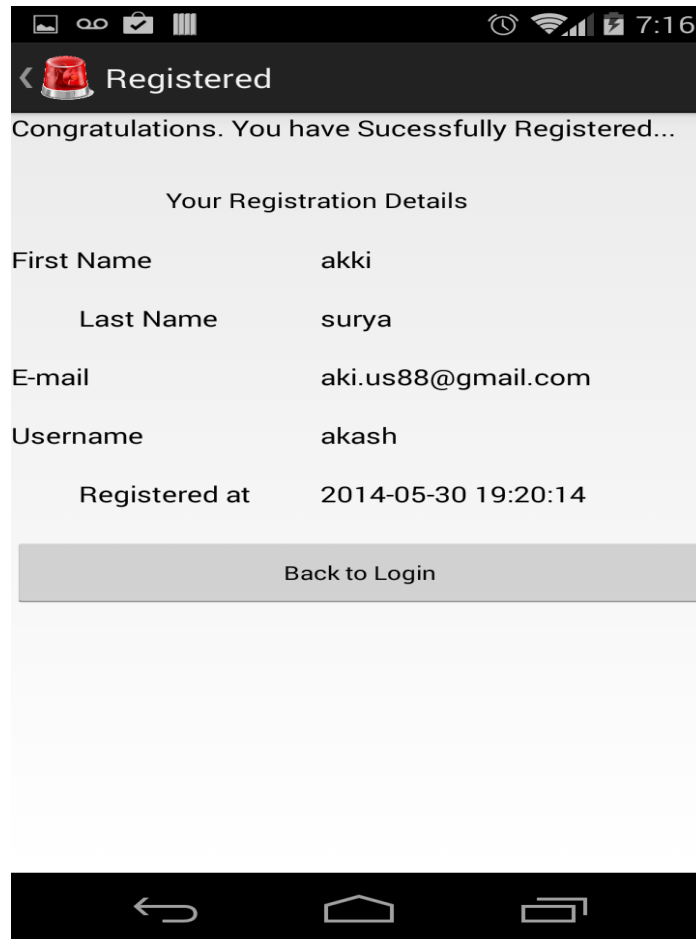


Figure 6-3 Registered screen

Once the user has successfully registered he sees the above screen as a confirmation. The screen shows the various details the user registered with.

6.1.4 Reset password

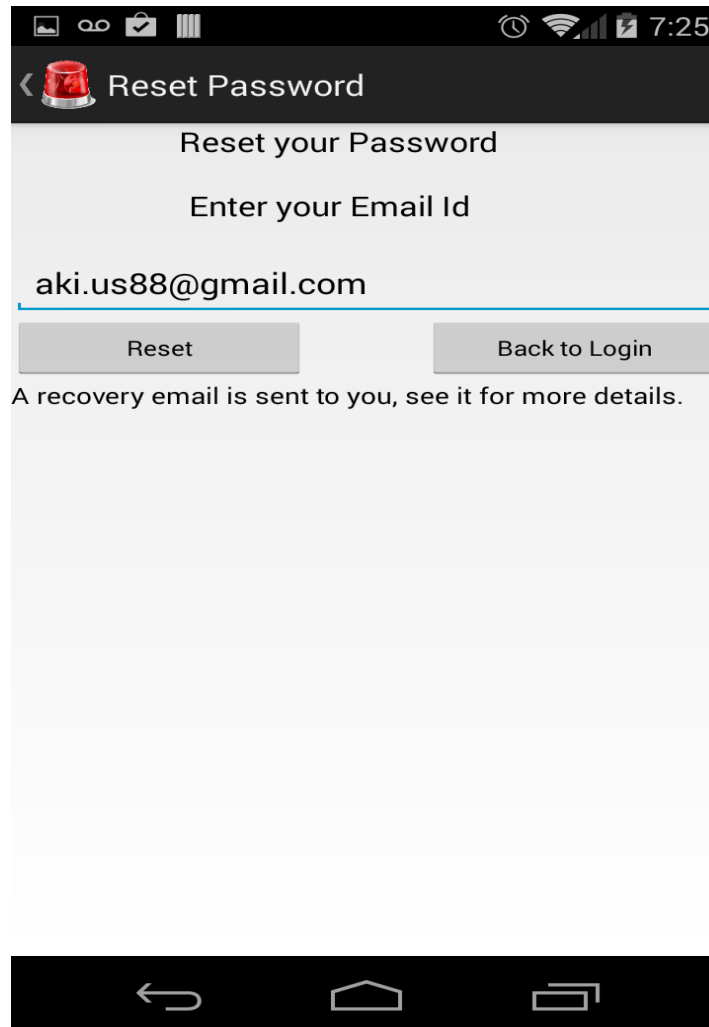


Figure 6-4 Password reset screen

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 send to reset 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.

