RESEARCH ARTICLE OPEN ACCESS

Remotely Control of Android Based OS Mobile Application

Dhananjay Singh
M.Tech(SE)
Gautam Buddha university
Greater Noida – India

ABSTRACT

Modern hand held devices such as smart phones and PDAs have become increasingly powerful in recent years. Dramatic breakthroughs in processing power along with the number of extra features included in these devices have opened the doors to a wide range of commercial possibilities. In particular, most cell phones regularly include cameras, processors comparable to PCs from only a few years ago, and internet access. However, even with all these added abilities, there are few applications that allow much passing of the environmental information and location based services.

As mobile devices become more like PCs they will come to replace objects we tend to carry around such as checkbooks, credit cards, cameras, planners, mp3 players, etc. In short, we will be using them to accomplish our daily tasks. It could be that a person misplaces his/her mobile so he couldn't make use of these services so to find misplaced phone, application named Remotely Control of Android based OS Mobile Applicationwill be helpful.

The prime objective of "Remotely Control of Android based OS Mobile Application" Application is to create a fully-fledged Android application which could help in locate the mobile when it get misplaced or lost. This allows the user to get the GPS location, changes the mobile profile from Silent mode to Ringer mode, changes the ring volume to maximum and plays a music tone while misplaced, remotely using another phone having SMS functionality i.e. by sending certain code words to misplaced or lost mobile phone.

Keywords:- GPS location, changing its profile mode, changing its ringtone volume, Play a music tone, Android limitation, Application behavior, Android SDK Manager Etc.

I. NTRODUCTION

The purpose of this document is to present a detailed description of the Remotely Control of Android based OS Mobile Application. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate. It will provide the technical, functional and non-functional features, required to develop an android mobile application named Remotely Control of Android based OS Mobile Application. The entire application is designed to provide user flexibility for finding the mobile.

What Is Android?

Android is a Linux-based operating system for mobile devices such as smart phones and tablet computers. It is developed by the Open Handset Alliance led by Google. The android SDK provides the tools and APIs necessary to begin developing applications on the android platform using the Java programming language.

Recent releases:

1: Gingerbread refined the user interface, improved the soft keyboard and copy/paste features, better native code support (which improves gaming performance), added SIP support (VoIP calls), and added support for Near Field Communication.

2: Honeycomb was a tablet-oriented release which supports larger screen devices and introduces many new user interface features, support for multicore processors, hardware acceleration for graphics and full system encryption. The first device featuring this version, the Motorola Xoom tablet, went on sale in February 2011.

3: Honeycomb, released in May 2011, and added support for extra input devices, USB host mode for transferring information directly from cameras and other devices, and the Google Movies and Books apps.

4: Honeycomb, released in July 2011, added optimization for a broader range of screen sizes, new "zoom-to-fill" screen compatibility mode, loading media files directly from SD card, and an extended screen support API. Huawei Media Pad is the first 7 inch tablet to use this version

5: 4.0 Ice Cream Sandwich, announced on October 19, 2011, brought Honeycomb features to smart phones and added new features including facial recognition unlock, network data usage monitoring and control, unified social networking contacts, photography enhancements, offline email searching, app folders, and information sharing using NFC. Android 4.0.4 Ice Cream Sandwich is the latest Android version that is available to phones. The source code of Android 4.0.1 was released on November 14, 2011

System Overview: This section provides a detail overview of the Remotely Control of Android based OS Mobile Application.

II. APPLICATION PERSPECTIVE

This application provides facilities like, when the person has misplaced/lost the mobile he/she can send message with some predefined codes through any other mobile. These codes will activate the application. The application would then provide GPS location, change its profile mode from silent to general, enhance the ringtones volume & play a ringtone. It also allows us to change the predefined codes in its settings option.

Operating environment

1. Windows operating system for development.
2. Android Mobile Operating System for deployment.

Characteristics

Mobile application is for the security of mobile in terms of lost or misplaced.

a) Providing GPS location.
b) Changing its profile mode.
c) Changing its ringtone volume.
d) Play a music tone.

