FIND YOUR ANDROID

Enrollment No.: 101246

Name of Student: Rachit Saini

Supervisor's Name: Ms. Ramanpreet Kaur

May-2014

Submitted in partial fulfillment of the Degree of Bachelor of Technology

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN (H.P)

Table of Contents

CERTIFICATE

This is to certify that the work titled "**FIND YOUR ANDROID"** submitted by "**RACHIT SAINI"** in partial fulfillment for the award of degree of B. Tech Computer Science Engineering of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

____

(Signature of Supervisor)

Name of Supervisor: Assistant Professor Ramanpreet Kaur

Designation: Assistant Professor

Date:

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to Ms. Ramanpreet Kaur, my Project Guide, for guiding and correcting me at every step of my work with attention and care. She has taken pain to go through the project and make necessary correction as and when needed. Thanks and appreciation to the helpful people at college for their support. I would also thank my university and my faculty members without whom this project would have been a distant reality. I also extend my heartfelt thanks to my family and friends for their undaunted support and faith in me.

Signature of the Student……………………………….

Name of the Student – Rachit Saini

Date -

SUMMARY

Objective

To develop an android application that provides location tracking functionality as well as changing the profile from silent/vibration mode to ringer mode of the android device using SMS

Description

The application locates device by making device ring and get latitude and longitude of an Android device.

This Android application shares location information with the users through SMS. In order to do that, the application receives SMS, matches contents of SMS with the alert message declared by the user for ringing and getting location. If alert message matches then application makes the device to ring or get location details.

Why I chose this topic?

Android app development is something new and interesting. I wanted to learn it so that I can make it my specialization and build interactive and exciting android apps in the future.

I find this project a great opportunity to learn about mobile technology and android operating system.

--------------------------	----------------------------
Signature of Student	Signature of Supervisor
Name:	Name:
Date:	Date:

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

The **Find your Android app** provides location tracking functionalities to Android devices using SMS. This application locates device by making device ring and get latitude and longitude of the android device. Also, **Find your android app** has the capability of authentication to share the location information with the sender of SMS.

1.2 PROBLEM STATEMENT

If you lose your phone and it is on silent or vibration mode then this app can help find your lost phone by making it ring just by texting an alert message from any other phone.

Also, you can get the GPS location of your lost phone by texting another alert message for location retrieval.

1.3 PROJECT DESCRIPTION

1.3.1 Purpose:

If an android user wants to know the location of Android device then user has to send SMS to designated device. So that he can locate device either by it making ring or get actual location of device using GPS.

1.3.2 Scope:

This project supports only the Android OS and makes communication with the tracker through SMS messages only. The Architecture, Security and the accuracy of tracking unit itself are the scope of this project.

CHAPTER 2

LITERATURE SURVEY

2.1 PROPOSED SYSTEM

2.1.1 Ringer

Using simple SMS commands you can ring your Android Device even though it is in silent mode and thus locate your device locally.

2.2.2 Location Tracking

In this proposed system you can locate the phone that has been misplaced or stolen if it has this app is installed on the phone, it can be located by sending SMS with alert message. The system gets you current updated location.

2.2 HISTORY OF MOBILE OPERATING SYSTEM

Operating system is the heart of mobile devices, which controls and interacts with the mobile hardware. Similar precept to an operating system such as Windows, Mac OS and Linux, that controls the desktop or laptop. Device which runs on operating system are smart phones, PDA's and tablet computers.

Everyone wants to do everything fast and on the go. When people where sitting back and diddling with the heavy computers for accessing the internet. An operating system called **Palm OS** was launched in year **1996** which brought a drastic change in the communication world. With the introduction of **Palm OS 2.0** in the year **1997**, accessing and sending mail via mobile evolved. The time when **Palm OS** was standing alone in the Smart phone market in the year **2000**, another giant bounced into the market, introducing **Windows Pocket PC 2000** which almost had most of the features of a computer.

Entertainment on the go was achievable with windows by launching **Pocket PC 2002** which incorporated MSN messenger and media player with enhanced user interface. Bluetooth an extraordinary invention for file transfer wirelessly. Bluetooth integration was successfully implemented in **Windows Mobile 2003** and browsing was made more comfortable with the pocket internet explorer. When windows were acquiring the smart phones market, **Palm OS Cobalt** bounced back with wifi and Bluetooth connectivity in **2004**.

In **2005**, Google acquired the **Android Inc** and **Blackberry's OS 4.1** was made available in the market. Windows interfaced the GPS management and office mobile in their **windows mobile 5**. When everyone was going upwards in updating the version and integrating application in the smart phones. The release of "**iPhone**" in **2007** disrupted the mobile industry and gave a new era of smart phone operating system with user experience which relies on touch based user interaction.

In **2007**, a trendsetting year when Google formed the OHA with 79 other hardware, software and telecommunication companies to make entry in to the smart phone market by introducing a legendary open source operating system **"ANDROID"** resulted in **2008** with **Android 1.0** which was available in the market. Android came up with a middleware which is responsible for hardware and communication between applications, and provides open source Android SDK application that allows embedded systems developers to use it to develop their own customizable Android platform applications. Some notable top applications such as Google map, E-mail, Instant messaging, Browser, GPS, Multimedia messaging are widely made available to the people in large only because of Android.

The enhancing grandness of smart phones has sparked off intense contenders amongst software giants such as **Google, Microsoft, and Apple**, as well as mobile industry leaders **Nokia, RIM, and Palm** to keep on updating their technology. In **2009**, **Samsung** too joined the roads of smart phones when they released a new operating system called as **BADA platform**. Nevertheless **Hewlett Packard Web OS** was also introduced in the same year. But **Google's Android** was climbing so high in a year, they acquired the major share in the smart phone operating system by upgrading from **Android 1.0, 1.1-1.6 till 2.1 (Éclair)** and **version 3.1 (Honeycomb)** was released in **2011.**

CHAPTER 3

INTRODUCTION TO ANDROID

3.1 WHAT IS ANDROID

Android, the world's most popular mobile platform is based on the Linux kernel, and designed primarily for touchscreen mobile devices such as smartphones and tablet computers. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance—a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. The first publicly available smartphone running Android, the HTC Dream, was released on October 22, 2008.

Since then many more smartphone manufacturers released mobile devices driven by Android. In June 2010, there were 60 different Android based mobile device models distributed to 49 countries by 59 carriers and 21 OEMs[5]. Due to Android's openness, its rich developer toolset and because of the Java language, Android became quickly popular among mobile application developers. About 110,000 Android applications and games can be found in the Android market rising by about 15 percent each month making it the second biggest mobile application store behind Apple's App Store which offers about 225,000 apps.

3.2 ANDROID MARKETSHARE

Android powers hundreds of millions of mobile devices in more than 190 countries around the world. It's the largest installed base of any mobile platform and growing fast—every day another million users power up their Android devices for the first time and start looking for apps, games, and other digital content.

In terms of market share, Android is the most popular mobile OS and as of 2013; its devices also sell more than Windows, iOS and Mac OS devices combined. In the third quarter of 2013, Android's share of the global smartphone shipment market was 81.3%, the highest ever. As of July 2013 the Google Play store has had over 1 million Android apps published, and over 50 billion apps downloaded. A developer survey conducted in April–May 2013 found that Android is used by 71% of mobile developers.

The operating system's success has made it a target for patent litigation as part of the so-called "smartphone wars" between technology companies. As of September 2013, one billion Android devices have been activated.

Android's source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices. Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially, released running other operating systems.

3.3 ANDROID GUI

The user interface of Android is based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects. Internal hardware—such as accelerometers, gyroscopes, and proximity sensors—is used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented. Android allows users to customize their home screens with shortcuts to applications and widgets, which allow users to display live content, such as emails and weather information, directly on the home screen. Applications can further send notifications to the user to inform them of relevant information, such as new emails and text messages. Despite being primarily designed for phones and tablets, it also has been used in televisions, games consoles, digital cameras, and other electronics.