6.1.5 Main screen

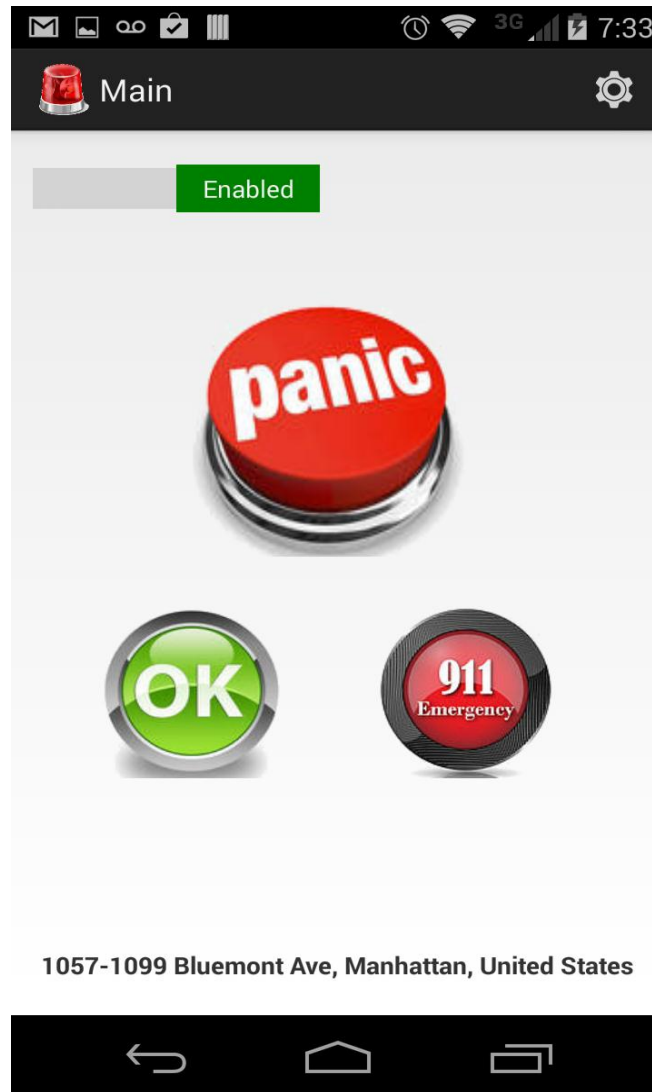


Figure 6-5 Main screen

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 message and email 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 911 button to call 911 directly from within the app. The user also sees his current location at the bottom of this screen.

6.1.6 Personal Setting page

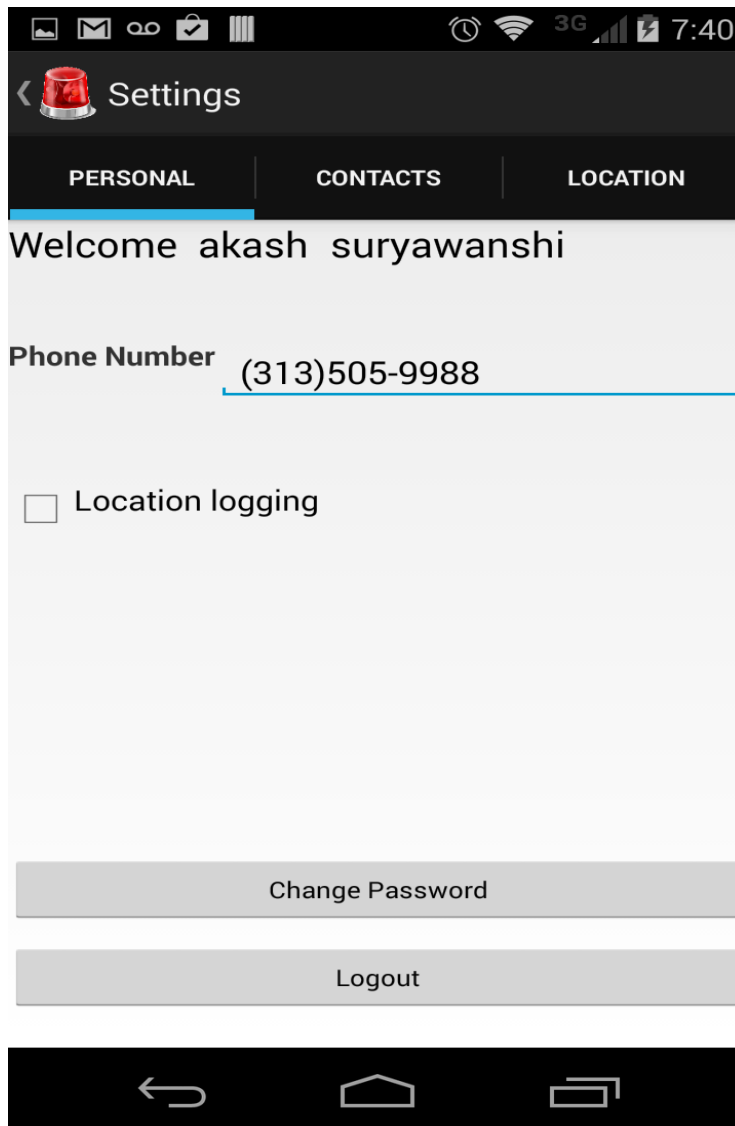


Figure 6-6 Personal setting screen

The user can click on the settings icon on the action bar to open the tabbed interface for settings. On the personal setting he can enter his phone number, start location tracking by clicking on the check box for location logging. The user can also change his password and logout by clicking on the respective buttons.