Developer – The role of a developer is to maintain the application. It is assumed
That the user is adept in API, Android and Java.

Functional Requirements:

This section of the document illustrates different functions provided by the application:

Description:

1. **GPS location** In case of the android mobile gets stolen or lost, the application Remotely Control of Android based OS Mobile Application will help in getting the GPS location of mobile on the mobile number with which message has been sent on stolen mobile.

-Default Keyword: #gpslocation#

2. Silent to Ringer mode

In case of mobile is on silent mode, we can change the profile of the android mobile device to ringer mode from silent mode, by sending the code word in message via another mobile device.

-Default Keyword: #ringmode#

3. Play tone

This feature will simply allow the application to play a music tone in media player when its gets the default code word in message

-Default Keyword: #playtone#

4. Volume The application on getting the coded message it will change the volume to maximum

-Default Keyword: #incvolume#

Rationale:

Once the application gets installed, he/she has can use the features of application by means of sending predefined code via message through any mobile device. When the user sends the coded message to the android device, the application sends back the current location of device.

Requirements:

In order to use this application, the User should have a mobile phone with Android Operating System on it.

- -min API level of use 10 i.e. 2.3.3(Ginger Bread)
- -screens MUST be at least 2.5 inches in physical diagonal size, density must be at least 100 dpi.
- -Device implementations MUST have at least 128MB of memory available to the kernel and user space
- -278 MB RAM
- -512 MHz processor

III. SOFTWARE REQUIREMENT

Eclipse IDE for java developers (3.7.2 Indigo)

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy and Scheme. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others. The initial codebase originated from VisualAge. The Eclipse SDK (which includes the Java development tools) is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the Eclipse Public License, Eclipse SDK is free and open source software. It was one of the first IDEs to run under GNU Class path and it runs without issues under IcedTea.

Downloading Link: http://www.eclipse.org/downloads/

Android Development Tool plugins for Eclipse

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, add packages based on the Android Framework API, debug your applications using the Android SDK tools, and even export signed (or unsigned) .apk files in order to distribute your application. Developing in Eclipse with ADT is highly recommended and is the fastest way to get started. With the guided project setup it provides, as well as tools integration, custom XML editors, and debug output pane, ADT gives you an incredible boost in developing Android applications. Many of the tools that you can start or run from the command line are integrated into ADT. They include:

- **Traceview:** Allows you to profile your program's execution (**Window > Open Perspective >Traceview**).
- **Android:** Provides access to the Android SDK Manager and AVD Manager. Other android features such as creating or updating projects (application and library) are integrated throughout the Eclipse IDE.
- **Hierarchy Viewer:** Allows you to visualize your application's view hierarchy to find inefficiencies (**Window > Open Perspective > Hierarchy Viewer**).
- **Pixel Perfect:** Allows you to closely examine your UI to help with designing and building. (**Window > Open Perspective > Pixel Perfect**).
- **DDMS:** Provides debugging features including: screen capturing, thread and heap information, and logcat (**Window > Open Perspective > DDMS**).
- **adb:** Provides access to a device from your development system. Some features of adb are integrated into ADT such as project installation (Eclipse run menu), file transfer, device enumeration, and logcat (DDMS). We must access the more advanced features of adb, such as shell commands, from the command line.
- **ProGuard:** Allows code obfuscation, shrinking, and optimization. ADT integrates ProGuard as part of the build, if you enable it.
- **Android SDK Tools**
- Contains tools for debugging and testing your application and other utility tools. These tools are installed with the Android SDK starter package and receive periodic updates. We can access these tools in the <sdk>/tools/ directory of our SDK. To learn more about them, see SDK Tools in the developer guide.

Downloading Link: http://dl.google.com/android/installer_r16-windows.exe

SDK Manager

The Android SDK Manager is the tool that you use to install and upgrade SDK packages in our development environment. We can launch the Android SDK Manager in one of the following ways.