3.4 ANDROID FEATURES

- **Messaging**
  SMS are available forms of messaging, including threaded text_messaging and Android Cloud__To__Device__Messaging(C2DM) and now enhanced version of C2DM, Android Google_Cloud_Messaging (GCM) is also a part of Android Push Messaging service.

- **Web browser**

The web browser available in Android is based on the open-source Blink (previously Web Kit) layout engine, coupled with Chrome's V8_JavaScript_engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

- **Voice based features**

Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards. As of Android 4.1, Google has expanded Voice Actions with the ability to talk back and read answers from Google's Knowledge Graph when queried with specific commands. The ability to control hardware has not yet been implemented.

- **Multi-touch**

Android has native support for multi touch which was initially made available in handsets such as the HTC_Hero. The feature was originally disabled at the kernel level. Google has since released an update for the Nexus_One and the Motorola Droid which enables multi-touch natively.

- **Multitasking**

Multitasking of applications, with unique handling of memory allocation, is available.

- **Screen capture**

Android supports capturing a screenshot by pressing the power and volume-down buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations or otherwise by using a PC connection (DDMS developer's tool). These alternative methods are still available with the latest Android.

- **Multiple language support**

Android supports multiple languages.

- **Openness**

Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It is built to be truly open

- **All applications are created equal**

Android does not differentiate between the phone's core applications and third-party applications. They can all be built to have equal access to a phone's capabilities providing users with a broad spectrum of applications and services. With devices built on the Android Platform, users will be able to fully tailor the phone to their interests.

- **Breaking down application boundaries**

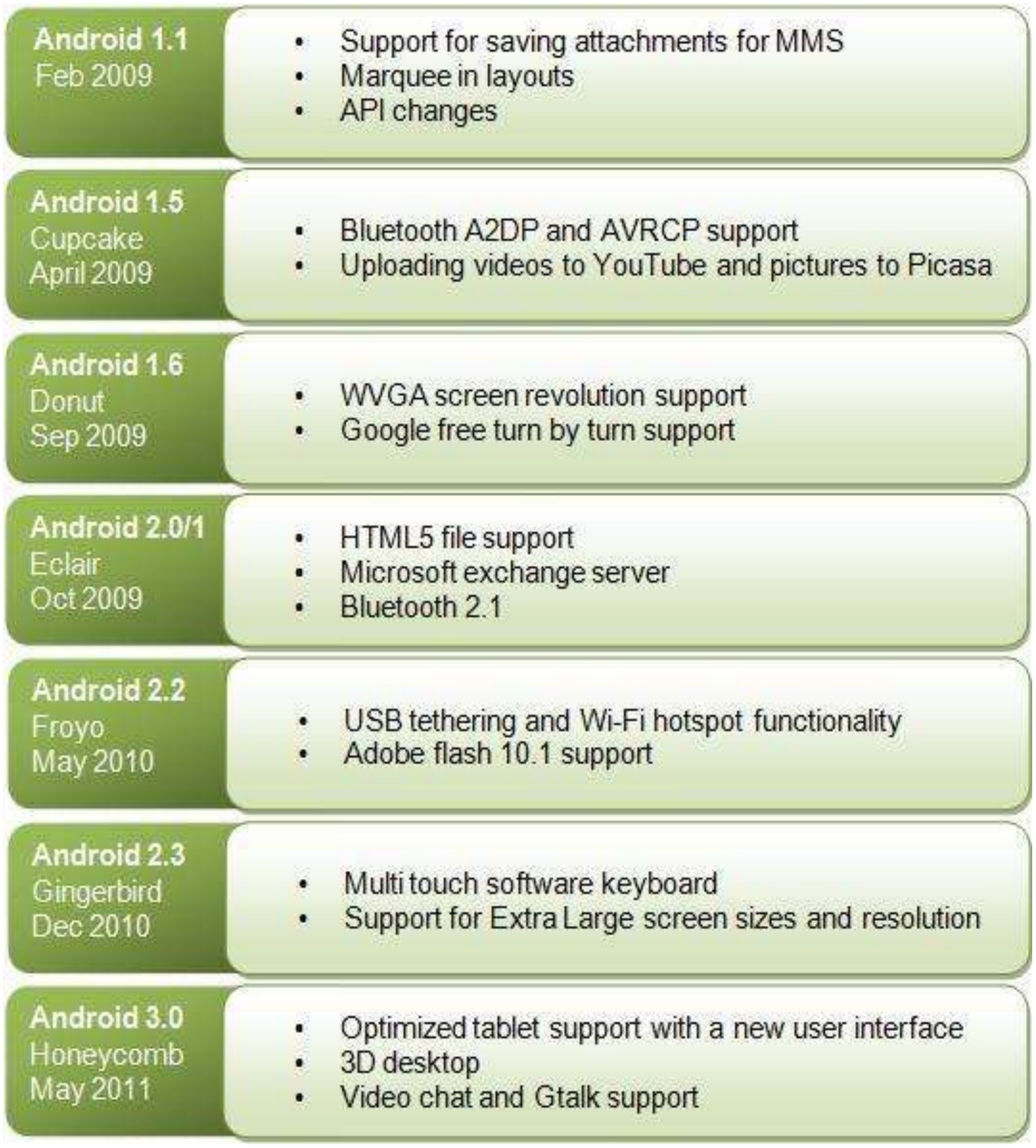
Android breaks down the barriers to building new and innovative applications. For example, with Android, a developer could build an application that enables users to view the location of their friends and be alerted when they are in the vicinity giving them a chance to connect.

- **Memory Management**

Since Android devices are usually battery-powered, Android is designed to manage memory (RAM) to keep power consumption at a minimum, in contrast to desktop operating systems which generally assume they are connected to unlimited mains_electricity. When an Android app is no longer in use, the system will automatically suspend it from memory.

3.5 ANDROID VERSIONS

After original release there have been number of updates in the original version of Android.



Android 1.1 Feb 2009	• Support for saving attachments for MMS • Marquee in layouts • API changes
Android 1.5 Cupcake April 2009	• Bluetooth A2DP and AVRCP support • Uploading videos to YouTube and pictures to Picasa
Android 1.6 Donut Sep 2009	• WVGA screen revolution support • Google free turn by turn support
Android 2.0/1 Eclair Oct 2009	• HTML5 file support • Microsoft exchange server • Bluetooth 2.1
Android 2.2 Froyo May 2010	• USB tethering and Wi-Fi hotspot functionality • Adobe flash 10.1 support
Android 2.3 Gingerbird Dec 2010	• Multi touch software keyboard • Support for Extra Large screen sizes and resolution
Android 3.0 Honeycomb May 2011	• Optimized tablet support with a new user interface • 3D desktop • Video chat and Gtalk support

Figure 1: Android Versions

3.6 Architecture of Android OS

The skeleton of Android framework and its constituents are shown in the following figure:

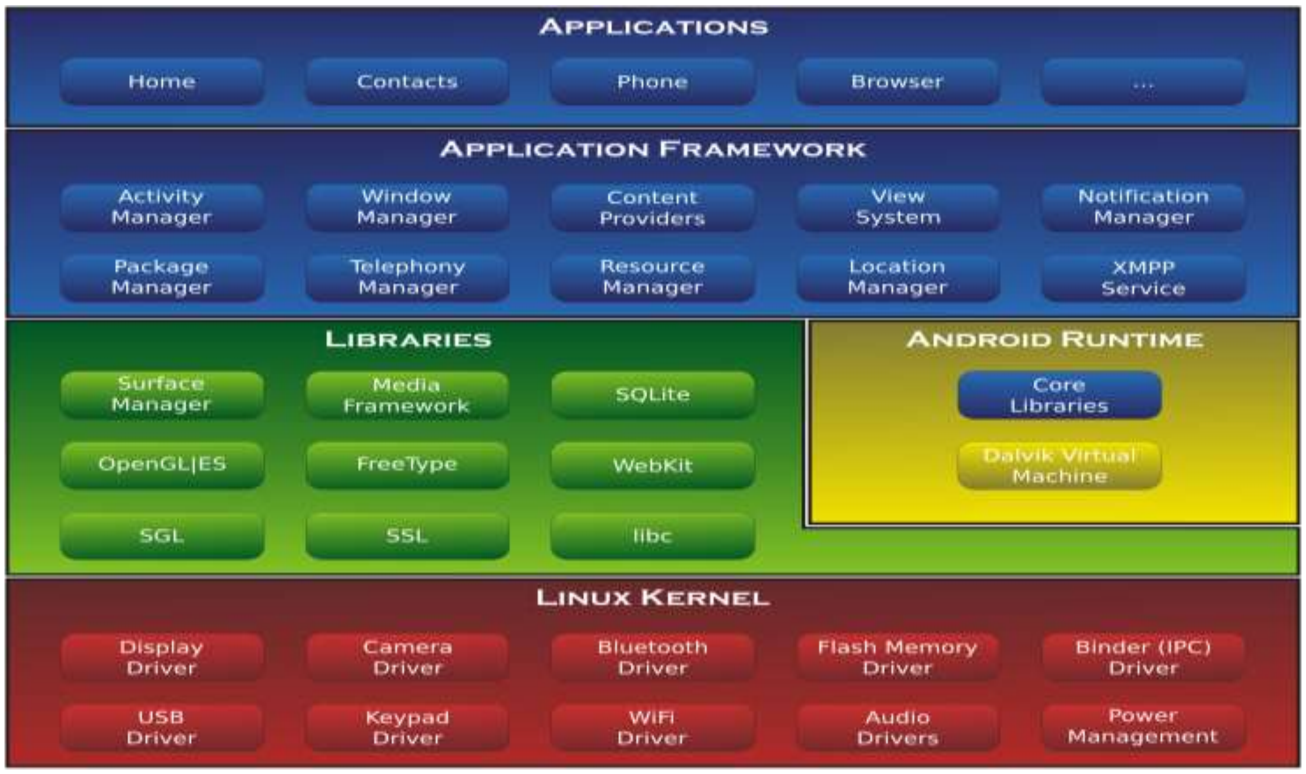


Figure 2: Architecture of Android OS

- **Applications Layer**

Android ships with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts and others. All applications are built using the Java. Each of the application aims at performing a specific task that it is actually intended to do.

- **Application Framework Layer**

The next layer is the application framework. This includes the programs that manage the phone's basic functions like resource allocation, telephone applications, switching between processes or programs and keeping track of the phone's physical location. Application developers have full access to Android's application framework. This allows them to take advantage of Android's processing capabilities and support features when building an Android application. We can think of the application framework as a set of basic tools with which a developer can build much more complex tools.

- **Libraries Layer**

Also below the application layer are native libraries written in C/C++. These libraries are implemented and complied for the specific underlying hardware architecture. A selection of common C/C++ libraries like the OpenGL 2D and 3D graphics library, the SQLite database library, the standard C system library (libc) and various other libraries are forming among additional Android specific libraries the layer on top of the Linux kernel.

- **Android Runtime Layer**

Below the application framework, the Android runtime can be found. The runtime consists of core Android libraries and the Dalvik virtual machine (VM). Besides Android specific libraries, the Android runtime includes a subset of core libraries from the Java Standard Edition and the Java Mobile Edition. The Dalvik VM is a virtual machine similar to Java's VM, however differs in some important aspects. Designed and developed by Dan Bornstein at Google, the Dalvik VM does not execute .class files, but highly optimized, compact .dex, Dalvik Executables, files. Android applications which are usually written in Java (actually it is also possible to develop C/C++ code using the Android NDK[7]) and compiled to .class files using the Java language compiler are transformed during compile time to .dex files utilizing the "dx" tool that is part of the Android SDK. The Dalvik VM has been designed for low memory requirements and to run multiple VM instances efficiently, while each Android application is executed in its own VM running in its own process.

- **Linux Kernel**

The Linux kernel, originally developed by Linus Torvalds in 1991, builds the foundation for Android's memory and process management as well as security and power management and networking services. In addition, the kernel contains all necessary hardware drivers providing a hardware abstraction layer for layers on top of the Linux kernel layer.

CHAPTER 4

SOFTWARE AND TOOLS REQUIRED

4.1 ECLIPSE

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It can be used to develop applications in Java and, by means of various plug-ins, other programming languages like c, c++, android etc.

4.1.1 Features of Eclipse

1) It is an open source.
2) It is strongly recommended by Android developer.
3) It is directly linked with compiler, so we don't need to compile the program
4) It has good UI(user interface)

4.1.2 Downloading and Installing Eclipse

Step 1: Go to http://www.eclipse.org/downloads/

Step 2: For Windows users, you will have to know what type of version of your OS you have. If your computer is a 64 bit Windows, select Windows 64 and if you have a 32 bit Windows, select Windows 32 bit.

Step 3: Once you have downloaded the Eclipse archive you will need to decompress the zip file, which will create the unzipped Eclipse folder. You may want to extract the archive to the root of C:\ drive, thus creating the folder "C:\eclipse", or just moved the extracted eclipse folder to the root of C:\ drive if you extracted it already. Since Eclipse does not have any installer, there will be a file inside the Eclipse folder named eclipse.exe ( ). You can double click on the file to run Eclipse

Step 4: After eclipse has been fully installed and extracted, create a workspace folder where you will contain all the program files you create.

Step 5: Now that you have finished installing Eclipse, restart your computer.
Restarting your computer refreshes system memory and allows registration or configuration changes made by installers and uninstallers to take effect

4.2 JAVA DEVELOPMENT KIT (JDK)

A Java Development Kit (JDK) is a program development environment for writing Java applets and applications. It consists of a runtime environment that "sits on top" of the operating system layer as well as the tools and programming that developers need to compile, debug, and run applets and applications written in the Java language.

4.2.1 Downloading and Installing JDK

Step 1: Visit the Java downloads page on Oracle's website to find the JDK environment download. Scroll down until you find Java SE 6 Update 43, and download JDK.

Step 2: Once you have selected download, accept the terms of service and choose the correct OS corresponding for the specific JDK. (Windows, Mac, Linux, etc.)

Step 3: Once the download is complete, double click the file to begin the installation of JDK.

Step 4: After the initial installation is done, a pop up asking you where your source java files will be. You can choose to change where you want to keep your folder but it’s best to stick with what you were given first.

4.3 ANDROID SDK

A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications.

Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linux kernel.

4.3.1 Downloading and Installing Android SDK

Step 1: Download the Android SDK, go to http://developer.android.com/sdk/index.html, Expand "Download For Other Platforms", Under "SDK Tools Only", Download the appropriate SDK Tools for your operating platform. Choose the ZIP version, e.g., android-sdk_r22.6-windows.zip (104 MB).

Step 2: Install Android SDK, unzip the downloaded file into a folder of your choice. Take note of the installed directory.

Step 3: Install Android Platforms and Add-ons via "SDK Manager"

The Android SDK comprises 2 parts: the "tools" and the "Platforms & Add-ons". The previous step installed the basic "tools", which are executable that support app development.

Now, we have to setup our Android "Platforms & Add-ons".

1. Launch Android's "SDK Manager", which is responsible for managing the Android components. Launch the SDK manager by running (double-clicking) "SDK Manager.exe" under the Android installed directory.
2. In "Add Platforms and Packages", select your target Android platforms and add-ons packages. For novices, select "Android SDK Platform-Tools", and at least one (the latest) Android platform (e.g., Android 4.4 (API 19)) ⇒ "Install".

4.4 ANDROID DEVELOPER TOOLS (ADT) plugin

ADT (Android Developer Tools) is a plugin for Eclipse that provides a suite of tools that are integrated with the Eclipse IDE. It offers you access to many features that help you develop Android applications quickly. ADT provides GUI access to many of the command line SDK tools as well as a UI design tool for rapid prototyping, designing, and building of your application's user interface.

