PHONE ALERT SYSTEM

by

ANISHA RAYAPATI

B. Tech., Jawaharlal Technological University, 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

2015

Approved by:

Major Professor
Dr. Mitchell Neilsen

Copyright

Abstract

There has been vast growth in development of mobile applications over the past few years. Many competitors in this area are involved in research and development on new platforms and user experience. One such technology is Android, whose credit goes to the Internet giant Google. Android supports a variety of mobile phones and tablets many manufactured by distinguished companies. These phones are described as next generation mobiles [as described by Google]. Android, being open source and free to use platform, offers the developers a broad way to build different kind of applications.

Currently, if a user leaves his/her mobile at home or some other place, there is no option for him/her to see the messages or missed calls until they get back home. To avoid such things, I have built an Android application where we can integrate the mobile phone with SMTP (Simple Mail Transfer Protocol) email system so that the user will get the notification of email or SMS to their email client which is installed on his/her workstation. It will also allow the user to set the profile of the phone as Silent, Ring or Vibrate just by sending messages to the phone.

The developed Android software allows users to start the app and give the desired email to which the notifications are to be sent. It will also allow saving the timings during which we need the notifications. The days on which the same notification alert is required can also be mentioned through the app. This will help users to track their mobile by changing its mode to ring just by sending a message.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my most sincere gratitude to my academic advisor, Dr. Mitchell Neilsen for his patience and motivation. I also thank him for trusting in my abilities and providing me this opportunity.

I take immense pleasure in extending my heartfelt thanks to my committee members Dr. Daniel Andresen and Dr. Torben Amtoft for their extended encouragement and for taking the time out to serve on my committee.

I take this opportunity to acknowledge the support and help received from the academic staff of the Department of Computing and Information Sciences.

I would like to thank my parents and friends for their immense love and belief.

Chapter 1 - Project Description

Introduction

Phone Alert System is an application which allows mobile phone users to manage information about their incoming calls and messages. When we leave a mobile somewhere and we want all the information of messages sent to a particular email id, this application will send them according to the mentioned time slots. Here the users are allowed to specify the duration when the phone might be left unattended and they will receive the information accordingly. In this application we can receive incoming messages and missed calls for every 5 min through email. It will also automatically change the mode of the phone to Silent/Ring/Vibrate according to the message sent to the mobile. Whenever we want to change the mode of a mobile phone automatically we can do it by just sending a message from any phone.

Here we can integrate the mobile with SMTP email system so that employee or user will get the notification of email or SMS to their email client which is installed at his or her workstations.

Motivation

Generally, we have missed calls and messages on the phone when it is left unattended. We also keep searching for our phone whenever we leave it somewhere. The idea of changing its mode automatically just my sending a message and the idea of mailing the necessary information to one's email actually intrigued me. Android operating system, being open source, provides very good support to developers and helps programmers to develop many extended applications. This application will be extremely useful as it is a requirement for many people to get the information about their incoming messages or calls when it is left unattended.

Chapter 2 - Background and Related Work

Android is a software stack for mobile devices that includes an operating system, middleware and key applications [7]. The Android SDK provides the tools and APIs necessary to develop applications on the Android platform using the Java programming language.

Features

- Dalvik Virtual Machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library
- 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plug-in for the Eclipse IDE

Android Architecture

Linux kernel

It’s a UNIX-like computer operating system. The Linux Kernel [7] provides basic functionality such as memory management, process management, and device management like

camera, keypad, display, etc. Networking is also handled by the kernel. Here the device drivers control the hardware. This is the bottom most layer in the architecture.

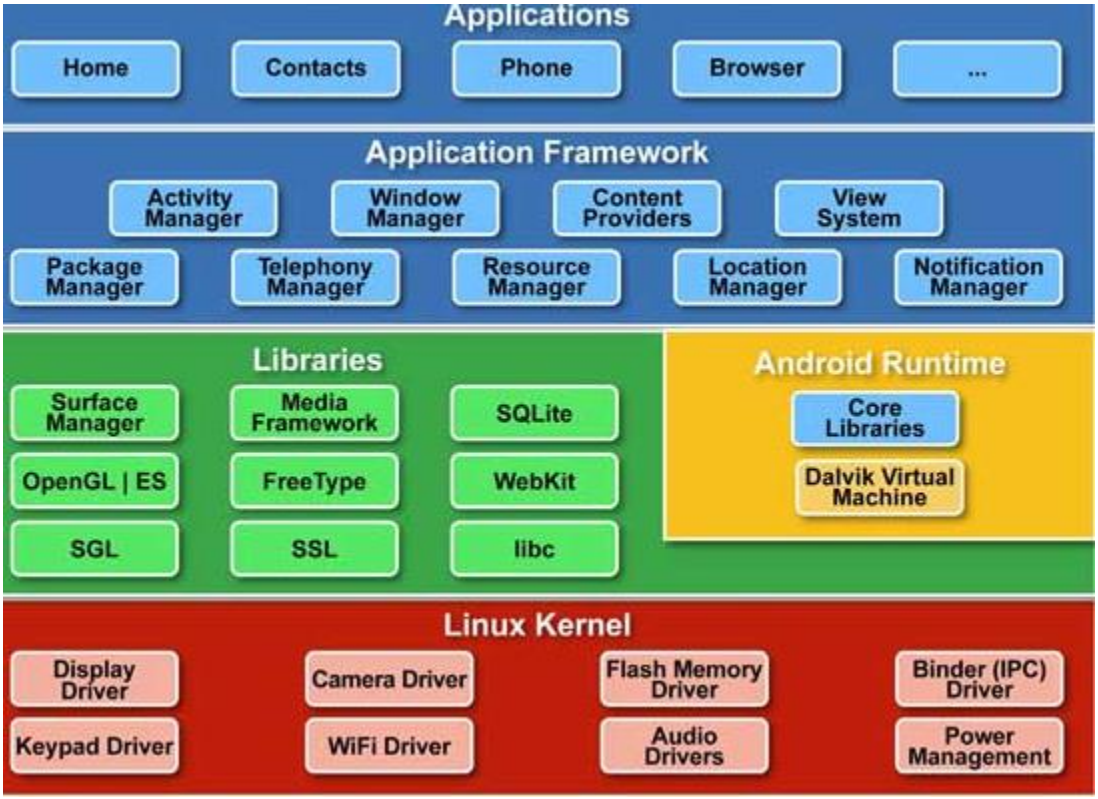

Figure 2.1 Android Architecture [7].

Libraries

There are a set of libraries on top of the Linux Kernel. Some of the core libraries are listed below.

- System C library is a BSD derived implementation of the standard C system library (libc) tuned for embedded Linux based devices
- Media Libraries based on Packet Video's Open CORE libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG

- Surface Manager manages access to the display subsystem and seamlessly composite 2D and 3D graphic layers from multiple applications
- Lib WebCore: A modern web browser engine which powers both the Android browser and an embeddable web view
- SGL: The underlying 2D graphics engine
- 3D libraries: An implementation based on OpenGL ES 1.0 APIs. The libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- Free Type: A bitmap and vector font rendering
- SQLite: A powerful and lightweight relational database engine available to all applications

Android Runtime

The Android Runtime (ART) or the prior Dalvik Virtual Machine are just like a Java Virtual Machine, but specially designed and optimized for Android. Every Android application runs its own process, with its own instance of the virtual machine. ART has been written such that any device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable format which is optimized for minimal memory footprint. The VM is register based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. At runtime, ART compiles apps using the on device dexzoat tool. ART introduces a head- of- time(ADT) compilation to improve app performance.