AVD Manager

An Android Virtual Device (AVD) is an emulator configuration that lets our model an actual device by defining hardware and software options to be emulated by the Android Emulator. The easiest way to create an AVD is to use the graphical AVD Manager, which we launch from Eclipse by clicking **Window >AVD Manager**. We can also start the AVD Manager from the Command line by calling the android tool with the avd options, from the**<sdk>/tools/** directory.

We can also create AVDs on the command line by passing the android tool options. For more information on how to create AVDs in this manner, see Managing Virtual Devices from the Command Line.

An AVD consists of:

- A hardware profile: Defines the hardware features of the virtual device. For example, you can define whether the device has a camera, whether it uses a physical QWERTY keyboard or a dialing pad, how much memory it has, and so on.
- A mapping to a system image: We can define what version of the Android platform will run on the virtual device. We can choose a version of the standard Android platform or the system image packaged with an SDK add-on.
- Other options: We can specify the emulator skin we want to use with the AVD, which lets we control the screen dimensions, appearance, and so on. we can also specify the emulated SD card to use with the AVD.
- A dedicated storage area on our development machine: the device's user data (installed applications, settings, and so on) and emulated SD card are stored in this area.

We can create as many AVDs as you need, based on the types of device you want to model. To thoroughly test your application, you should create an AVD for each general device configuration (for example, different screen sizes and platform versions) with which our application is compatible and test our application on each one. Keep these points in mind when you are selecting a system image target for our AVD.

- The API Level of the target is important, because our application will not be able to run on a system image whose API Level is less than that required by our application, as specified in the minSdk Version attribute of the application's manifest file. For more information about the relationship between system API Level and application minSdk Version, see Specifying Minimum System API Version.
- We should create at least one AVD that uses a target whose API Level is greater than that required by your application, because it allows we to test the forward-compatibility of your application. Forward-compatibility testing ensures that, when users who have downloaded your application receive a system update, our application will continue to function normally.
- If our application declares a uses-library element in its manifest file, the application can only run on a system image in which that external library is present. If we want to run our application on an emulator, create an AVD that includes the required library. Usually, we must create such an AVD using an Add-on component for the AVD's platform (for example, the Google APIs Add-on contains the Google Maps library). To learn how to manage AVDs using a graphical tool, read Managing AVDs with AVD Manager. To learn how to manage AVDs on the command line, read Managing AVDs from the Command Line.

Android SDK Platforms (2.3.3 API Level 10 GingerBread)

The platform tools are typically updated every time we install a new SDK platform. Each update of the platform tools is backward compatible with older platforms. Usually, we directly use only one of the platform tools— the Android Debug Bridge (adb). Android Debug Bridge is a versatile tool that lets we manage the state of an emulator instance or Android-powered device. We can also use it to install an Android application (.apk) file on a device. The other platform tools, such as aidl, aapt, dexdump, and dx, are typically called by the Android build tools or Android Development Tools (ADT), so we rarely need to invoke these tools directly. As a general rule, we should rely on the build tools or the ADT plugin to call them as needed.

Java Development Toolkit 5 or 6 The Java Development Kit (JDK) is an Oracle Corporation product aimed at Java developers. Since the introduction of Java, it has been by far the most widely used Java Software Development Kit (SDK). On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007; Sun contributed the source code to the OpenJDK. The JDK is a development environment for building applications, applets, and components using the Java programming language. The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform. These tools are designed to be used from the command line. Except for the applet viewer, these tools do not provide a graphical user interface. Downloading Link: http://java.sun.com/javase/downloads/index.jsp

Non Functional Requirements

- Better network coverage of operator
- High processing capability of Processor and
- high RAM so GPS function will take less time to give accurate location