Because ADT is a plugin for Eclipse, you get the functionality of a well-established IDE, along with Android-specific features that are bundled with ADT.

4.4.1 Installing Eclipse ADT Plugin

Step 1: Launch Eclipse.

Step 2: Install Eclipse ADT: From Eclipse's "Help" menu ⇒ "Install New Software..." ⇒ In "Work with", enter https://dl-ssl.google.com/android/eclipse/ ⇒ Check ALL boxes ⇒ Next ⇒ Finish ⇒ Restart Eclipse to use ADT plugin.

Step 3: Configure Eclipse ADT: From Eclipse's "Window" menu ⇒ Preferences ⇒ Android ⇒ In "SDK Location", select the Android SDK installed directory (that you have chosen in Step 2).

CHAPTER 5

ANDROID APPLICATIONS

5.1 INTRODUCTION

An Android application is a bundle of application components, resources and data files packaged into an Android package (.apk) file. Nonetheless, an Android application is a loosely coupled conglomerate of separately executable components. Therefore, instead of having a main method as a single entry point to start an application, familiar from desktop application development, in Android each component of an application can be started individually assuming the application allows it. For example, if application A implemented a fancy scrollable list of contacts and application B needs to display a list of contacts, application B could simply start, if possessing the permission, the specific application component which implements the scrollable contacts list of application A in order to display the list of contacts without implementing this particular feature again. Native Android applications offer already many useful application components which can be reused in that way. In part two of this thesis it will be shown that App uses the same concept by calling the configuration screen of an App Plug-in to smoothly integrate plug-ins developed by third parties.

Each Android application package is described by one structured XML file (AndroidManifest.xml). It declares the components available within the application and defines permissions required by external applications to start components. Additionally, the manifest file specifies other useful meta-information needed by Android to install the application properly, such as icon, title, version, and so on.

5.2 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 application and its data. However, it is important to note that an application can launch another program that will run with the launching application's UID.

For a developer to run an application on the Android phone, his or her 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.

5.3 APPLICATION COMPONENTS

The basic components of an Android application include Activity, Broadcast Receiver, Service, and Content Provider. Each of these which when used for any application has to be declared in the AndroidManifest.xml. The user interface of the component is determined by the Views. For the communication among these basic components we use Intents and Intent filters which play crucial role during app development.

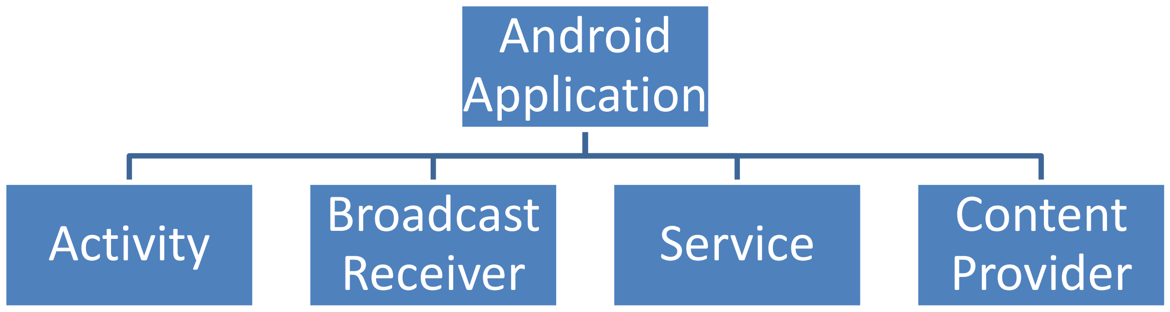


Figure 3: Structure of Android Components

- **Activity**

An Activity is, fundamentally, an object that has a lifecycle. An Activity is a chunk of code that does some work; if necessary, that work can include displaying a UI to the user. It doesn't have to, though-some Activities never display UIs. Typically, we will designate one of our application's Activities as the entry point to our application.

- **Broadcast Receiver**

Broadcast Receiver is yet another type of component that can receive and respond to any broadcast announcements.

- **Service**

A Service is a body of code that runs in the background. It can run in its own process, or in the context of another application's process, depending on its needs. Other components "bind" to a Service and invoke methods on it via remote procedure calls. An example of a Service is a media player; even when the user quits the media-selection UI, she probably still intends for her music to keep playing. A Service keeps the music going even when the UI has completed.

- **Content Provider**

Content Provider is a data storehouse that provides access to data on the device; the classic example is the Content Provider that's used to access the user's list of contacts. Our application can access data that other applications have exposed via a Content Provider, and we can also define our own Content Providers to expose data of our own.

5.4 APPLICTION 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 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 applications promised features. and the permissions required. These permissions are different from file permissions. Once an application is installed, its permissions can’t be changed. The less permission an application needs, the more comfortable the user should feel installing the application.

5.5 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.

5.6 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 [3]. 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.

CHAPTER 6

LOCATION BASED SERVICES IN ANDROID

Android's Network Location Provider determines user location using cell tower and Wi-Fi signals, providing location information in a way that works indoor and outdoor, responds faster, and uses less battery power. The purpose of location-based services is to find the Physical location of the device. Access to the location-based services is handled by the LocationManager system Service. To access the Location Manager, request an instance of the LOCATION_SERVICE using the get System Service() method. Current Location can be fetched using two ways:

1. GPS (Global Positioning System)
2. Network Service Location

6.1 GPS (GLOBAL POSITIONING SYSTEM)

The Global Positioning System (GPS) uses a constellation of 24 satellites orbiting the earth. GPS finds the user position by calculating differences in the times the signals, from different satellites, take to reach the receiver. GPS signals are decoded, so the smart phone must have in-built GPS receiver. To get access to GPS hardware of android we request using following statement **LocationManager.GPS_PROVIDER;**

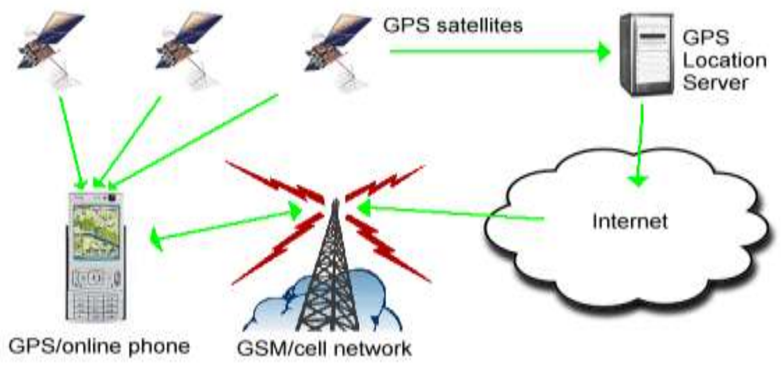


Figure 4: Architecture of A-GPS System

6.2 NETWORK SERVICE LOCATION

The current cell ID is used to locate the Base Transceiver Station (BTS) that the mobile phone is interacting with and the location of that BTS. It is the most basic and cheapest method for this purpose as it uses the location of the radio base station that the cell phone is connected to. A GSM cell may be anywhere from 2 to 20 kilometers in diameter. Other approaches used along with cell ID can achieve location granularity within 150 meters. The granularity of location information is poor due to Wide Cell Range. The advantage is that no additional cost is attached to the handset or to the network to enable this service.
To get access to Network Provider android we request using following statement **LocationManager.NETWORK_PROVIDER;**