6.1.7 Change password

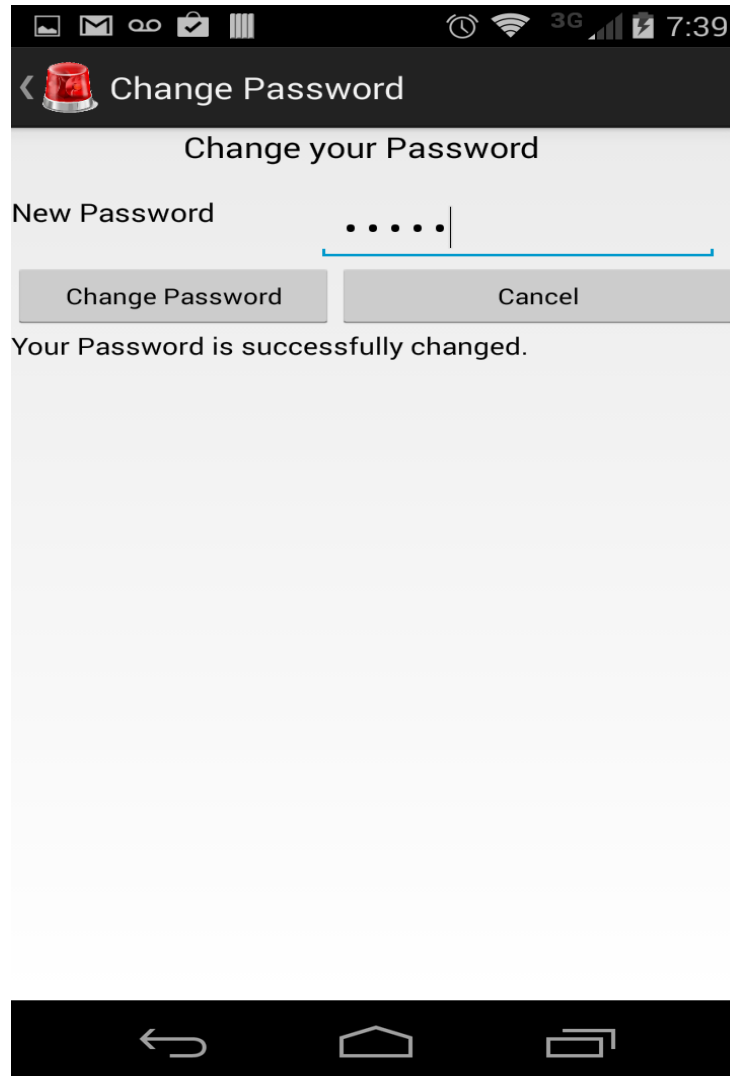


Figure 6-7 Change password screen

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.

6.1.8 Contacts setting

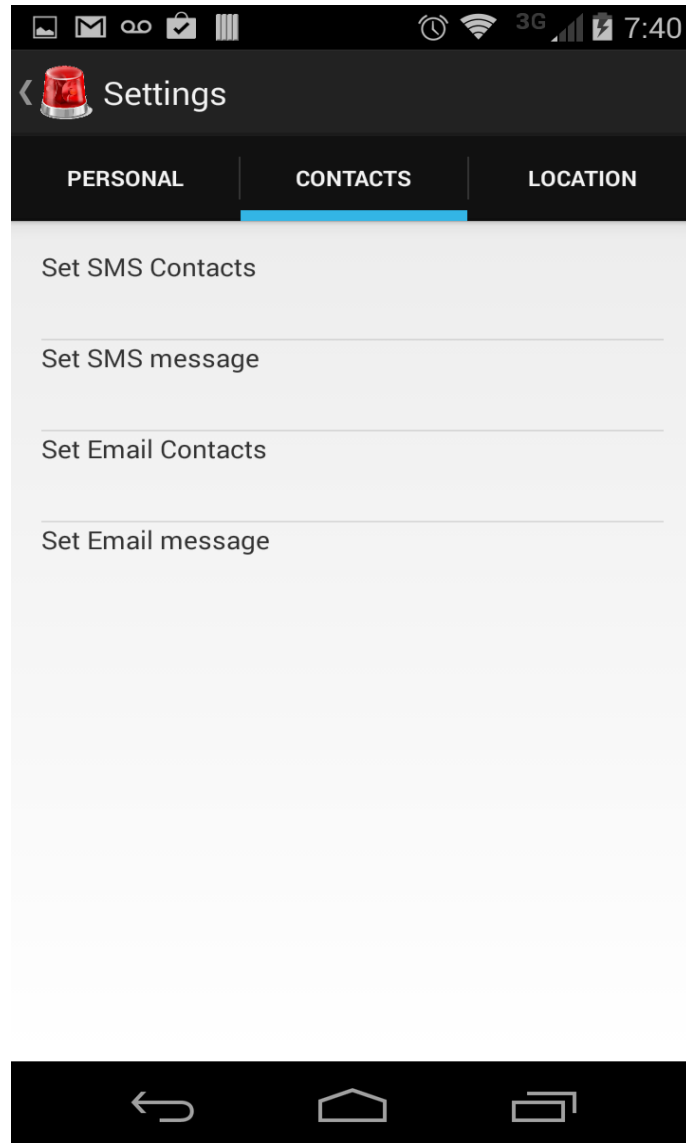


Figure 6-8 Contacts setting screen

The user can set various contacts to send the text message and the emails to from the above screen. He can also set the message that must be sent in these text and email messages separately.