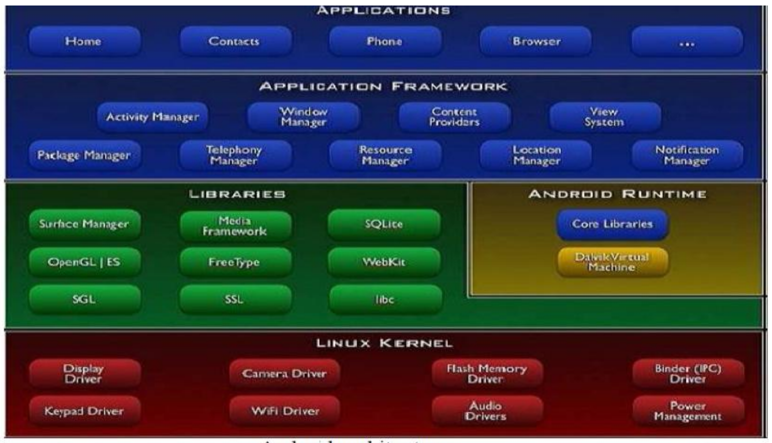
Other Requirements

The SIM in the android mobile must have minimum some balance so that the application can send message back in case of providing the current GPS location.

Android Framework

Android is the mobile phone platform led by Google's Open Handset Allowance (OHA). Android has a unique security model in which the user is in complete control of the device. It is an open source platform based on Linux. All applications are written in Java and compiled into acustom byte-code (DEX). Each application executes in its own process with its own instance of the Dalvik virtual machine interpreter.

IV. ANDROID ARCHITECTURE



Android architecture

Application behavior:

Every application in Android runs as a separate process with a unique UID, unlike a desktop computer where all the applications run with the same UID. The UID of an application in android protects its data. Programs cannot typically read or write

Each other's data, and sharing between applications must be done explicitly. Due to this feature, a compromise such as a buffer overflow attack is restricted to the launch another program that will run with the launching application's UID. For a developer to run an application on the Android phone, application needs to be signed. Developers can generate self-signed certificates and use this for code signing. Code signing is done to enable developers to update their own applications without creating complicated interfaces and permissions.

Application components:

Applications are comprised of components. Components interact using Intent messages. Recipient components assert their desire to receive Intent messages by Defining Intent filters.

There are four types of components used to construct applications:

1. **Activity** components interact with the user via the touch screen and keypad. Only one activity is active at a time, and processing is suspended for all other activities.

2. **Service** components provide for background processing when an application's activity leaves focus and another GUI application comes in the foreground.

3. **Broadcast** receiver components provide a general mechanism for asynchronous event notifications. The receivers receive Intent messages that are implicitly addressed with action strings; for instance, dialing a number is associated with the action outgoing call action.

4. **Content provider** components are the preferred method for sharing data between applications. These APIs implement an SQL-like interface; however, the

backend implementation is up to the application developer.

Application level security framework:

Applications need approval to do things their user might object to, such as sending SMS messages, using the contacts database, or using the camera. To keep track of what the application is permitted to do, Android maintains manifest permissions that are enforced by the middleware reference monitor. The permission label is a unique text string that can be defined by the OS as well as by a third-party developer. These permissions indicate what resources and interfaces are available to the application at run-time. An example of a permission is READ_CONTACTS, which permits the application to read the user's address book. In addition to reading and writing data, permissions allow applications to access system services such as dialing a number without prompting the user or taking complete control of the screen and obscuring the status bar. A developer should specify all permissions that his or her application requires in the AndroidManifest.xml file; however, it is not necessary that all permissions be granted. When the application is getting installed, the user has the choice to decide whether or not to trust the software based on the application's promised features and the permissions required. These permissions are different from file permissions. Once an application is installed, its permissions cannot be changed. The fewer permissions an application needs, the more comfortable the user should feel installing the application. Permissions have a protection level.

The four protection levels are outlined are:-

Normal: Permissions for application features with minor consequences such as VIBRATE, which lets applications vibrate the device. Suitable for granting rights not generally of keen interest to users; users can review but may not be explicitly warned.