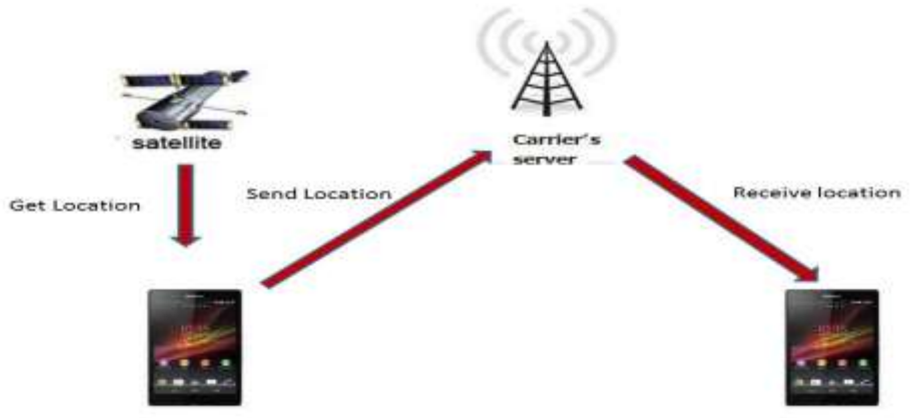


Figure 5: Architecture of Network Service Location

CHAPTER 7

SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification (SRS) is the starting point of the software development activity. It is a complete description of the behavior of a system which is to be developed. The SRS document enlists all necessary requirements for project development. To derive the requirements we need to have clear and thorough understanding of the product which is to be developed. This is prepared after detailed communication with project team and the customer.

A SRS is a comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform.

An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development cost. A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

7.1 CHARCTERSTICS OF SRS

- **Correct** - An SRS is correct if, and only if, every requirement stated therein is one that the software shall meet. Traceability makes this procedure easier and less prone to error.
- **Unambiguous** - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. As a minimum, this requires that each characteristic of the final product be described using a single unique term.
- **Verifiable** – It is verifiable if there exists some finite cost-effective process with which a person or machine check whether software product meets requirements.
- **Consistent** - Consistency refers to internal consistency. If an SRS does not agree with some higher-level document, such as a system requirements specification, then it is not correct. An SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict.

- **Modifiable** – SRS is said to be modifiable if its structure and style are such that any changes to the requirements can be made easily, completely and consistently while retaining the structure and style.
- **Traceable** – SRS is said to be traceable if the origin of each of its requirements is clear and it facilitates the referencing of each requirement in future enhancement.
- **Ranked for importance or stability** – SRS is ranked for importance or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement.

7.2 FUNCTIONAL REQUIREMENTS

Modules: This application contains two important modules.

1. Ringer

2. Location Tracker

1. **Ringer**

- Be able to recognize the attention word received through SMS.
- Be able to handle the phone state to ring automatically.
- Be able to send phone state through SMS.

2. **Location Tracker**

- Be able to detect the current location of Android device.
- Be able to retrieve the device, SIM card & location details.
- Be able to send retrieved details through SMS.

7.3 NON FUNCTIONAL REQUIREMENTS

- **Performance Requirements:**

Application must respond within 5 seconds excluding GPS enabling time. The user must use the required option to get the information of the users.

- **Reliability:**

This application has various other features like SMS this can be extensible with many features in the user devices.

- **Availability:**

This proposed system find extended application who are installed this application those users can be get the location of the device and send the details back to requesting phone.

- **Maintainability:**

Since we are using JAVA software to support our application no maintenance is very easy and economical also.

- **Portability:**

The project is built using JAVA and can be run on any device which uses android OS.

- **Safety Requirements:**

It is better to use the antivirus and keep on checking for the latest updates of the application.

- **Security Requirements:**

The application will prompt the user for upgrading and downloading new features updated by the developer.

7.4 SYSTEM REQUIREMENTS

7.4.1 Hardware Requirements

- **On Developer Side**

Processor	:	Dual core or above.
RAM	:	4GB.
Hard disk	:	40GB or above.
Monitor	:	15’’ LCD or CRT Monitor or above.
Keyboard	:	Standard windows keyboard

- **On Client Side**

Device	:	GPS enabled Android OS mobile.

7.4.2 Software Requirements

Development Kit	:	Android SDK 2.3, Java JDK 1.6.
Languages	:	Java.
IDE	:	Eclipse Helios, Android Emulator.
Platform	:	Window 7/XP.

CHAPTER 8

SYSTEM DESIGN

System design is the solution to the creation of a new system. This phase is composed of several systems. This phase focuses on the detailed implementation of the feasible system. It emphasis on translating design specifications to performance specification is system design. System design has two phases of development logical and physical design.

During logical design phase the analyst describes inputs (sources), outputs (destinations), databases (data stores) and procedures (data flows) all in a format that meats the uses requirements. The analyst also specifies the user needs and at a level that virtually determines the information flow into and out of the system and the data resources. Here the logical design is done through data flow diagrams and database design.

The physical design is followed by physical design or coding. Physical design produces the working system by defining the design specifications, which tell the programmers exactly what the candidate system must do.

The programmers write the necessary programs that accept input from the user, perform necessary processing on accepted data through call and produce the required report on a hard copy or display it on the screen.

8.1 SYSTEM ARCHITECTURE

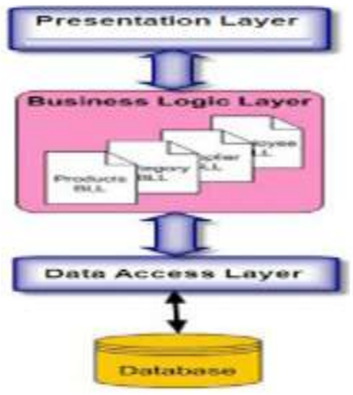
8.1.1 Architectural Design:

3-Tier architecture is also called layered architecture. Some people called it n-tier architecture. Layer architectures are essentially objects and work in object oriented environment. 3-tier architecture is a very well-known architecture in the world of software development, it doesn't matter whether you are developing web based application or desktop based, it is the best architecture to use.

- **Presentation Layer**
  Presentation layer consists of pages like .java or desktop based form where data is presented to users for getting input from users.

- **Business Logic layer or Business Access Layer**
  Business logic layer contains all of the business logic. Its responsibility is to validate the business rules of the component and communicating with the Data Access Layer. Business Logic Layer is the class in which we write functions that get data from Presentation Layer and send that data to database through Data Access Layer.

- **Data Access Layer**
  Data Access Layer is also the class that contains methods to enable business logic layer to connect the data and perform desired actions. These desired actions can be selecting, inserting, updating and deleting the data. DAL accepts the data from BAL and sends it to the database or DAL gets the data from the database and sends it to the business layer. In short, its responsibility is to communicate with the backend structure.



<u>Fig 6: Illustration of 3-Tier Architecture</u>

8.2 UML DIAGRAMS

8.2.1 State Diagram

A State diagram is a graph whose nodes are states and whose directed arcs are transitions between the states. It specifies the state sequences caused by event sequences. State names must be unique within the scope of the diagram. State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented in series of events that could occur in one or more possible states.

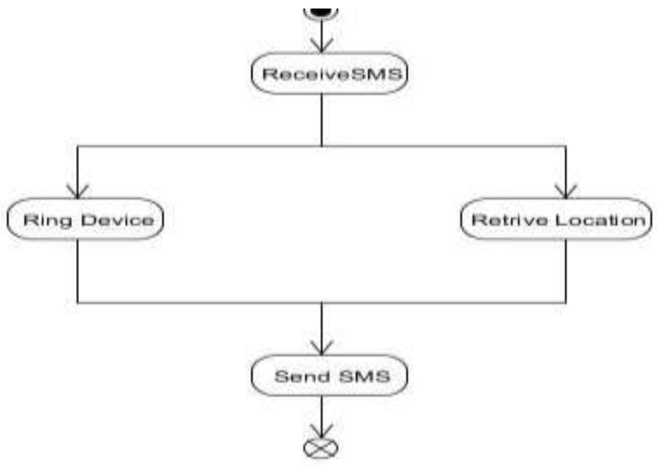


Fig 7: State Diagram

8.2.2 Activity Diagram

Activity Diagram shows the sequence of steps that make up complex process. It shows the flow of control, similar to sequence but focuses on operation rather than on objects.