6.1.9 Set SMS Contacts

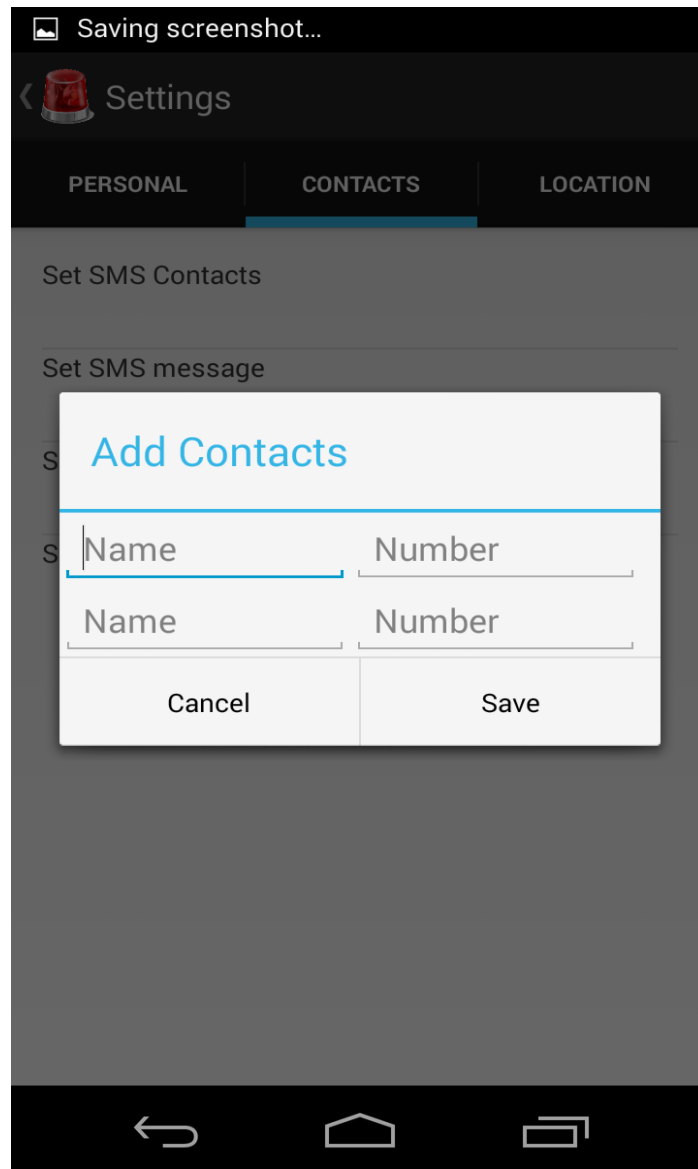


Figure 6-9 SMS contacts screen

The user can set two contacts to will receive the SMS using the above dialog box. The user can either enter a name and a corresponding number or he can click on the number text box to open the contacts app. On the contacts app he can click on a particular contact to add that contact as the name and his primary phone number as the number to send the SMS to.

6.1.10 Set SMS Message

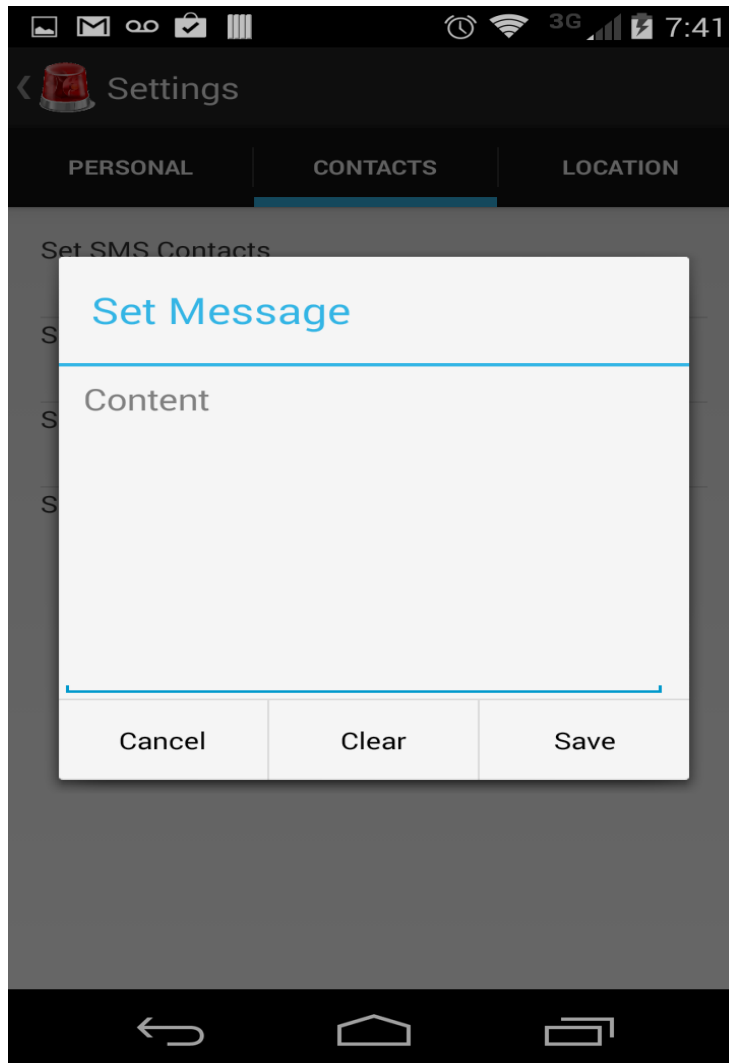


Figure 6-10 SMS message screen

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.