Dangerous: Permissions such as WRITE_SETTINGS or SEND_SMS are dangerous as they could be used to reconfigure the device or incur tolls. This level marks permissions in which the users will be interested or be potentially surprised. Android will warn users about the need for these permissions on install.

Signature: These permissions can be granted only to other applications signed with the same key as this program. This allows secure coordination without publishing a public interface.

Signature Or System:

Same as Signature except that programs on the system image also qualify for access. This allows programs on custom Android systems to also get the permission. This protection is to help integrate system builds and won't be typically used by developers. The permission label policy is used to protect applications from each other and also various components within an application. In the mobile phone environment, it is difficult for the operating system to manage access control policies of hundreds of unknown applications. Therefore, Android simplifies this by having the developers define their permission labels to access their interfaces. By doing so, the developer does not need to know about existing and future applications; permission labels allow the developer to indirectly influence security decisions.

Files and preferences:

Android uses UNIX-style file permissions. Each application has its own area on the file system that it owns. This is similar to programs having a home directory to go along with their User IDs. This feature is limited only to the internal phone memory and not the external memory. The standard way for applications to expose their private data to other applications is through content providers.

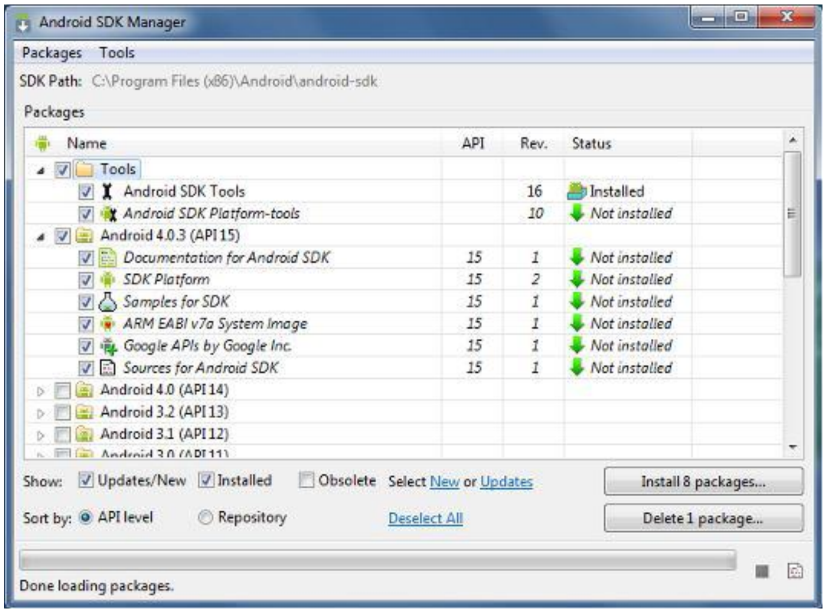
Android limitation:

The current security policy of Android works on a static level only during installation to identify whether the application is permitted all the requested permissions from the user. Once the permission is granted, there is no way to govern to whom these rights are given or how they are later exercised. Permissions are asserted as vague suggestions as to what kinds of protections the application desires. One must place good faith in the user and the OS to make good choices about permissions granted to the application which, in many cases, may not be the absolute best Choice. Due to the above architecture, Android system libraries have limited ability to control installed third-party applications that can be granted permissions to use their interfaces. This implies that there is no control to restrict an installed application based on its signatures. Further, it is not possible to define the desirable configurations of an installed third-party application such as the minimum version and the set of permissions it is allowed or disallowed. This implies that Android applications built with the right set of permissions protect the system from malicious applications but provides severely limited infrastructure for applications to protect themselves.

Android SDK Manager. On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.

On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then execute android To download packages, use the graphical UI of the Android SDK Manager to browse the SDK repository and select new or updated packages (see figure 1). The Android SDK Manager installs the selected packages in our SDK environment. The Android SDK Manager's **Available Packages** panel, which shows the SDK packages that are available for us to download into your environment.

Android SDK manager



SDK Path: C:\Program Files (x86)\Android\android-sdk

Packages

Name	API	Rev.	Status
☑ Tools			
☑ Android SDK Tools		16	Installed
☑ Android SDK Platform-tools		10	Not installed
☑ Android 4.0.3 (API 15)			
☑ Documentation for Android SDK	15	1	Not installed
☑ SDK Platform	15	2	Not installed
☑ Samples for SDK	15	1	Not installed
☑ ARM EABI v7a System Image	15	1	Not installed
☑ Google APIs by Google Inc.	15	1	Not installed
☑ Sources for Android SDK	15	1	Not installed
☐ Android 4.0 (API 14)			
☐ Android 3.2 (API 13)			
☐ Android 3.1 (API 12)			
☐ Android 3.0 (API 11)			

Show: ☑ Updates/New ☑ Installed ☐ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Done loading packages.

V. AVAILABLE PACKAGES

By default, there are two repositories of packages for our SDK: Android Repository and Third party Add-ons.

The Android Repository offers these types of packages:

- **SDK Tools** — Contains tools for debugging and testing our application and other utility tools. These tools are installed with the Android SDK starter package and receive periodic updates. We can access these tools in the <sdk>/tools/ directory of our SDK. To learn more about them, see SDK Tools in the developer guide.
- **SDK Platform-tools** — Contains platform-dependent tools for developing and debugging our application. These tools support the latest features of the Android platform and are typically updated only when a new platform becomes available. We can access these tools in the <sdk>/platform-tools/ directory. To learn more about them, see Platform Tools in the developer guide.
- **Android platforms** — An SDK platform is available for every production Android platform deployable to Android-powered devices. Each SDK platform package includes a fully compliant Android library, system image, sample code, and emulator skins. To learn more about a specific platform, see the list of platforms that appears under the section "Downloadable SDK Packages" on the left part of this page.
- **USB Driver for Windows** (Windows only) — Contains driver files that we can install on our Windows computer, so that we can run and debug our applications on an actual device we do not need the USB driver unless we plan to debug your application on an actual Android-powered device. If we develop on Mac OS X or Linux, we do not need a special driver to debug our application on an

Android-powered device. See Using Hardware Devices for more information about developing on a real device.

- **Samples** — Contains the sample code and apps available for each Android development platform. If we are just getting started with Android development, make sure to download the samples to our SDK.

Recommended Packages:

- **Documentation** — contains a local copy of the latest multi version documentation for the Android framework API.
- The Third party Add-ons provide packages that allow us to create a development environment using a specific Android external library (such as the Google Maps library) or a customized (but fully compliant) Android system image. We can add additional Add-on repositories by clicking **Add Add-on Site**.

Recommended Packages:

The SDK repository contains a range of packages that we can download. Use the table below to determine which packages we need, based on whether we want to set up a basic, recommended, or full development environment:

Environment	SDK Package	Comments
	SDK Tools	If we have just installed the SDK starter package, then we already have the latest version of this package. The SDK Tools package is required to develop an Android application. Make sure we keep this up to date.
Basic	**SDK Platform Tools**	This includes more tools that are required for Application development. These tools are platform-dependent and typically update only when a new SDK platform is made available, in order to support new features in the platform. These tools are always backward compatible with older platforms, but you must be sure that you have the latest version of these tools when we install a new SDK platform.
	SDK Platform	We need to download at least one platform into our environment, so that we will be able to compile our application and set up an Android Virtual Device (AVD) to run it on (in the emulator). To start with, just download the latest version of the platform. Later, if we plan to publish your application, we will want to download other platforms as well, so that we can test your application on the full range of Android platform versions that our application supports.

Testing: This module is involving both Software validation and verification.