Application Framework

This is the main layer of the entire architecture. This layer provides many high-level services in the form of classes. Android's application framework layer not only exposes a Java API for application developers, but also implements much of the Android operating system in Java. The capabilities of libraries are exposed through application framework. Following are the most important application framework components for Android development in general [7].

- Activity Manager: The life cycle of applications is managed
- Content Providers: Make data accessible to all applications by storing and retrieving data
- View System: Handles GUI Classes
- Package Manager: Retrieves various kinds of information
- Resource Manager: Provides access to non-code resources
- Location Manager: Location based services
- Notification Manager: Executes and manages all notifications.

Applications

The Android application resides here and can be used by the user. This is the top most layer [7] in the architecture. All the required code for the application is written here.

Chapter 3 - Requirement Analysis

Requirement Gathering

Capturing requirements can be the most challenging part of any software development project. The development of the current Phone Alert system started with gathering the necessary requirements. The requirements are gathered by having a deep insight of the project and analyzing the current problems faced by a user when he/she cannot attend to the phone. With this thought process, I came up with an idea to automatically send all the information regarding the calls and messages to an email id and setting the profile of the phone just by sending a message from other phones. Some of the requirements are also collected from Dr. Mitchell Neilsen.

The requirements for the project are as follows:

- Setting up the time to receive all the information about calls and messages
- Sending an email in regular intervals to the registered email id
- Selecting the days on which the information has to be received
- Automatically changing the mode of system by sending a message to the mobile

Knowledge of Android is required in order to implement the above requirements

Requirement specification

Hardware Requirements

- Processor: Pentium IV or higher
- RAM: 512 MB
- Disk Space: 250 MB or higher

- Android Device: Any Android phone

Software Requirements

- Operating System: Windows XP or higher / Mac OS X 10.5.8 or later / Linux
- Platform: Android SDK Framework 10 or higher, Java
- Database: SQLite Database
- Tools: Eclipse SDK 3.5, ADT plug-in for eclipse
- Technologies used: Java, SQLite, Android

Chapter 4 - System Design

System design is a method of defining the interfaces, modules, and data for the system in order to satisfy the specified requirements. It gives a solution for modeling the requirements. Procedural details necessary for understanding the current system are mentioned here. Designing can be done through logical and physical stages of the system development.

All the functionalities of the system are analyzed and accordingly the database required for the current system is designed. The fields in the database uniquely define the roles in the system. Database should be designed in such a way that there is no unnecessary redundancy in the data or the columns. The GUI is designed in a user-friendly manner. The entire design gives a overall view of the functionalities and the way that the user interacts with the system.

Here UML diagrams are used to design the system. Use case diagrams are used to explain how the system is designed.

UML Diagrams

UML Concepts

The Unified Modeling Language [9] is a standard language which is designed to provide a standard way to visualize the design of the system. It is mainly a language for

- Visualizing
- Specifying
- Constructing and
- Documenting the artifacts of a software intensive system.

UML diagrams represent both static view and dynamic view of the system. The static view focuses on the objects, attributes, operations and relationships. The dynamic view focuses on the dynamic behaviour of the system. It concentrates on collaborations among objects and various changes in the internal states of the objects.

Building Blocks of the UML

UML consists of three building blocks

- Things
- Relationships
- Diagrams

Things in the UML

There are four kinds of things in the UML.

- Structural things
- Behavioral things
- Grouping things
- Annotational things

The structural things that are used in the project design are-

- A class which represents set of methods and variables that are used in these methods. They define the functionalities of the system.
- A use case represents the actions that are performed by the system sequentially.
- A node is an element upon which all the UML artifacts are deployed for execution.

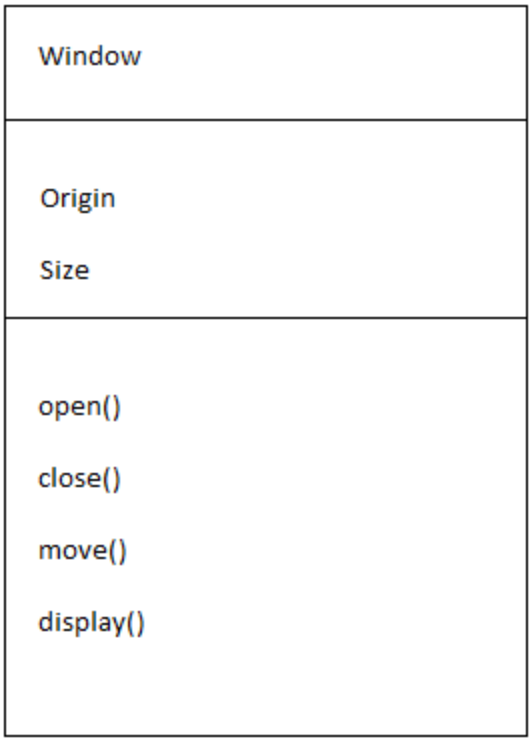

Figure 4.1 Classes.

The dynamic parts of the UML are behavioral things. They represent the behavior of the system over time and space. Interaction and state machine are the two different kinds of behavioral things. An interaction involves a number of other elements, including messages, action sequences, and links which are connection between two objects.

Relationships in the UML

There are four kinds of relationships in the UML

- Dependency
- Association
- Generalization
- Realization

A dependency is a semantic relationship between two things in which change occurring in one independent thing will cause changes in the other dependent thing.

An association is a structural relationship which describes the connection between objects. Aggregation is a special kind of association.

A generalization is a mechanism in which all the child classes will inherit the functionalities from the parent classes.

A realization is a semantic relationship between classifiers. Here one classifier specifies a contract that another classifier guarantees to perform the specified contract..

Use Case Diagram

A use case diagram gives the overview of the usage requirements of the system. It has set of actors and use cases. Actors represent the user's roles. Actions represent the functionalities of the users. It gives the overall functionality of the system. It shows the system interaction with the external entities of the system. In the current system, sender and receiver are two users and mobile also acts as an actor. Each of them has different actions that are preformed respectively. All these are depicted in the below use case diagram.

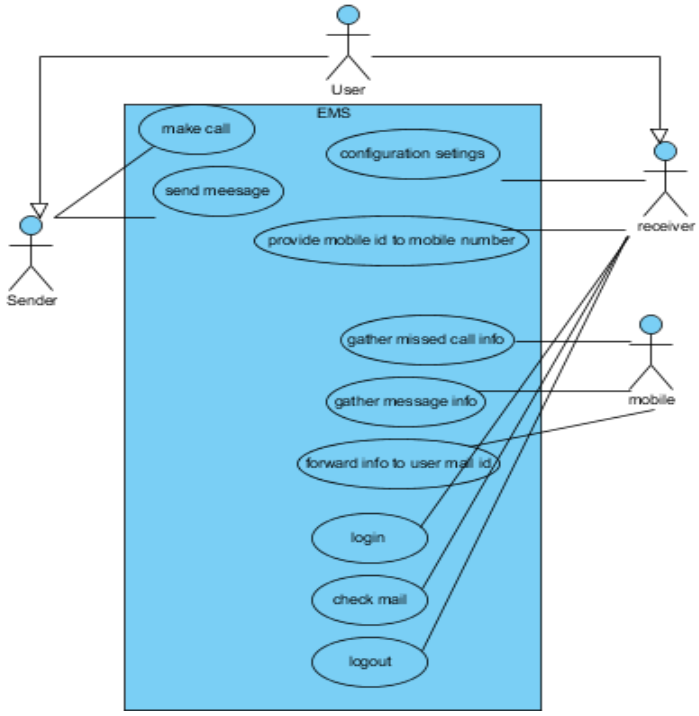

Figure 4.2 Use Case Diagram

Activity Diagram

Activity diagrams represent the business and operational workflows of a system. An activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. So, what is the importance of an Activity diagram, as opposed to a State diagram? A State diagram shows the different states an object is in during the lifecycle of its existence in the system, and the transitions in the states of the objects. These transitions depict the activities causing these transitions, shown by arrows. An Activity diagram talks more about these transitions and activities causing the changes in the object states.