The components used in this are as follows:

○ **Rounded Rectangle**	– It indicates the process.
○ **Arrow**	– It indicates transition line.
○ **Rhombus**	– It indicates the decision.
○ **Bars**	– It represents the start or end of concurrent activities.
○ **Solid Circle**	– It represents the initial state of workflow.
○ **Encircled Black Circle**	–It represents the final state of workflow.

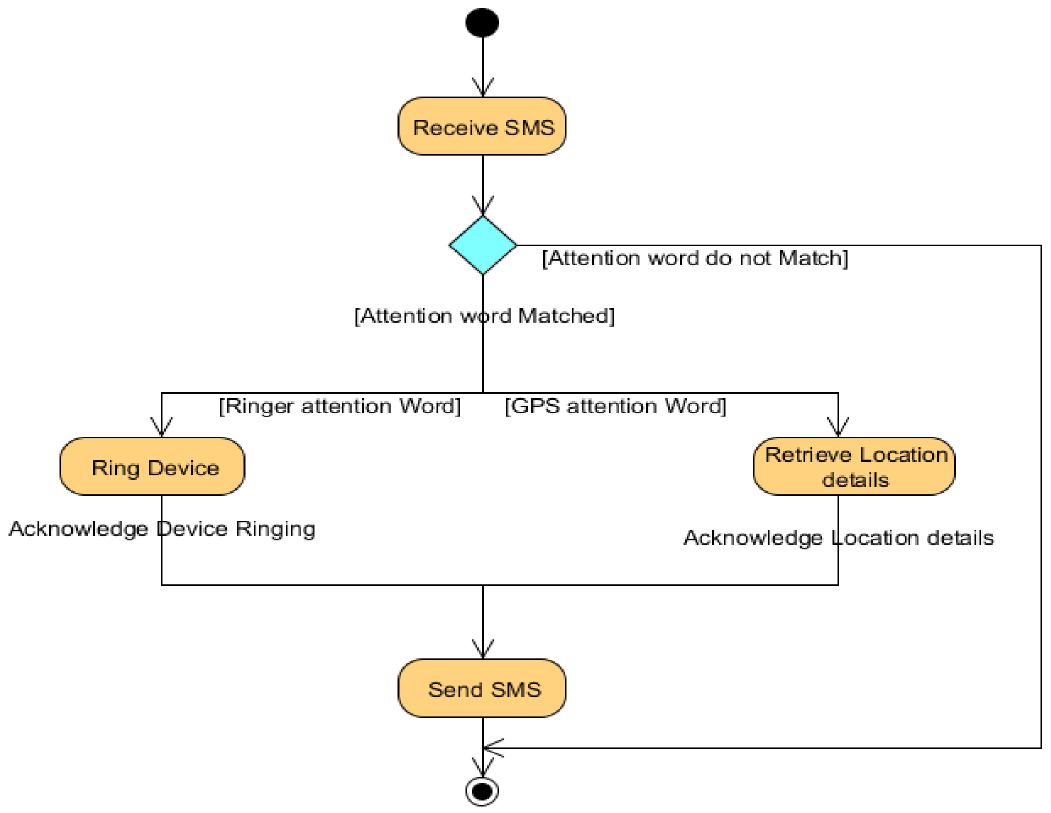

Receive SMS
[Attention word do not Match]
[Attention word Matched]
[Ringer attention Word]
[GPS attention Word]
Ring Device
Retrieve Location details
Acknowledge Device Ringing
Acknowledge Location details
Send SMS

Fig 8: Activity Diagram

8.2.3 Sequence Diagram

A Sequence diagram shows how a set of objects communicate with each other to perform a complex task. This type of diagram allows the other developer to verify that the interaction is correct.

A Sequence diagram shows, as parallel **vertical lines (lifelines),** different processes or objects that live simultaneously, and as **horizontal arrows**, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

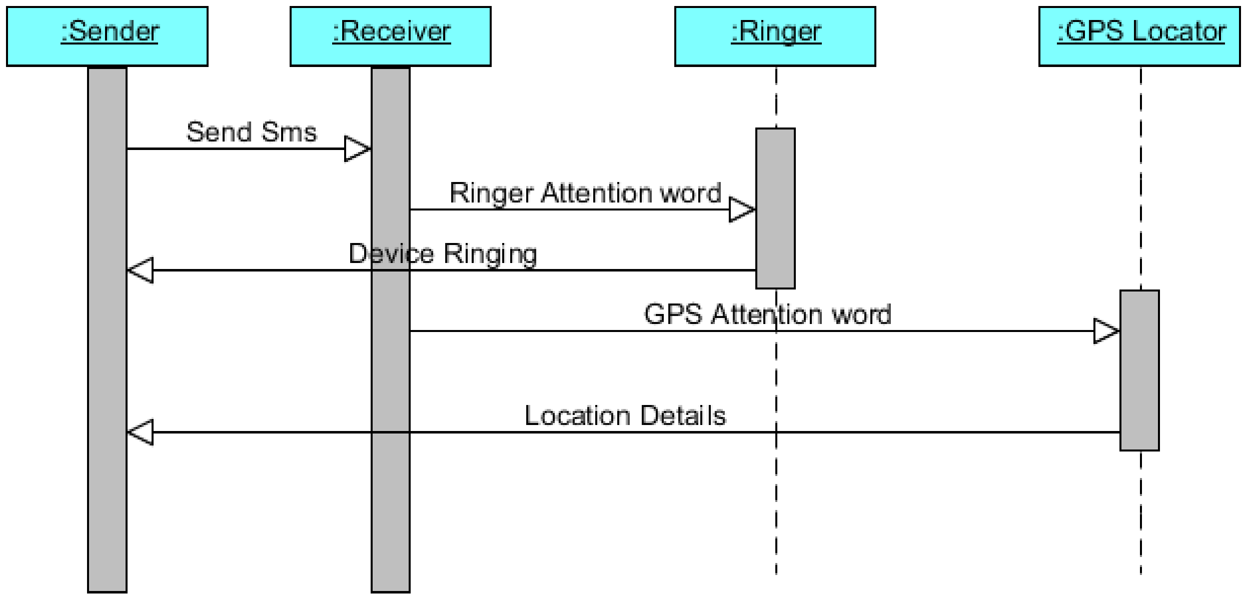


Fig 9: Sequence Diagram

CHAPTER 9

IMPLEMENTATION

9.1 INTRODUCTION

After designing the new system, the whole system is required to be converted into computer understanding language. Coding the new system into computer programming language does this. It is an important stage where the defined procedures are transformed into control specifications by the help of a computer language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer as programs. The programs coordinate the data movements and control the entire process in a system. It is generally felt that the programs must be modular in nature. This helps in fast development, maintenance and future change, if required.

The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing user to take over its operation for use and evaluation. Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operated the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

9.2 MODULES

1. Send alert message: Perform predefined action according to alert word and abort broadcasting.
2. Make device ring.
3. Acknowledges device ringing status to the user by sending SMS.
4. Get latitude and longitude of device.
5. Send device location to sender of SMS.
6. Exit Application

9.3 IMPLEMENTATION OF MODULES

- **Broadcast receiver that alerts application when each new SMS arrived.**

This module decides which action has to perform when attention word matches with the keyword for ringer volume. If it is matched then it starts activity which enables device ringing. If attention word matches with the keyword for getting location then it starts activity which retrieves location of device and sends information to the sender of SMS. At the same time it aborts message broadcasting so that message can't be reached to inbox of native messaging application.

If attention word is not matched with the specified key word than it simply allow broadcasting so that message can be reached to inbox of native messaging application.