6.1.11 Set Email contacts

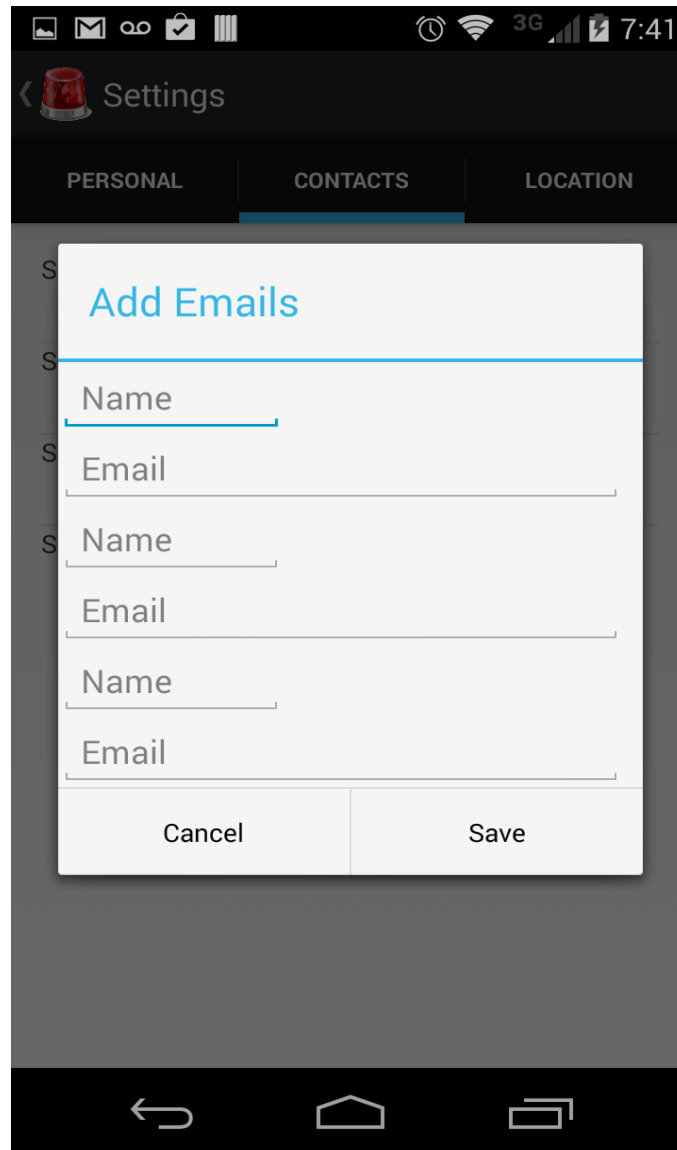


Figure 6-11 Email contacts screen

The user can set three contacts that receive the emails when he presses the panic button. He can either enter the name and the email id of the contacts or he can click on the email text box to show the contacts list. When the user clicks on a particular contact in the list there name and the primary email id is set in the dialog box.