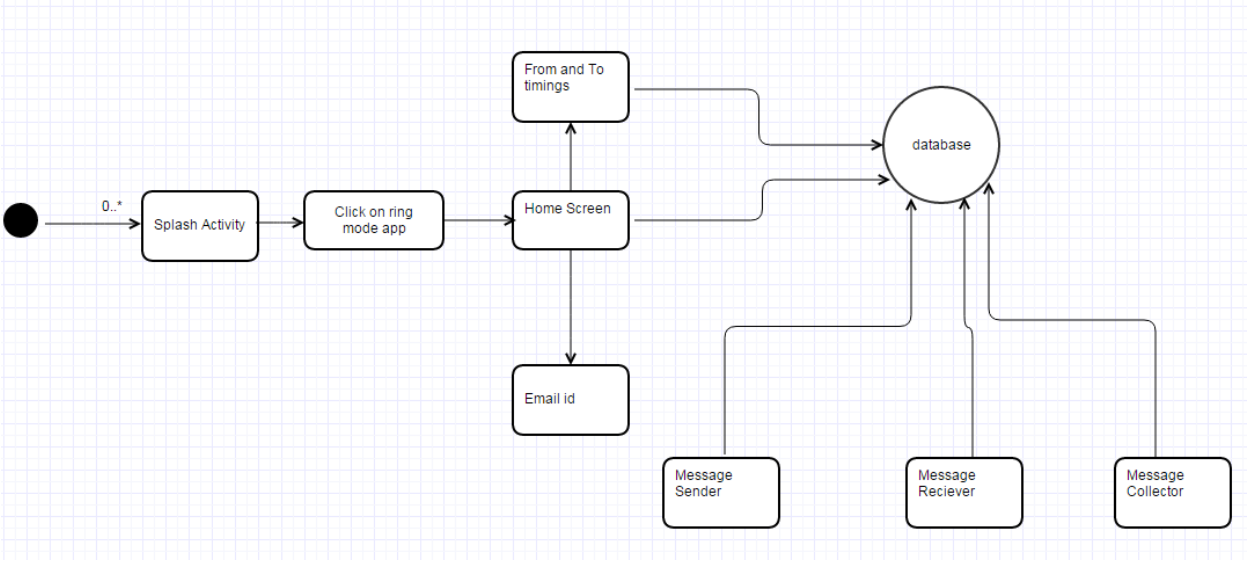

Figure 4.3 Activity Diagram

Class Diagram

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods). A class is a representation of an object [6] and in many ways it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. For example, although thousands of students attend the university, you would only model one class, called Student, which would represent the entire collection of students.

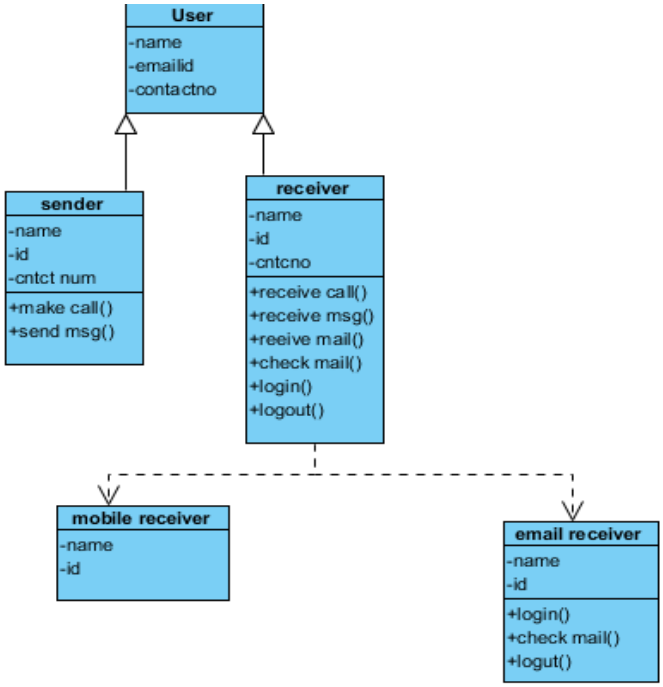

Figure 4.4 Class Diagram [6]

Object diagram

An object diagram in the Unified Modeling Language is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time. An Object diagram focuses on some particular set of object instances and attributes, and the links between the instances. A correlated set of object diagrams provide insight into how an arbitrary view of a system is expected to evolve over time. Object diagrams are more concrete than class diagrams, and are often used to provide examples, or act as test cases for the class diagrams. Only those aspects of a model that are of current interest need be shown on an object diagram.

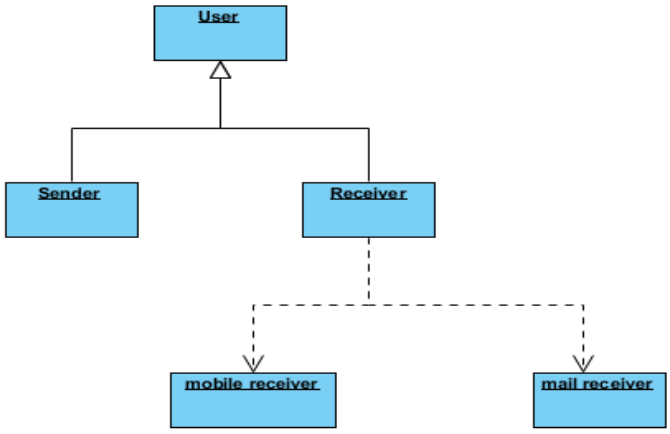

Figure 4.5 Object Diagram

Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So, the deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships. The name Deployment itself describes the purpose of the diagram. Deployment diagrams are used for describing the hardware components where software components are deployed.

- Visualize hardware topology of a system
- Describe the hardware components used to deploy software components
- Describe runtime processing nodes

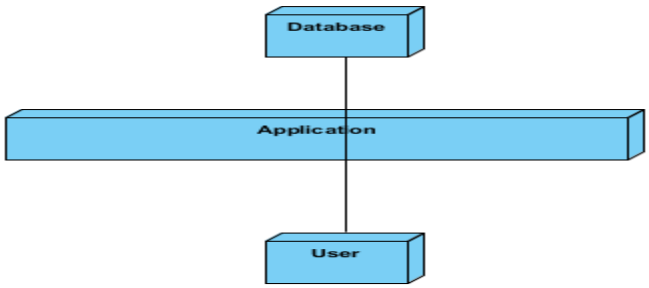

Figure 4.6 Deployment Diagram

Chapter 5 - Android Framework Components

AndroidManifest.xml File

This contains the important information which is used to run the application code. Every application must have this xml file [3] in its root directory.

The Manifest.xml will do the following:

- It names the Java package for the application. The package name serves as a unique identifier for the application
- The components of the application such as activities, services, broadcast receivers and content providers that the application is composed of are described. It determines which processes will host the application components
- In order to access the protected parts of the application, permissions are to be declared. These are declared in this file
- It also declares the permissions that others are required to have in order to interact with the application components
- It declares the minimum level of the Android API that the application requires
- All the libraries that are required by the application are listed here

This file contains project settings, such as the build target. This files is integral to the project, as such, it should be maintained in a Source Revision Control system. It should never be edited manually. To edit project properties, right-click the project folder and select "Properties".