Step 1: START
Step 2: SMS received.
Step 3: Checks attention word.
Step 4: If attention word for ringer matches then starts ringing activity and abort broadcasting.
Step 5: If attention word for getting location matches then starts ringing activity and aborts broadcasting.
Step 6: If attention word not matched then allow broadcasting.
Step 7: End

- **Enable device ringing and acknowledges the user.**

In this module we provide the functionality of making device ringing by sending an attention word to android device. **Find Your Android** recognizes the keyword and makes device ringing no matter it is in silent or vibrate mode. So user can locate his phone.
Step 1: START
Step 2: Checks device it in silent or vibrate mode.
Step 3: If it is in silent or vibrate mode than set device to ringing mode.
Step 4: Enable device ringing.
Step 5: Acknowledges user that device ringing by sending device status information to user.
Step 6: END

- **Get location And Acknowledges user.**

In this module we provide the functionality of getting location details of device and the same will be sent to user. **Find Your Android** recognizes the keyword and retrieves latitude and longitude of device and the same will be sent to sender of SMS. So user can locate his phone.

Step 1: START
Step 2: Checks that internet is available.
Step 3: If internet is available then get location details from Network Provider.
Step 4: If internet is not available then Checks is GPS turned on.
Step 5: If GPS is available then get location details.
Step 6: Send location information to user.
Step 7: End

CHAPTER 10

CODE IMPLEMENTATION

10.1 ACTIVITY_MAIN.XML CODE

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center_horizontal"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

  <Button
      android:id="@+id/btn_Settings1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentBottom="true"
      android:layout_alignParentLeft="true"
      android:layout_marginBottom="161dp"
      android:layout_marginLeft="60dp"
      android:onClick="appSettings"
      android:text="App Settings" />

  <TextView
      android:id="@+id/textviewforvol"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="84dp"
      android:text="Find your lost phone"
      android:textAppearance="?android:attr/textAppearanceLarge" />
  </RelativeLayout>
```

10.2 APP SETTINGS.XML

```
<?xml version="1.0" encoding="utf-8" ?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

    <TextView android:id="@+id/textViewsms"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center_horizontal"
      android:layout_marginTop="20dp"
      android:text="SMS Text Settings"
      android:textAppearance="?android:attr/textAppearanceLarge" />
  <TextView android:id="@+id/textviewforvol"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginLeft="10dp"
      android:layout_marginTop="40dp"
      android:text="For Volume Increase"
      android:textAppearance="?android:attr/textAppearanceLarge" />
- <EditText android:id="@+id/edit1"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="20dp"
```

```
android:layout_marginRight="10dp"
android:layout_marginTop="10dp" android:ems="10">
<requestFocus />
</EditText>
<TextView android:id="@+id/textViewforloc"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginLeft="10dp"
android:layout_marginTop="10dp"
android:text="For Location "
android:textAppearance="?android:attr/textAppearanceLarge" />
<EditText                  android:id="@+id/edit2"                  android:layout_width="match_parent"
android:layout_height="wrap_content"                                 android:layout_marginLeft="20dp"
android:layout_marginRight="10dp" android:layout_marginTop="10dp" android:ems="10" />
- <LinearLayout android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal">
<Button android:id="@+id/btnsave"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:onClick="onSave" android:text="Save" />
<Button android:id="@+id/btncancel"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:onClick="onCancel" android:text="Cancel" />
</LinearLayout>
</LinearLayout>
```

10.3 ANDROID MANIFEST.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="saini.findmyphone"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
 <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>

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

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="saini.findmyphone.MainActivity"
            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="saini.findmyphone.AppSettings"
            android:label="@string/app_name" >
        </activity>

         <receiver android:name="saini.findmyphone.SmsReader">

            <intent-filter>
               <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>

        </receiver>

    </application>

</manifest>
```

10.4 MAINACTIVITY.JAVA

```
package saini.findmyphone;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.widget.TextView;
public class MainActivity extends Activity {

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
                @SuppressWarnings("unused")

                TextView txt2 = (TextView) findViewById(R.id.textviewforvol);

        }
        @Override
        public boolean onCreateOptionsMenu(Menu menu) {

                // Inflate the menu; this adds items to the action bar if it is present.

        getMenuInflater().inflate(R.menu.main, menu);

                return true;
        }
        public void appSettings(View v)
        {
                Intent settings = new Intent(this, AppSettings.class);

                startActivity(settings);
        }
}
```

10.5 SMS READER.JAVA

```
package saini.findmyphone;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.SharedPreferences;

import android.location.Criteria;

import android.location.Location;
import android.location.LocationManager;

import android.media.AudioManager;

import android.media.Ringtone;

import android.media.RingtoneManager;

import android.net.Uri;

import android.os.Bundle;

import android.telephony.SmsManager;

import android.telephony.SmsMessage;

import android.util.Log;

import android.widget.Toast;



public class SmsReader extends BroadcastReceiver {

        String pref_volume, pref_loc ;


        @Override

        public void onReceive(Context arg0, Intent arg1) {

                // TODO Auto-generated method stub

                // Retrieves a map of extended data from the intent.

                final Bundle bundle = arg1.getExtras();

                Log.i("SmsReceiver", "On Receive Invoked");

                readPrefereces(arg0) ;
```

```
String phoneNumber ;
                    try {
                                 if (bundle != null) {
                             final Object[] pdusObj = (Object[]) bundle.get("pdus");
                               for (int i = 0; i < pdusObj.length; i++) {
 SmsMessage currentMessage = SmsMessage.createFromPdu((byte[]) pdusObj[i]);
                              phoneNumber = currentMessage.getDisplayOriginatingAddress();


                              String senderNum = phoneNumber;
                              String message = currentMessage.getDisplayMessageBody();
                                        Log.i("SmsReceiver", "senderNum: "+ senderNum + "; message: "
          + message);


                             // Show alert
                              int duration = Toast.LENGTH_LONG;
                              Toast toast = Toast.makeText(arg0, "senderNum: "+ senderNum + ",
message: " + message, duration);
                              toast.show();
                              if (message.equalsIgnoreCase(pref_volume))
                              {
                                        toast = Toast.makeText(arg0, "Mute remove requested", duration);
                                        toast.show();
```

```
        Log.i("SmsReceiver", "Mute remove requested");
AudioManageraudioManager=
(AudioManager)arg0.getSystemService(Context.AUDIO_SERVICE);
                        if((audioManager.getRingerMode()==
AudioManager.RINGER_MODE_VIBRATE)    ||    (audioManager.getRingerMode()    ==
AudioManager.RINGER_MODE_SILENT))
                        {
    int maxVolume = audioManager.getStreamMaxVolume(AudioManager.STREAM_RING);

audioManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
audioManager.setStreamVolume(AudioManager.STREAM_RING, maxVolume,
                        AudioManager.FLAG_SHOW_UI+
AudioManager.FLAG_PLAY_SOUND);
                        }
                        playAlertSound(arg0);
                      }

                      if (message.equalsIgnoreCase(pref_loc))
                      {
                            String smsText ;
                      toast = Toast.makeText(arg0, "Location SMS Requested", duration);
                            toast.show();
                            Log.i("SmsReceiver", "Location SMS Requested");
                            smsText = getLocation(arg0) ;
                            sendSMS(phoneNumber,smsText);
        }
```

```
   } // end for loop
                      } // bundle is null
                 } catch (Exception e) {
                    Log.e("SmsReceiver", "Exception smsReceiver" +e);
                 }
         }
         public void sendSMS(String phoneNumber, String msgText)
         {
                 // Get the object of SmsManager
                 final SmsManager sms = SmsManager.getDefault();
                 sms.sendTextMessage(phoneNumber, null, msgText, null, null );
         }
         private void playAlertSound(Context cont) {
         Uri notification= RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
                 Ringtone rAlert = RingtoneManager.getRingtone(cont , notification);
                 rAlert.play();
                 try
{
                 Thread.sleep(10000);
                 }
                 catch (Exception e)
                 {
```

```
                Log.e("Phone Alert", e.getMessage());
            }
            rAlert.stop();
        }
            public String getLocation(Context cont)
        {
            String locStr ;
            double latitude, longitude ;
        /*              Get the Location Manager Instance           */
            LocationManager          locMgr          =          (LocationManager)
    cont.getSystemService(Context.LOCATION_SERVICE);
            /* Set the criteria for selecting the Location Providers. */
        Criteria criteria = new Criteria();
         String provider = locMgr.getBestProvider(criteria, true);
         /* Get the last known address from location provider */
            Location loc=locMgr.getLastKnownLocation(provider);
            if (loc!= null)
            {
          latitude = loc.getLatitude();
                longitude = loc.getLongitude();
                        locStr            =            "Latitude:"+
            String.valueOf(latitude)+"\nLongitude:"+ String.valueOf(longitude) ;
            }
            else
            {
```