6.1.12 Set Email message

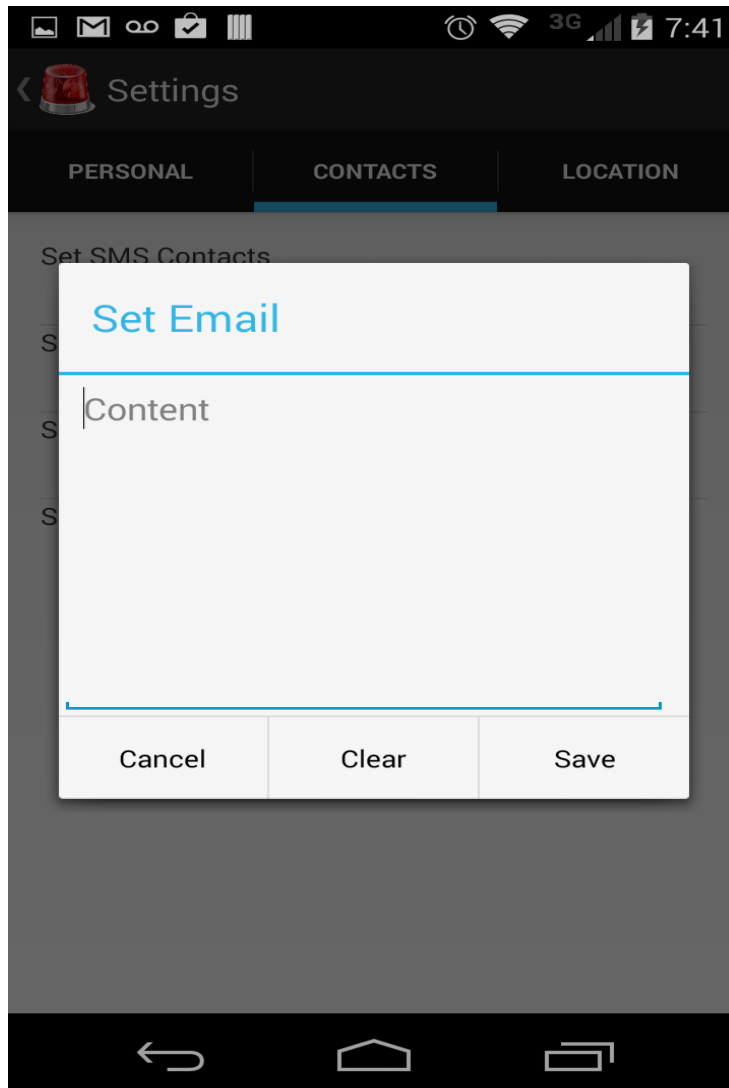


Figure 6-12 Email message screen

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

6.1.13 Location records

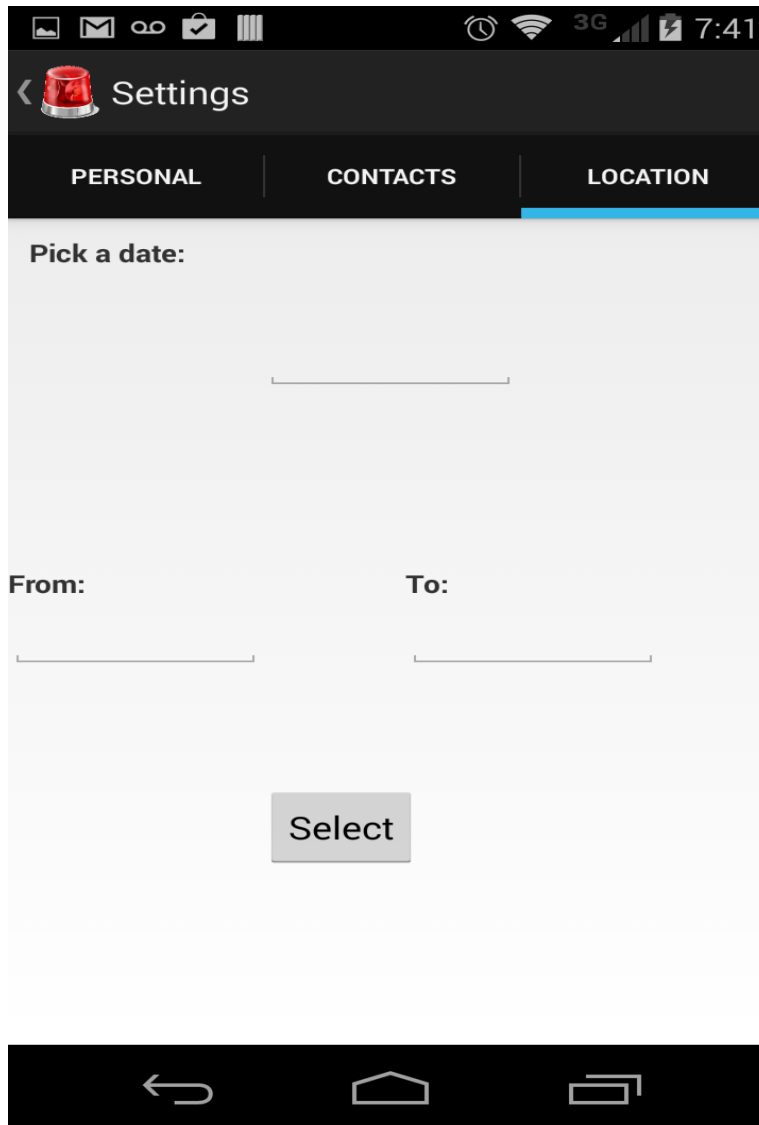


Figure 6-13 Location records screen

The user can see his location history by setting various fields in the above screen. He must provide a date, a start time and an end time to view his location history. The To time has to be greater than the From time.