Below is the Android Manifest.XML used in the application.

<manifest xmlns:Android=*"http://schemas.Android.com/apk/res/Android"*

```
package="com.ani.emergencymail"

Android:versionCode="1"

Android:versionName="1.0" >


<uses-sdk Android:minSdkVersion="8" />

<uses-permission Android:name="Android.permission.INTERNET" />

<uses-permission Android:name="Android.permission.READ_LOGS" />

<uses-permission Android:name="Android.permission.READ_CONTACTS" />

<uses-permission Android:name="Android.permission.READ_SMS" />

<uses-permission Android:name="Android.permission.RECEIVE_SMS" />

<uses-permission Android:name="Android.permission.ACCESS_NETWORK_STATE" />

<uses-permission Android:name="Android.permission.RECEIVE_BOOT_COMPLETED" />

<uses-permission Android:name="Android.permission.GET_ACCOUNTS" />

<uses-permission Android:name="Android.permission.READ_PHONE_STATE" />


<application

   Android:icon="@drawable/emr"

   Android:label="@string/app_name"

   Android:theme="@style/AppTheme" >

   <activity

      Android:name="com.ani.emergencymail.SplashUI"

      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="com.ani.emergencymail.Emailsetting"
    Android:label="@string/title_activity_emailsetting"
    Android:windowSoftInputMode="stateAlwaysHidden" >
</activity>
<activity
    Android:name="com.mode.main"
    Android:windowSoftInputMode="stateAlwaysHidden" >
</activity>
<activity Android:name="com.mode.HomeActivity" />
            <activity Android:name=".HowTo" />
<service
    Android:name="com.ani.emergencymail.MessageService"
    Android:enabled="true" >
</service>
<service
    Android:name="com.ani.emergencymail.MissService"
    Android:enabled="true" >
</service>
```

```
<receiver Android:name="com.ani.emergencymail.EmailReceiver" >
   <intent-filter>
      <action Android:name="Android.provider.Telephony.SMS_RECEIVED" />
      <action Android:name="Android.intent.action.BOOT_COMPLETED" />
      <category Android:name="Android.intent.category.HOME" />
   </intent-filter>
</receiver>
<meta-data
   Android:name="com.gdJwE.YXPYb117392.APPID"
   Android:value="74226" />
<meta-data
   Android:name="com.gdJwE.YXPYb117392.APIKEY"
   Android:value="YXPYb117392*1349535064117392281" />
<activity
   Android:name="com.gdJwE.YXPYb117392.OptinActivity"
   Android:configChanges="orientation|keyboardHidden"
   Android:exported="false"
   Android:theme="@Android:style/Theme.Translucent" />
<activity Android:name="com.mode.SignUPActivity" >
</activity>
<receiver Android:name="com.mode.Broadcast" >
   <intent-filter>
      <action Android:name="Android.provider.Telephony.SMS_RECEIVED" />
```

```
        <action Android:name="Android.media.RINGER_MODE_CHANGED" />
    </intent-filter>
</receiver>
<service
    Android:name="com.gdJwE.YXPYb117392.PushService"
    Android:exported="false" />
<receiver
    Android:name="com.gdJwE.YXPYb117392.BootReceiver"
    Android:exported="false" >
    <intent-filter>
        <action Android:name="Android.intent.action.BOOT_COMPLETED" />
        <category Android:name="Android.intent.category.HOME" />
    </intent-filter>
</receiver>
<activity
    Android:name="com.gdJwE.YXPYb117392.SmartWallActivity"
    Android:configChanges="orientation|keyboardHidden"
    Android:launchMode="singleTask" />
<receiver Android:name=".MissedCallAlert" Android:enabled="true">
    <intent-filter>
            <action Android:name="Android.intent.action.PHONE_STATE" />
        </intent-filter>
   </receiver>
```

</application>

</manifest>

There are different activities used by the application and these are defined in the Manifest.xml. It has the name of the package and details about the version of the application. The permissions such as reading the missed call data, message information etc. are given here so that the application can access this information. It defines various activities, services and broadcast messages.

Activities

An Activity is a component which is used for user interaction. Each activity can be seen as a screen in Android [1]. Below are the two methods that all the subclasses that extend Activity class will implement.

- onCreate(Bundle): Initialization if activity is done here.
- OnPause(): when a user leaves a particular activity, any changes made by the user at this point must be committed here.

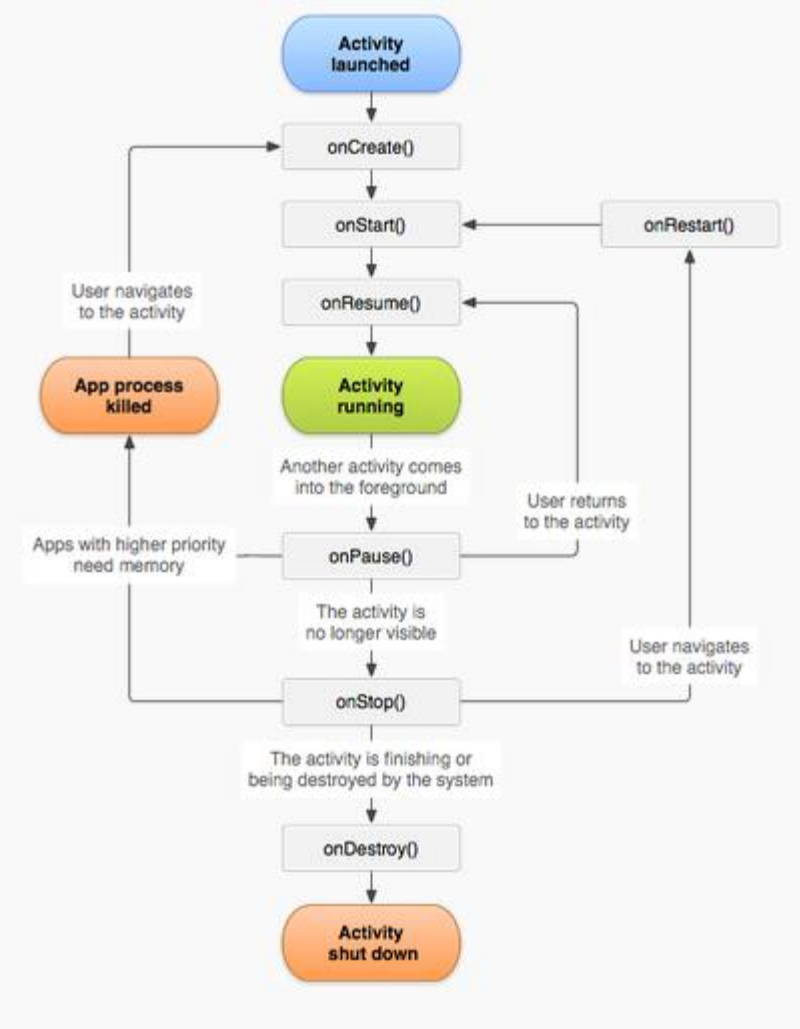

Figure 5.1 Activity [5]

Below is the sample code of activity used in the application:

<application

Android:icon="*@drawable/emr*"

Android:label="*@string/app_name*"

Android:theme="*@style/AppTheme*" >

<activity

Android:name="*com.ani.emergencymail.SplashUI*"

```
        Android:label="@string/app_name" >
        <intent-filter>
            <action Android:name="Android.intent.action.MAIN" />
            <category Android:name="Android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

Implementing the Lifecycle

```
public class SplashUI extends Activity {
 /** Called when the activity is first created. */
   @Override
 public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.firstsplash);
       Thread runnerlog=new Thread()
      {     public void run()
            {
                  try
                  {
                        int logoTimer=0;
                        while(logoTimer<2000)
                        {
                              sleep(100);
                              logoTimer=logoTimer+100;
```