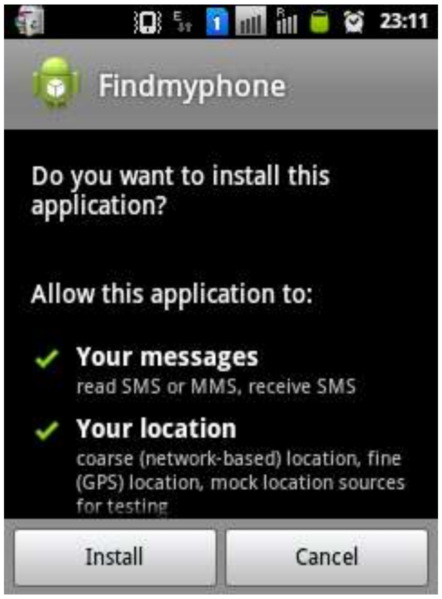
```
            locStr = "Location Not Available" ;
                        }
             Log.d("Location", provider +":"+ locStr);
                return locStr ;
            }
            public void readPrefereces(Context ctx)
            {
                        SharedPreferences sp ;
                        sp = ctx.getSharedPreferences("Findmyphone",0);
                        pref_volume=sp.getString("pref_Volume", "Volume");
                        pref_loc=sp.getString("pref_Location","Location");
            }

}
```

10.6 APPSETTINGS.JAVA

```
package saini.findmyphone;
import android.app.Activity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import android.content.SharedPreferences;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;


public class AppSettings extends Activity {

            SharedPreferences sp ;

            EditText et_vol, et_loc;

 @Override

protected void onCreate(Bundle savedInstanceState) {

            // TODO Auto-generated method stub

            super.onCreate(savedInstanceState);

            sp = getSharedPreferences("Findmyphone", MODE_PRIVATE);

String vol=sp.getString("pref_Volume", "Volume");

            String loc=sp.getString("pref_Location","Location");

            setContentView(R.layout.app_settings);

            et_vol=  (EditText)findViewById(R.id.edit1);

            et_loc= (EditText) findViewById(R.id.edit2);

            et_vol.setText(vol);

            et_loc.setText(loc);

}
public void onCancel(View V)
{
        finish() ;
}
 public void onSave(View v)
{
                sp = getSharedPreferences("Findmyphone", MODE_PRIVATE);
                SharedPreferences.Editor sp_edit = sp.edit();
                String ch_vol, ch_loc ;
                ch_vol = et_vol.getText().toString();
                ch_loc = et_loc.getText().toString();
                if (ch_vol.length()> 0)
                sp_edit.putString("pref_Volume", ch_vol);
                if (ch_loc.length()> 0)
                sp_edit.putString("pref_Location", ch_loc);
                sp_edit.commit();
                Toast.makeText(this,"Saved..", Toast.LENGTH_LONG).show();
                finish();


 }
}
```

CHAPTER 11

OUTPUT AND SNAPSHOTS



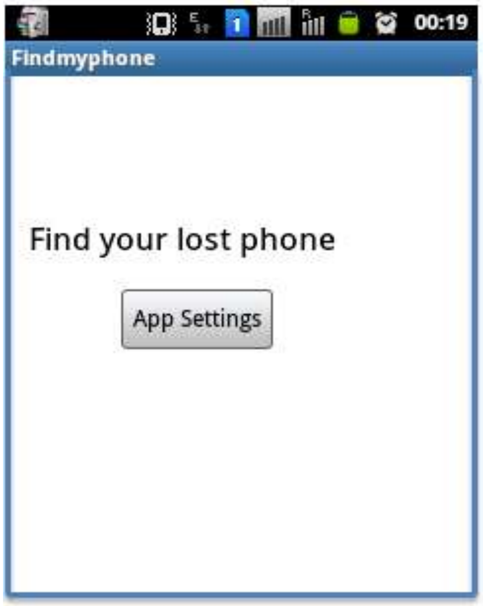
Do you want to install this application?

Allow this application to:

✓ **Your messages**
read SMS or MMS, receive SMS

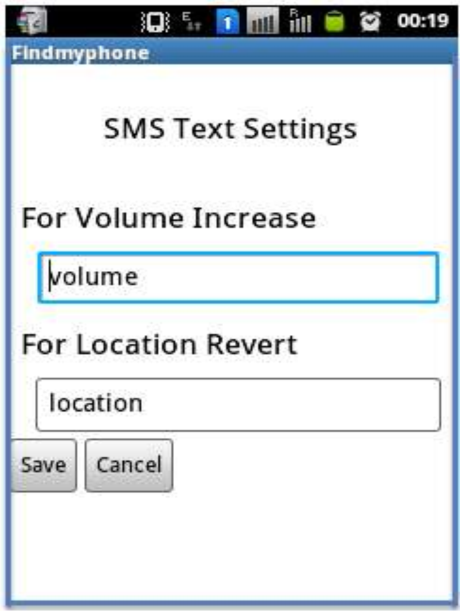
✓ **Your location**
coarse (network-based) location, fine (GPS) location, mock location sources for testing

Snapshot 1: Installing the app on your phone



Find your lost phone

Snapshot 2: Welcome Screen of the app



SMS Text Settings

For Volume Increase

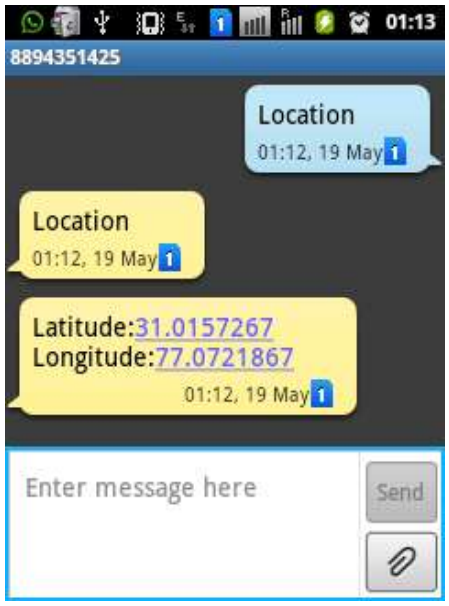
volume

For Location Revert

location

Snapshot 3: Setting the attention word for

volume and location by the user



Location
01:12, 19 May

Location
01:12, 19 May

Latitude:31.0157267
Longitude:77.0721867
01:12, 19 May

Snapshot 4: Receiving the location after sending alert message

CHAPTER 12

CONCLUSION

Find your android phone is a unique & efficient application, which is used to track the lost/ misplaced android phone.

All the features work on SMS basis. Therefore, incoming SMS format plays a vital role. Our android application running in the cell monitors all the incoming messages. If the SMS is meant for the application, it reads the same and performs the expected task.

I have created features, which will enhance the existing cell tracking system. Application stands different from the existing system which makes the application a simple & unique one.

REFERENCES

1. Hello Android, the Pragmatic Programmers (2009),E. Burnette.

2. Professional Android 2 Application Development, R. Meier, Wiley (2010).

3. Beginning Android 2, M. Murphy, Apress (2010).

4. Android A programmers guide Jerome DiMarzio

5. Android Developer Guide: http://developer.android.com/guide/index.html.

6. Android API: http://developer.android.com/reference/packages.html