6.1.14 Date Picker

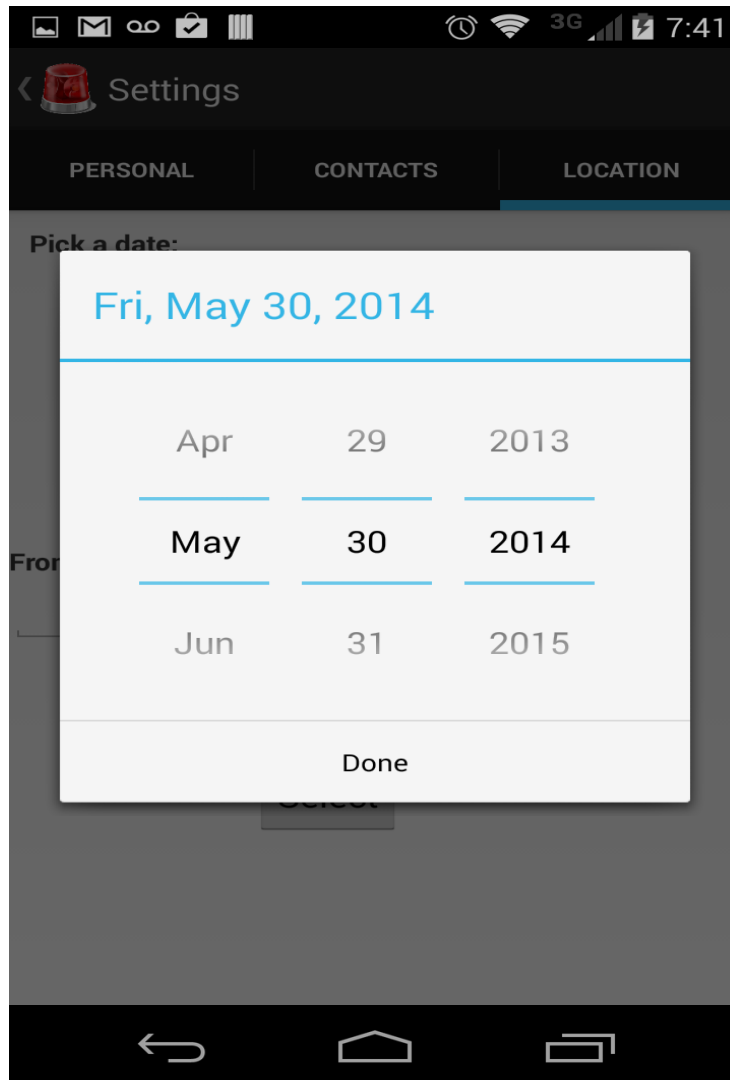


Figure 6-14 Date picker screen

The user is presented with the above date picker fragment to select a date for which he wishes to analyze the location history data.

6.1.15 Time Picker

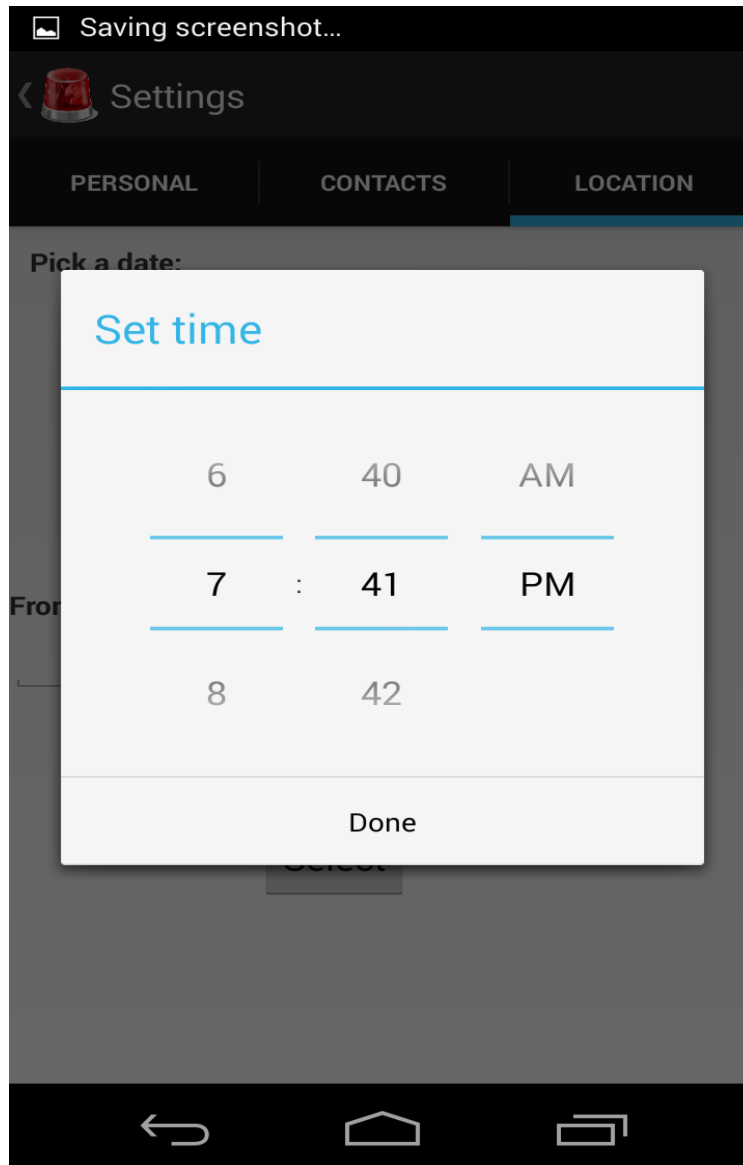
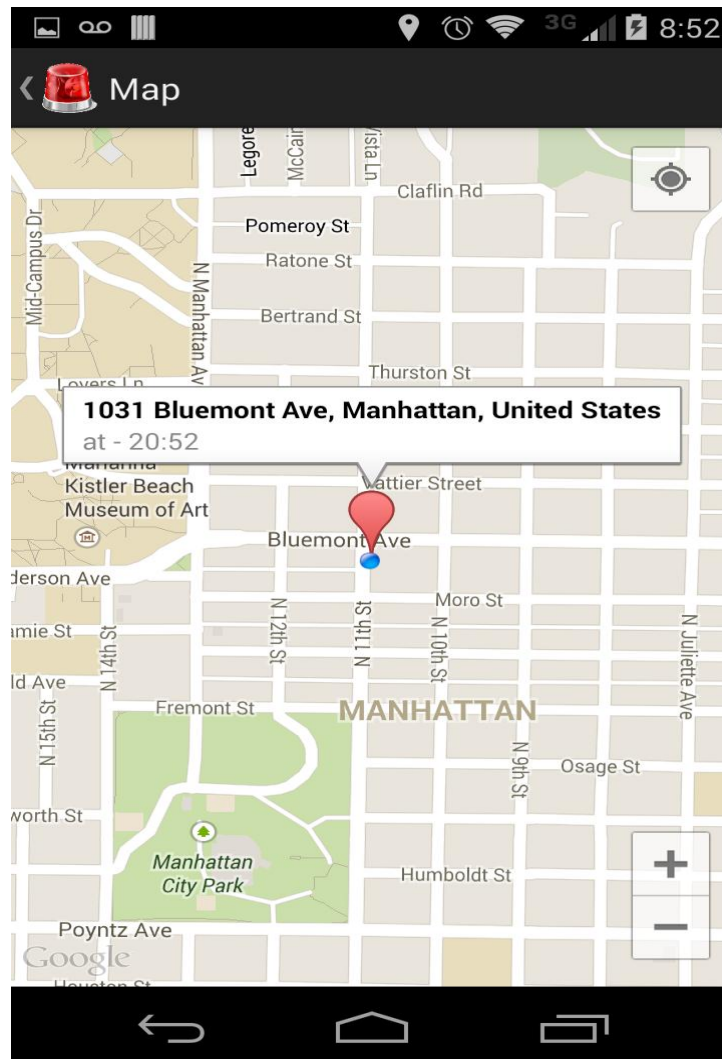