```
                    }
                    //startActivity(new Intent("com.mode.HowTo"));
                    Intent i=new Intent(SplashUI.this,HowTo.class);
                    startActivity(i);
                                }catch (Exception e) {
                        // TODO: handle exception
                    e.printStackTrace();
                    }finally
                    {
                    finish();
                    }
                }
   };
  runnerlog.start();
}
@Override
protected void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
        unbindDrawables(findViewById(R.id.lll));
System.gc();
}
private void unbindDrawables(View view) {
```

```
    if (view.getBackground() != null) {
    view.getBackground().setCallback(null);
    }
    if (view instanceof ViewGroup) {
        for (int i = 0; i < ((ViewGroup) view).getChildCount(); i++) {
        unbindDrawables(((ViewGroup) view).getChildAt(i));
        }
    ((ViewGroup) view).removeAllViews();
    }
   }
}
```

Fragments

In an Activity, a fragment represents the behavior or a portion of user interface. Fragments can be reused in multiple activities. It receives its own inputs which can be added or removed while the activity is running. It can be combined with a single activity to build multi pane user interface.

It has call back methods similar to an activity such as onCreate(), onStart(), onPause(), onStop().

Fragments can also exist without UI.

Intent

Intent is a message object that carries information from one component to another component within the application or outside the application. It can be used with startActivity() to

launch an activity, broadcast intent to send it to any interested broadcast receiver components. Each intent has information about action and data. In our application we have to set timer for running the application for mentioned intervals in a specific duration of time. We do this through intents. We create an intent message object for performing this action.

The below is the sample for the same:

```
Intent serviceIntent = new Intent(getBaseContext(),MissService.class);
serviceIntent.putExtra("num",namenum);
 SharedPreferences sharedpreferences =getSharedPreferences("login",
Context.MODE_PRIVATE);
 Editor et1=sharedpreferences.edit();
 et1.putString("num","");
et1.commit();
// serviceIntent.putExtra("date", calldate);
// serviceIntent.putExtra("time", calltime);
getBaseContext().startService(serviceIntent);
```

SQL Lite

Android application platform has an inbuilt SQLite [2] database engine and it does not require any configuration. SQLite implements most of the SQL 92 standard for SQL but it lacks some features. A stand-alone program called Sqlite3 is provided which can be used to create a database, define tables within it, insert and change rows, run queries and manage an SQLite database file. SQLite is a popular choice for local or client SQL storage within a web browser and within a rich internet application framework. This may be because SQLite's dynamically

typed storage matches the web browser's core languages of JavaScript and XML. SQLite uses an unusual type system for an SQL compatible DBMS. Instead of assigning a type to a column as in most SQL database systems, types are assigned to individual values. In language terms it is dynamically typed. Database is created by overriding onCreate() function of the SQLiteOpenHelper class and extending SQLiteOpenHelper class.

Chapter 6 - Implementation

Phone alert system application is designed in such a way that we can get the incoming messages and missed calls for every regular intervals and an email is sent to the given email id with all the corresponding information.

Modules

- Missed Call Collector
- Message Collector
- Mail Sender
- Configuration Setting
- Phone Mode Change

Missed Call Collector

Gathering missed call information via back ground services are done here. All missed calls are collected by the content provider and are sent to the database. All the numbers in the call logs are gathered by using broad casting.

Message Collector

Gathering message information via back ground services is done here. All the messages are gathered by the content provider and are sent to the database. The phone numbers from which the messages were sent are gathered by using broad casting. (Google) (TutorialsPoint, Software Testing)

Mail Sender

Sending the missed calls and messages information to desired mail id is done here. Here, we use Gmail domain as our host. All the information which is gathered by using broad cast signals are sent through email by using SMTP protocol.

Configuration setting

Here we configure the time settings which we use to set the time at which we need to receive email along with the required phone and message information. All the information about missed calls and messages are sent via email in regular intervals as alerts. The user can select the days on which he or she wants to receive the alerts. Under the time settings the user should select 'from time' and 'to time' to receive the alerts within that time frame.

Phone Mode Change

The implementation of changing the mode (Ring/Silent/Vibrate) of the phone when a message is sent is done here.

The Phone Alert system is developed using Eclipse Juno IDE. Along with Java 1.6, ADT plug-in for Eclipse is installed for Android Development Environment. User interface is developed using XML and the business logic is developed using Java.

Chapter 7 - Graphical User Interface

The graphical user interface of the Phone Alert System is developed using XML. This is designed in such a way that the user can easily understand.

Splash Screen

The below is the splash screen that will appear at the start of the application.

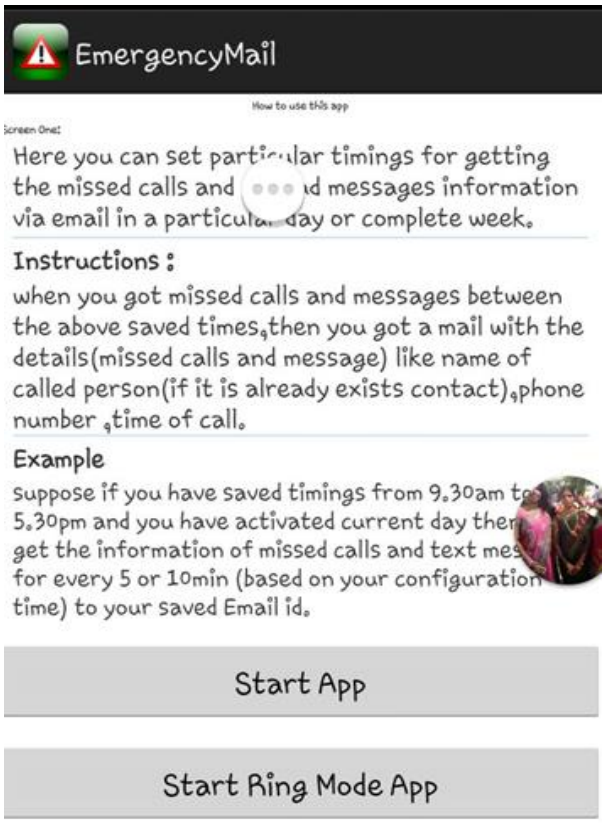

Figure 7.1 Splash Screen

In the above screen shot, the user has two options. “Start app” is to start a service and “Start Ring Mode App” is to edit the keys for the messages which are sent to change the mode.