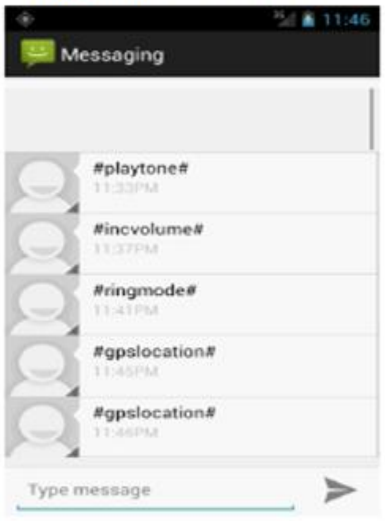
Software Verification and Validation:

Verification	**Validation**
Are you building it right?	Are you building the right thing?
Ensure that the software system meets all the functionality.	Ensure that functionalities meet the intended behaviour.
Verification takes place first and includes the checking for documentation, code etc.	Validation occurs after verification and mainly involves the checking of the overall product.
Done by developers.	Done by Testers.

Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.	Have dynamic activities as it includes executing the software against the requirements.

VI. RESULT

final thread of messages sent between 9129188354 & 7388585997


11:46

Messaging

#playtone#
11:33PM

#incvolume#
11:37PM

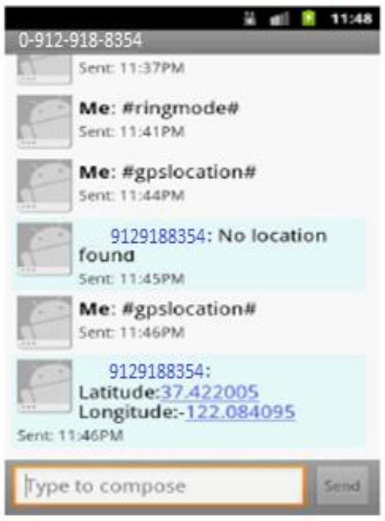
#ringmode#
11:41PM

#gpslocation#
11:45PM

#gpslocation#
11:46PM

Type message

7388585997


11:48

0-912-918-8354

Sent: 11:37PM

Me: #ringmode#
Sent: 11:41PM

Me: #gpslocation#
Sent: 11:44PM

9129188354: No location found
Sent: 11:45PM

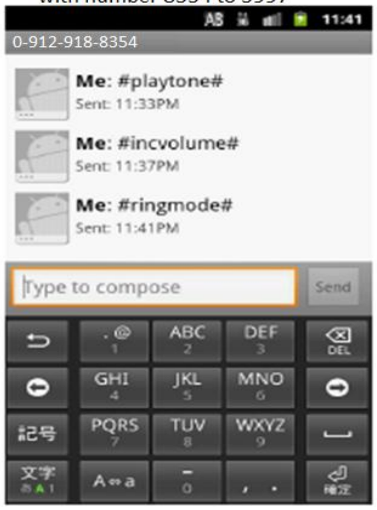
Me: #gpslocation#
Sent: 11:46PM

9129188354:
Latitude:37.422005
Longitude:-122.084095
Sent: 11:46PM

Type to compose

Send

Composing and sending tophone with number 8354 to 5997


11:41

0-912-918-8354

Me: #playtone#
Sent: 11:33PM

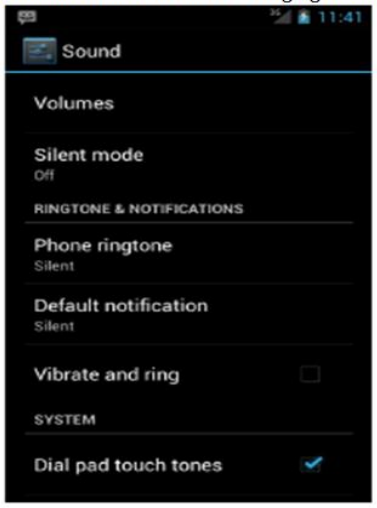
Me: #incvolume#
Sent: 11:37PM

Me: #ringmode#
Sent: 11:41PM

Type to compose

Send

phone 8354 received message from 5997 and switches 8354from silent to ringing mood


11:41

Sound

Volumes

Silent mode
Off

RINGTONE & NOTIFICATIONS

Phone ringtone
Silent

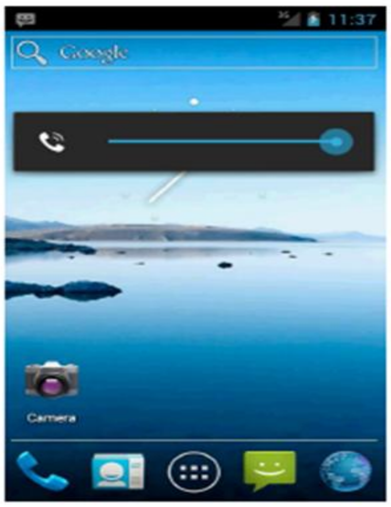
Default notification
Silent

Vibrate and ring

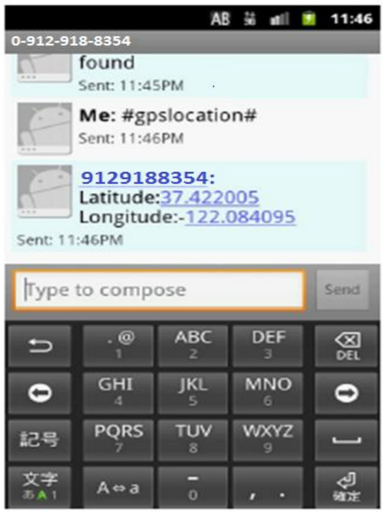
SYSTEM

Dial pad touch tones ✓

Increasing volume i.e. on level 7 (max) after receiving message from 8354



GPS Location as recieved from 8354 on 5997



0-912-918-8354

found
Sent: 11:45PM

Me: #gpslocation#
Sent: 11:46PM

9129188354:
Latitude:37.422005
Longitude:-122.084095
Sent: 11:46PM

VII. SCOPE AND FUTURE DIRECTION

Students, employees have to keep their mobile phones silent most of the time because they on ringing they disturb other person studying or doing their work and many a times they forgot their phone after keeping them somewhere so this android based application allows the user to get the GPS location, changes the mobile profile from Silent mode to General mode, changes the ringtone volume accordingly and plays a music tone while misplaced. This all work is done by sending a code by from any other mobile message facility. And for future we are working on to add some features like getting contacts from phone by sending message

VIII. CONCLUSION

Students, employees have to keep their mobile phones silent most of the time because they on ringing they disturb other person studying or doing their work and many a times they forgot their phone after keeping them somewhere so this android based application allows the user to get the GPS location, changes the mobile profile from Silent mode to General mode, changes the ringtone volume accordingly and plays a music tone while misplaced.

- ➢ This all work is done by sending a code by from any other mobile message facility.
- ➢ This application hence, provides us an approach which enables us to manage android based OS applications remotely due to which we can make changes to our phone's setting i.e
  - Providing GPS location.
  - Changing its profile mode.
  - Changing its ringtone volume.
  - Play a music tone

REFERENCES

Books:-

1: wrox professional android 2 application development by reto meier

Online reading link:-

Location Manager

http://developer.android.com/reference/android/locatio n /LocationManager.html

Location

http://developer.android.com/reference/android/locatio n/Location.html

Shared Preferences

http://developer.android.com/reference/android/content /SharedPreferences.html

Location Listener

http://developer.android.com/reference/android/locatio n/LocationListener.html

Shared Preferences Editor

http://developer.android.com/reference/android/content/SharedReference.Editor.html

Broadcast Receiver

http://developer.android.com/reference/android/content/BroadcastReceiver.html

Sms Manager

http://developer.android.com/referece/android/telephony/SmsManager.html

Sms Message

http://developer.android.com/reference/android/telephony/SmsMessage.html

Audio Manager

http://developer.android.com/reference/android/media/AudioManager.html

Media Player

http://developer.android.com/reference/android/media/MediaPlayer.html