Figure 6-15 Time picker screen

The user is presented with the above time picker fragment to select a time from which he wishes to analyze the location history data.

6.1.16 Map



The user's location history will be displayed on a map as above. His current location will also be displayed by a blue dot on the map. The user can click on any of the markers to display the address and the time at which he was at that particular location

Chapter 7 - 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 provides 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.

7.1 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:

7.1.1 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

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

Table 7-1 Unit test cases - 1

7.1.2 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 shown	Pass
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

Table 7-2 Unit test cases - 2

7.1.3 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 send. 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 email contacts 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 911 button	Switch is enabled	A call should be initiated to 911	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

Table 7-3 Unit test cases - 3

7.1.4 Personal settings screen test cases

S.No	Test case	Pre-condition	Post-condition	Result
1.	Welcome message	User must be logged in	A welcome message with users first name and last name must be shown	Pass
2.	Phone number empty	None	Phone number must be displayed as empty	Pass

3.	Phone number displayed	User must have entered a phone number previously	The same phone number should be displayed in the corresponding text box	Pass
4.	Location tracking unchecked	None	Location tracking checkbox should not be checked	Pass
5.	Location tracking checked	User must have started the location tracking previously	Location tracking checkbox is checked	Pass

Table 7-4 Unit test cases - 4

7.1.5 Contacts setting screen test cases

S.No	Test case	Pre-condition	Post-condition	Result
1.	On clicking Set SMS contacts	None	A dialog box to enter the name and number of two contacts should be shown. Previously entered contacts must be retained	Pass
2.	On clicking Set SMS message	None	A dialog box to enter the message should be shown. Previously entered message should be retained	Pass
3.	On clicking Set Email contacts	None	A dialog box to enter the name and email of three contacts should be shown. Previously	Pass

			entered contacts must be retained	
4.	On clicking Set Email message	None	A dialog box to enter the message should be shown. Previously entered message should be retained	Pass

Table 7-5 Unit test cases - 5

7.1.6 Location records screen test cases

S.No	Test case	Pre-condition	Post-condition	Result
1.	On clicking the Pick a date text box	None	A date picker dialog must appear with current date selected by default	Pass
2.	On clicking the From text box	None	A time picker dialog must appear with current time selected by default	Pass
3.	On clicking the To text box	None	A time picker dialog must appear with current time selected by default	Pass
4.	On clicking the select button when	Some field is not entered	A notification to enter all fields must be shown	Pass
5.	On clicking the select buttons	All fields are entered, To time is earlier than From time	A notification to enter a To time that is greater than the From time should be shown.	Pass

Table 7-6 Unit test cases - 6

7.1.7 Map test cases

S.No	Test case	Pre-condition	Post-condition	Result
1.	Map center	User must have clicked on select button with valid entries on the location history screen	If there are location records for that time, the map should be centered at the first location	Pass
2.	On clicking current location control	None	User must be able to see his current location on the map represented by a blue dot	Pass
3.	On clicking a marker for a location	None	User must be able to see the address and the time he was at that location, for that particular location	Pass

Table 7-7 Unit test cases - 7

7.2 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

Table 7-8 Integration test cases

7.3 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 “Moto G” running Android Kitkat (4.4.2). The response times (worst case times in 10 runs) for different screens in the SOS app are as below:

Screen	Response time (ms)
Main screen (when logging in first time)	2904
Personal Setting screen	1046
Contacts Setting screen	736
Set SMS Contact dialog	437
Set SMS Message	410
Set Email Contact dialog	471
Set Email Message	419
Location history screen	549
Date picker dialog	438
Time picker dialog	453
Map (when 25 locations to be displayed)	3108

Table 7-9 Performance testing

In addition to the above screens the time taken to submit the text messages (to the service provider) and emails (to the PHP script for delivery) was measured to be 1594ms. 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 8 - 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 , emails and making calls to 911 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 9 - 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 911. 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 had 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. The app enabled me in understanding the basic of Android development and learning about SQLite databases, Google Maps API for Android and performance testing the app.

Chapter 10 - 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>.