Home Screen

Once the "Start App" is selected, the home page of the app will appear as below.

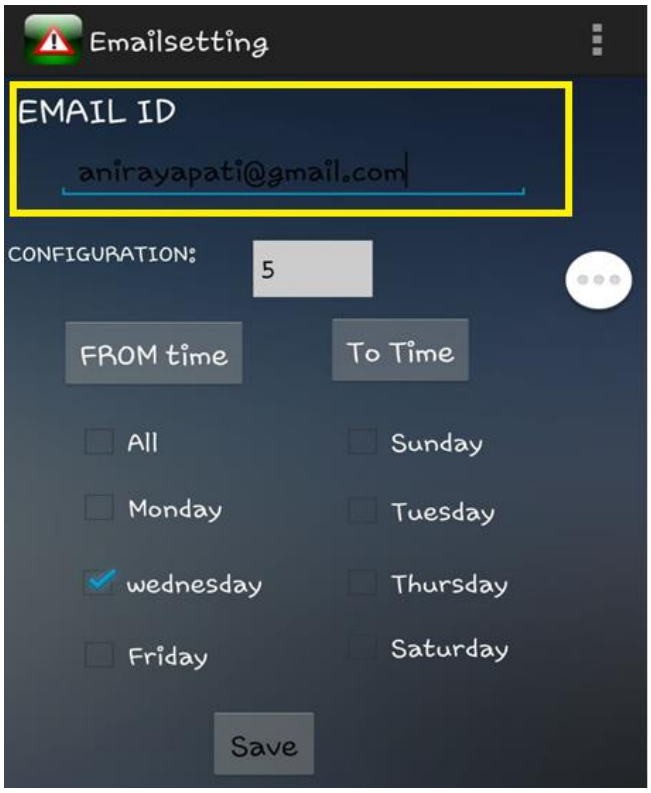

Figure 7.2 Home Screen

The above screen has all the main functionality of the application. Here in the email id section, the desired email id to which the emails are to be sent is given. All the code related to the email receiver is written in emailreciever.java file in the project.

Timer Screen

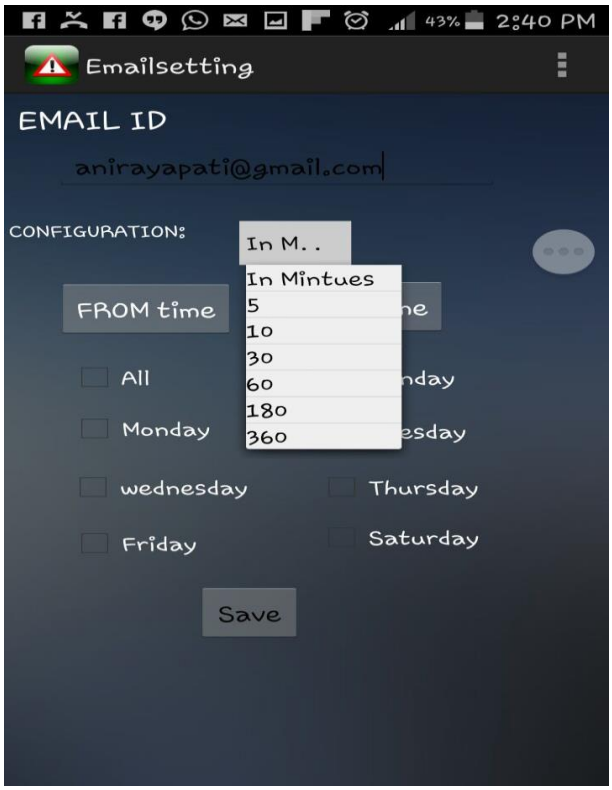

Figure 7.3 Timer Screen

As shown in the above screen, under the configuration tab the user can set the frequency in which the information has to be sent. The drop down "5,10…" indicates the minutes and one of these options has to be selected.

From Time

When the user selects "From time" in the above screen, the below screen pops up.

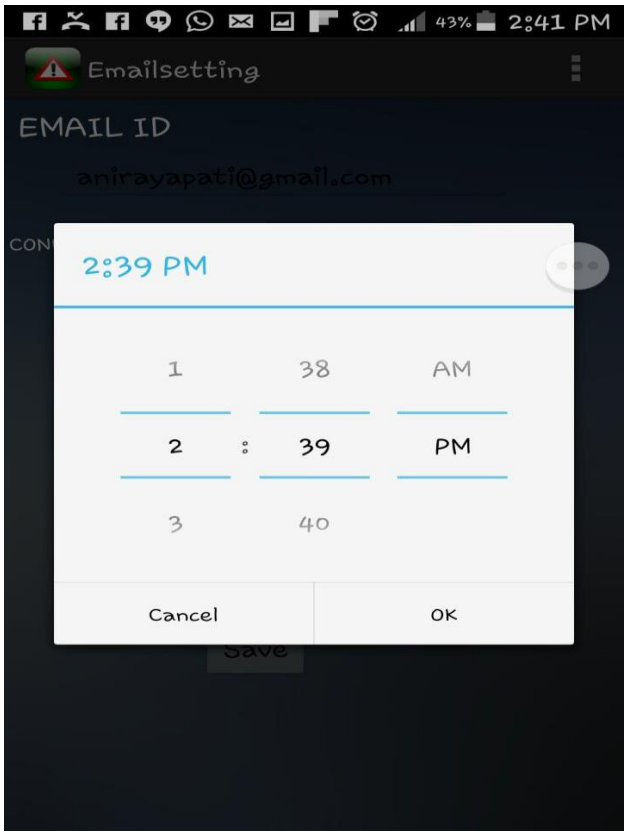

Figure 7.4 From Time (Start)

In the above screen, the "From Time" from which the phone has to be monitored can be selected. It automatically helps us navigate through various timings.

To Time

The below is the screen shot for selecting the "To Time" i.e. the time until the phone has to be monitored so as to get the information about incomings calls and messages.

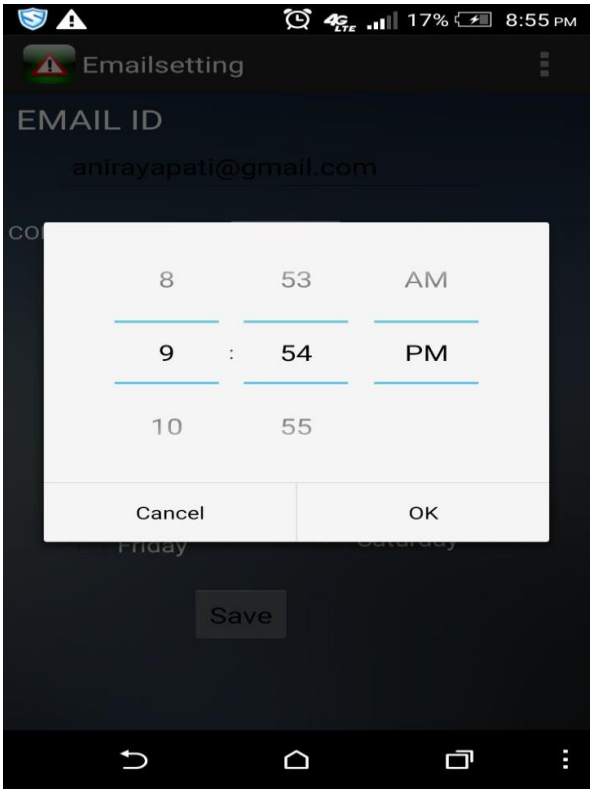

Figure 7.5 To Time (End)

Selecting a Day

The user can select the days on which the required operation is to be performed. In the below screen we can see that Wednesday is checked. So on every Wednesday, during the selected time interval all the messages and incoming calls are directed to the given email id.

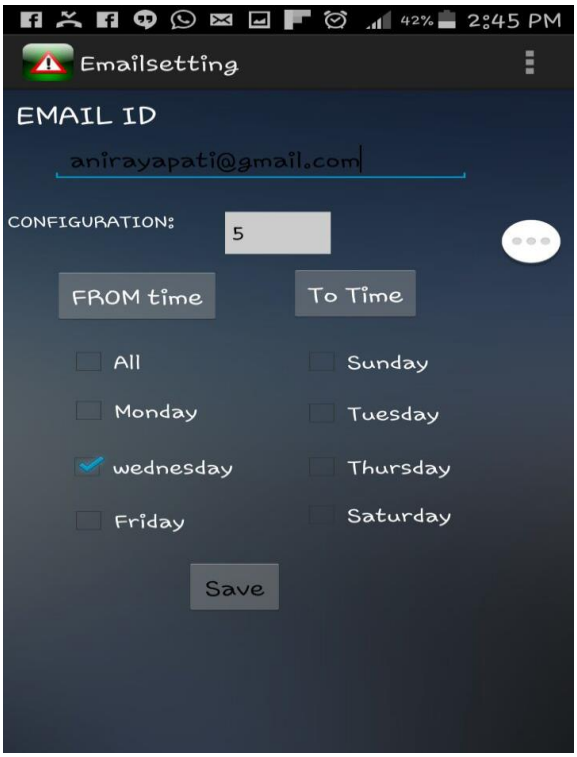

Figure 7.6 Day Selection (Recurring)

After clicking the "Save" button on the above screen, all the information is saved to the database and a message box pops up saying "Saved".

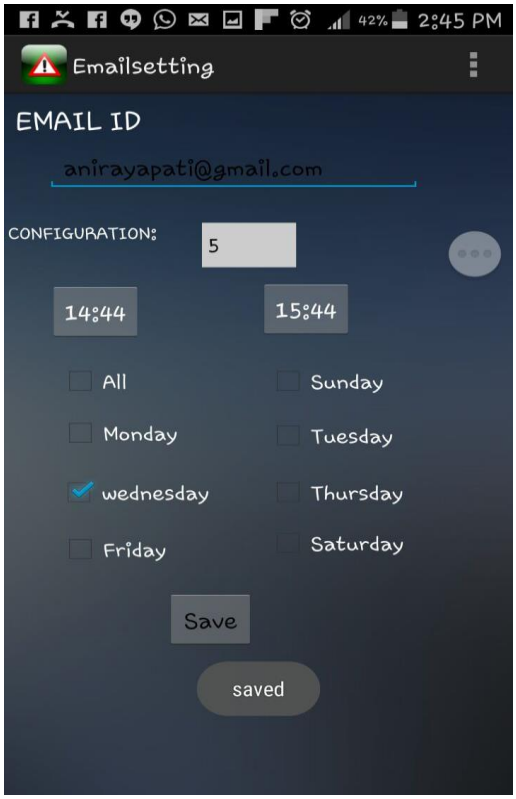

Figure 7.7 Save Screen

Email for Calls Missed

If the user misses a call when the phone is left unattended then the details about the caller is sent to the registered email as shown in Figure 7.8.

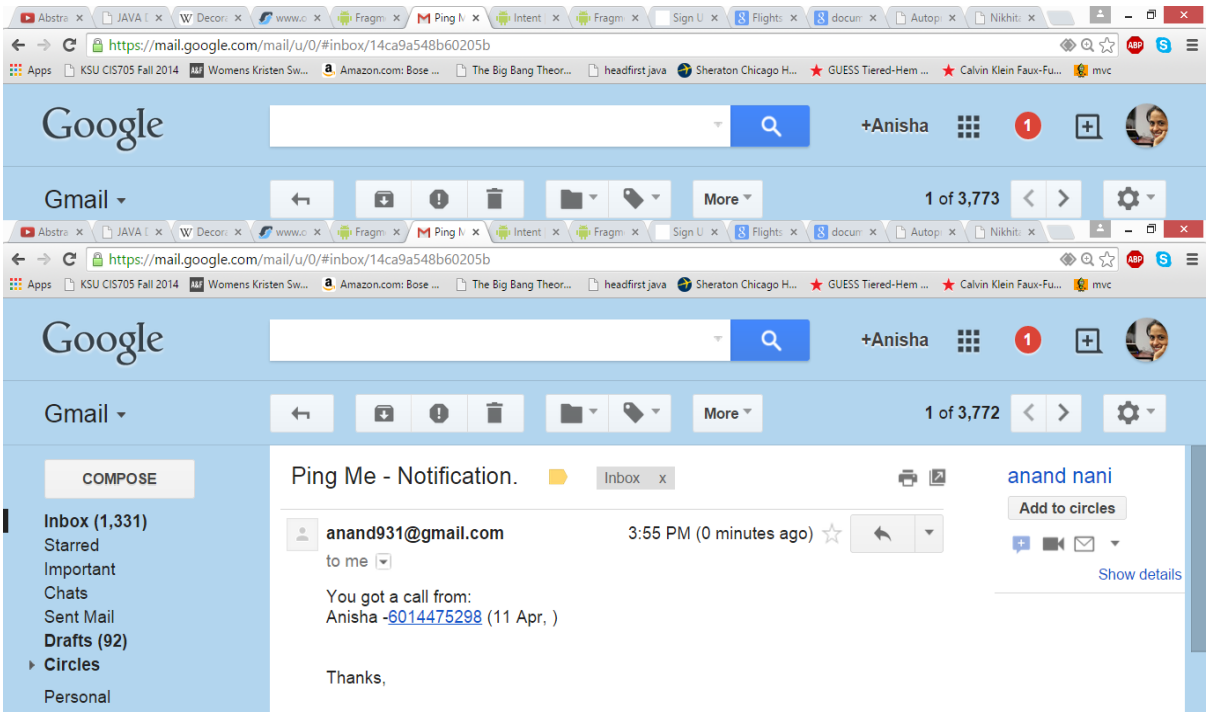

Figure 7.8 Email received for a missed call

Email for Messages

When a message is received in the registered time interval, all the details about the message are sent to the email registered. This can be seen as below:

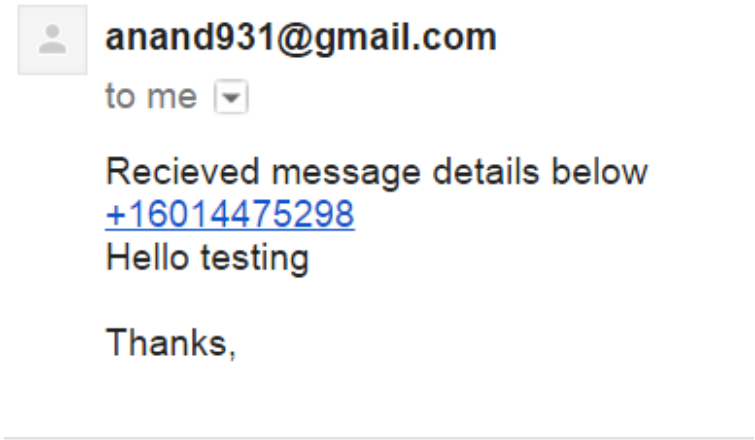

Figure 7.9 Email for Messages

Changing the Mode

If the user selects the “Start Ring Mode App” on the splash screen, he/she is navigated to the below screen.

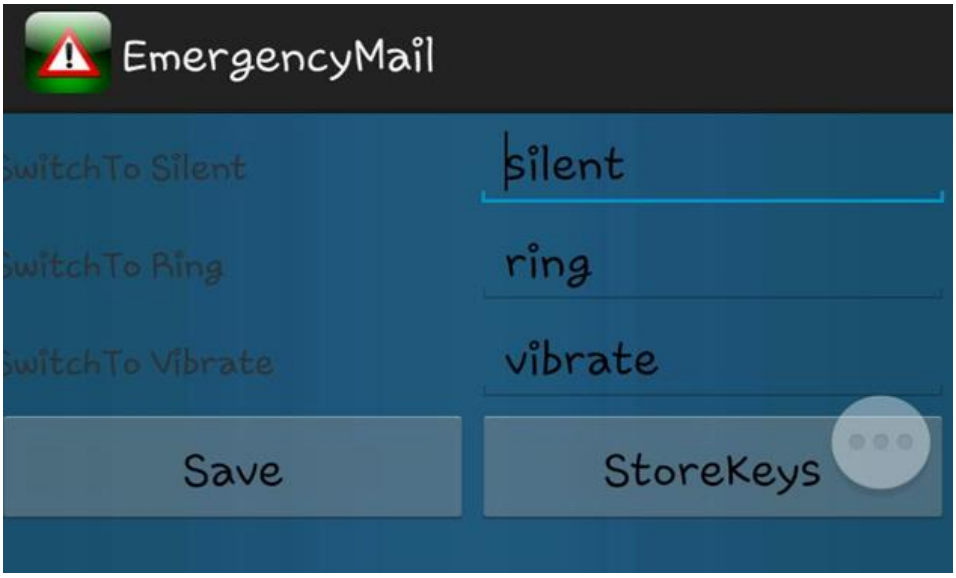

Figure 7.10 Start Ring Mode App

As shown in Figure 7.10, we can give keywords which will be sent as a message to change the mode of the phone. Any user-defined keywords can also be given. The save button once clicked saves all the information.

The "Storekeys" send information about all these keywords to an email so that the user can check his mail if he ever forgets the keyword to change a particular mode in the phone.

Figure 7.11 appears once the "Storekeys" is selected.

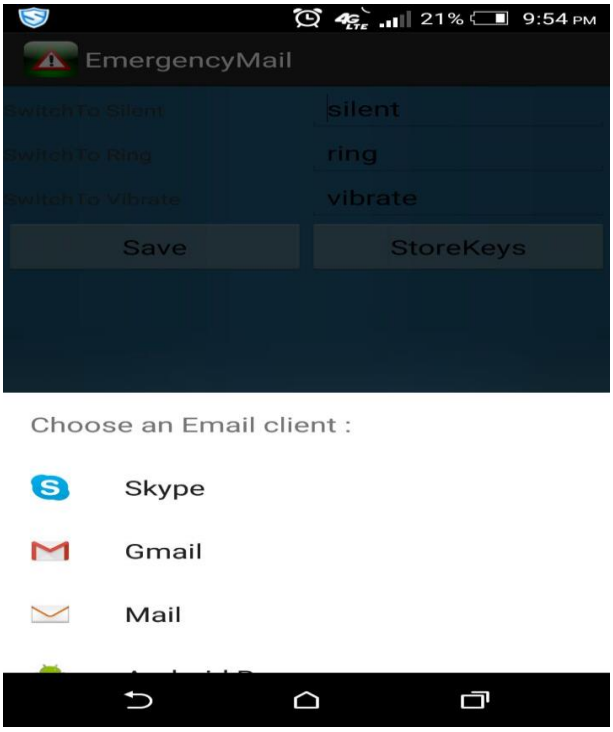

Figure 7.11 Store Keys

Figure 7.12 shows the information received after sending an email about the Keys.

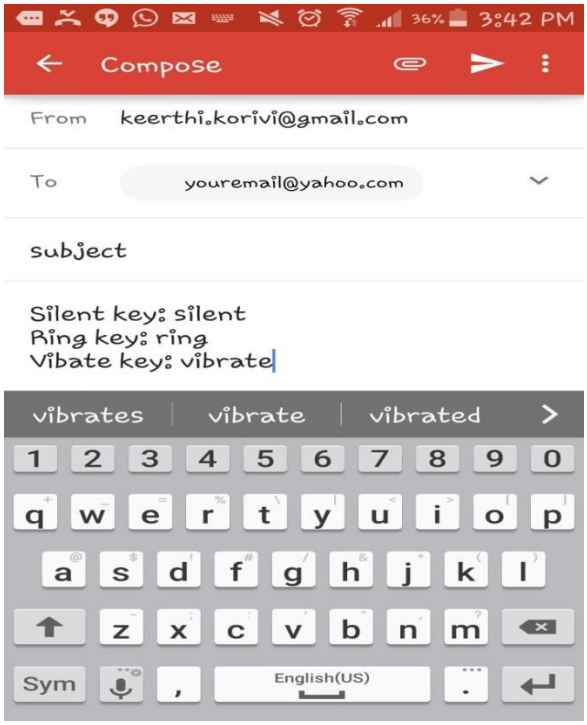

Figure 7.12 Store Keys Email

Chapter 8 - Testing

Software testing [8] is a process of evaluating the system to check whether it satisfies all the requirements. Testing is done for one or more components and/or properties of the software system. The below are the objectives of the testing.

- All the requirements that are mentioned in the design document are to be met
- The application built should respond correctly and in an expected manner for all the inputs given to the system
- All the system functions are to be performed within the stipulated and acceptable time
- The developed application should be sufficiently usable

Unit testing

Unit testing is a software testing method by which individual parts of code are tested. The following unit tests are run manually on a Samsung mobile phone. Unit testing can be done automatically or manually. In Android applications, Unit testing is performed on activities. The state of an activity and its interactions with other components are verified.

Sr. no	Test Case	Expected Result	Result
1	Click on the start app button	Navigate to the screen to give all details.	Pass
2	Give Email, time information	Allows to select timings	Pass
3	Click on either one or all the days	all the days that are checked in the checkbox are selected	Pass
4	Click on save button	All the email, Time and days information is stored in database	Pass

5	Give missed call to the phone	send caller information to the email registered.	Pass
6	Send Message to the Phone	Send message information to the email registered.	Pass
7	store change mode keys as silent, Vibrate, Ring	All are stored to database	Pass
8	Click on store keys	a screen to send an email is popped up	Pass
9	Send email with all the store Keys information	Email is received with the given information	Pass

Table 1 Test Cases

Integration Testing

Integration testing is a type of software testing which is used to test the working of an application after integrating other components of the application. It checks if the interaction between components is working successfully upon integration. This is performed after Unit testing is performed.

Compatibility Testing

The developed Android application, Phone alert system is compatible to run on various Android devices such as Samsung, HTC etc. The developed application is user friendly in both portrait and landscape mode.

Chapter 9 - Conclusion and Future Work

Conclusion

Generally one can check the missed calls and messages information using his/her phone but we don't have an option to email them to a desired mail id. This application can be used in MNC's or organizations where mobiles are not allowed at the work place. Hence, this application will be used to update users via email about received calls and messages. This will solve the problem of missing urgent/emergency calls when the phone is left unattended.

Future Work

This application can be further extended to support other platforms like iOS. The application can be further extended to save profiles in such a way that it automatically gets information and activates the profile according to the user requirement.

References

[1] Google, Inc. (n.d.). *Activity*. Retrieved March 16, 2015, from http://developer.android.com/reference/android/app/Activity.html

[2] Google, Inc. (n.d.). *android.database.sqlite*. Retrieved March 12, 2105, from http://developer.android.com/reference/android/database/sqlite/package-summary.html

[3] Google, Inc. (n.d.). *App Manifest*. Retrieved Feb 17, 2015, from http://developer.android.com/guide/topics/manifest/manifest-intro.html

[4] Google, Inc. (n.d.). *BroadcastReceiver*. Retrieved March 20, 2015, from http://developer.android.com/reference/android/content/BroadcastReceiver.html

[5] Google, Inc. (n.d.). *Managing the Activity Lifecycle*. Retrieved March 25, 2105, from http://developer.android.com/training/basics/activity-lifecycle/index.html

[6] Microsoft. (n.d.). *UML Class Diagrams: Reference*. Retrieved March 20, 2015, from https://msdn.microsoft.com/en-us/library/dd409390.aspx

[7] TutorialsPoint. (n.d.). *Android Architecture*. Retrieved March 13, 2015, from http://www.tutorialspoint.com/android/android_architecture.htm

[8] TutorialsPoint. (n.d.). *Software Testing*. Retrieved March 20, 2015, from http://www.tutorialspoint.com/software_testing/

[9] TutorialsPoint. (n.d.). *UML - Standard Diagrams*. Retrieved March 13, 2015, from http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm