INTRODUCTION

1.1 Overview

Android is a Linux based operating system designed primarily for touchscreen mobile devices such as Smartphone and tablet computers. Android is open source and Google releases the code under the Apache License. This open source code and permissive licensing allows the software to be freely modified and distributed by device manufacturers, wireless carriers and enthusiast developers. Additionally, Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in a customized version of the Java programming language.

These factors have contributed towards making Android the world's most widely used Smartphone platform, overtaking Symbian in the fourth quarter of 2010, and the software of choice for technology companies who require a low-cost, customizable, lightweight operating system for high tech devices without developing one from scratch. As a result, despite being primarily designed for phones and tablets, it has seen additional applications on televisions, games consoles, digital cameras and other electronics. Android's open nature has further 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.

Currently Android is the most popular mobile operating system in the world, it is using in lot of domains like Business, Education, Entertainment, Medical, etc. All users for this mobile stores lot many useful, important, personnel, and general data in the phone memory. If your mobile is lost, we assume you have a backup of all these data, but that person can able to access those data from your phone and he may misuse those data, but now with our proposed application in the Android phone, there should be no more worrying of misusing and unauthorized accessing of data once the device is lost.

1.2 Basics of Android applications

- Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers).

- Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications).

- The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required.

- Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages)

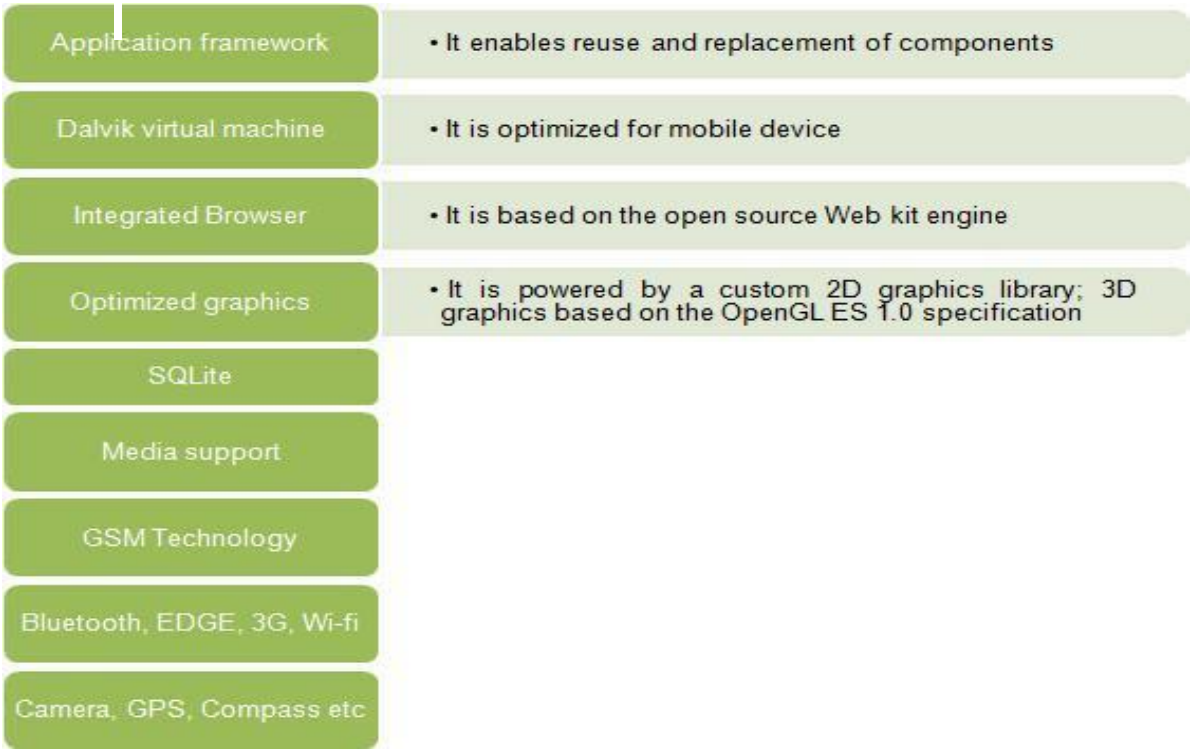

Figure 1.1: ARM Architecture Platform

LITERATURE SURVEY

2.1 Java

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java, GNU Classpath and Dalvik.

Sun Microsystems released the first public implementation as Java 1.0 in 1995. It promised "Write Once, Run Anywhere", providing no-cost run-times on popular platforms. Fairly secure and featuring configurable security, it allowed network- and file-access restrictions. Major web browsers soon incorporated the ability to run Java applets within web pages, and Java quickly became popular. With the advent of Java 2 (released initially as J2SE 1.2 in December 1998–1999), new versions had multiple configurations built for different types of platforms. For example, J2EE targeted enterprise applications and the greatly stripped-down version J2MEfor mobile applications (Mobile Java). J2SE designated the Standard Edition. In 2006, for marketing purposes, Sun renamed new J2 versions as Java EE, Java ME, and Java SE, respectively.

On November 13, 2006, Sun released much of Java as open source software under the terms of the GNU General Public License (GPL). On May 8, 2007, Sun finished the process, making all of Java's core code available under free software/open-source distribution terms, aside from a small portion of code to which Sun did not hold the copyright. Sun's vice-president Rich Green has said that Sun's ideal role with regards to Java is as an "evangelist."

2.1.1 Principles

There were five primary goals in the creation of the Java language

1. It should be "simple, object oriented and familiar".
2. It should be "robust and secure".
3. It should have "an architecture-neutral and portable environment".
4. It should execute with "high performance".
5. It should be "interpreted, threaded, and dynamic".

2.1.2 Java Features

A) Platform Independent

The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

B) Simple

There are various features that make the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we cannot say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and de allocation system.

C) Object-Oriented

To be an Object Oriented language, any language must follow at least the four characteristics.

- Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
- Encapsulation: It is the mechanism of combining the information and providing the abstraction.
- Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.
- Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

As the languages like Objective C, C++ fulfills the above four characteristics yet they are not fully object oriented languages because they are structured as well as object oriented languages. But in case of java, it is a fully Object Oriented language because object is at the outer most level of data structure in java. No stand alone methods, constants, and variables are there in java. Everything in java is object even the primitive data types can also be converted into object by using the wrapper class.

D) Robust

Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features make the java language robust.

E) Distributed

The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

2.2 Apache/ Tomcat

Apache/Tomcat (or Jakarta Tomcat or simply Tomcat) is an open source servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run.

Tomcat should not be confused with the Apache web server, which is a C implementation of an HTTP web server; these two web servers are not bundled together. Apache Tomcat includes tools for configuration and management, but can also be configured by editing XML configuration files.

Tomcat started off as a servlet reference implementation by James Duncan Davidson, a software architect at Sun Microsystems. He later helped make the project open source and played a key role in its donation by Sun to the Apache Software Foundation. The Apache Ant software build automation tool was developed as a side-effect of the creation of Tomcat as an open source project.

2.2.1 Features:

- Implements the Servlet 2.4 and JSP 2.0 specifications.
- Reduced garbage collection, improved performance and scalability.
- Native Windows and UNIX wrappers for platform integration.
- Faster JSP parsing.

2.3 Android

2.3.1 Overview

Android is a software stack for mobile devices that includes an operating system middleware and key applications. Android's mobile operating system is based on a modified version of the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform.

Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 150,000 apps available for Android. Android Market is the online app store run by Google, though apps can also be downloaded from third-party sites. Developers write primarily in the Java language, controlling the device via Google-developed Java libraries.

The Android open-source software stack consists of Java applications running on a Java-based, object-oriented application framework on top of Java core libraries running on a Dalvik virtual machine featuring JIT compilation. Libraries written in C include the surface manager, OpenCore media framework, SQLite relational database management system, OpenGL ES 2.0 3D graphics API, WebKit layout engine, SGL graphics engine, SSL, and Bionic libc. The Android operating system, including the Linux kernel, consists of roughly 12 million lines of code including 3 million lines of XML, 2.8 million lines ofC, 2.1 million lines of Java, and 1.75 million lines of C++.

2.3.2 Android Features

- **Handset layouts**

  The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional smartphone layouts.

- **Storage**

  SQLite, a lightweight relational database, is used for data storage purposes.

- **Connectivity**

  Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, WiFi, LTE, NFC and WiMAX.

- **Messaging**

  SMS and MMS 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.

- **Multiple language support**

  Android supports multiple languages.

- **Web browser**

  The web browser available in Android is based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores 100/100 on the Acid3 test on Android 4.0.

- **Java support**

  While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

2.4 Eclipse

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 and can be used to develop applications in Java and, by means of various plugins, and other programminglanguagesincluding Ada, C, C++, COBOL, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Scala, Clojure, and Scheme. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

The initial codebase originated from Visual Age. In its default form it is meant for Java developers, consisting of the Java Development Tools (JDT). Users can extend its abilities by installing plug-ins written for the Eclipse software framework, 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 is free and open source software. It was one of the first IDEs to run under GNU Classpath and it runs without issues under IcedTea.

2.4.1 History

Eclipse began as an IBM Canada project. It was developed by Object Technology International (OTI) as a Java-based replacement for the Smalltalk based Visual Age family of IDE products, which itself had been developed by OTI. In November 2001, a consortium was formed to further the development of Eclipse as open source. In January 2004, the Eclipse Foundation was created.

Eclipse 3.0 selected the OSGi Service Platform specifications as the runtime architecture. Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

2.4.2 Architecture

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox, an OSGi standard compliant implementation.

This plug-in mechanism is a light weight software component framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins. With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a *workspace*, in this case a set of metadata over a flat file space allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT. Language packs provide translations into over a dozen natural languages.

2.4.3 Rich Client Platform

Eclipse provides the Eclipse Rich Client Platform (RCP) for developing general purpose applications. The following components constitute the rich client platform:

- Equinox OSGi – a standard bundling framework
- Core platform – boot Eclipse, run plug-ins
- Standard Widget Toolkit (SWT) – a portable widget toolkit
- JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors.

2.4.4 Eclipse: The IDE for Unisys Clear Path Developers

By design, Eclipse itself is a deliberately skinny foundation with minimal functionality. It is built to be extended via the concept of plug-ins, which are code bundles that are loaded and used according to a specified configuration. The basic Eclipse structure defines extension points, which allow new modules (plug-ins) that extend the menus and offer more services.

Developers working within the Eclipse IDE can pick and choose the features they want to use from the wide, open-source world of plug-in options. The choices are nearly endless. Unisys is making Eclipse a major focal point for development of ClearPath applications. In fact, we plan to eventually replace current, proprietary solutions, such as Programmer's Workbench, with this open, industry-standard environment.

We have several objectives with our Eclipse offering. The first is to provide industry-standard IDE assistance for developing Java applications that use ClearPath assets. To this end, Unisys provides industry-standard Resource Adapters (RAs) for ClearPath database and transaction access and the Eclipse IDE offers features that assist with development of these applications. For both ClearPath platforms, there is an RA that supports Distributed Transaction Processing Services (DTP RA). For ClearPath OS 2200 environments, we support access to Business Information Server, DMS and RDMS databases, as well as to TIP/HVTIP transactions. For ClearPath MCP, there are RAs for Enterprise Database Server (DMS II) and COMS.

Our second major objective with Eclipse is to provide the ability to develop other 3GL applications for ClearPath environments using the same framework that can be used for Java and other non-ClearPath hosted applications. For ClearPath OS 2200 environments, we enable development of COBOL, Java, and PLUS applications, as well as the use of TelNet and CMplus.

For MCP, we support COBOL 74/85 and ALGOL, as well as the use of WFL. And by moving ClearPath development activities into Eclipse, programming teams will gain the additional benefits of a streamlined build and test process with a common IDE for all projects, regardless of language.

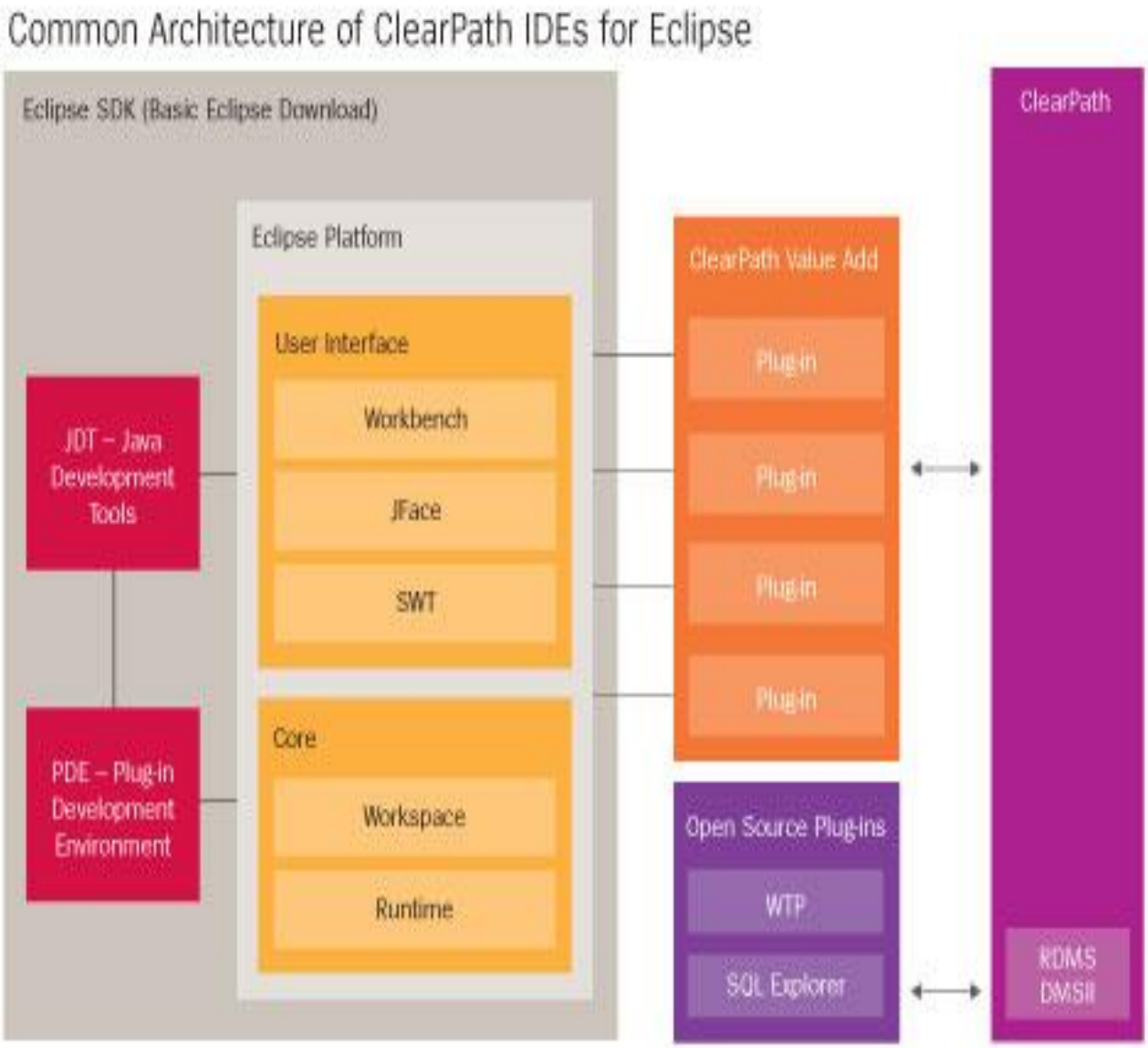

Figure 2.1: Architecture of Eclipse

The Unisys engineering team is fully invested in contributing to the open source movement in ways that benefit our ClearPath customers. We are directing our resources to creating ClearPath specific plug-ins that naturally build on the Eclipse IDE foundation and provide capabilities that make sense for developers on ClearPath systems. To that end, Unisys has built and released Eclipse plug-ins that enable development for ClearPath OS 2200 and MCP operating environments using the Eclipse IDE, as well as integration of ClearPath based assets in a service-oriented environment. So, what is Unisys providing exactly for Eclipse? An All-in-One package comprised of the following: The basic, currently released Eclipse IDE

- A selection of open-source plug-ins that we see as particularly valuable in a ClearPath development environment, such as the Eclipse Data Tools Project and Web Tools Project.
- Unisys plug-ins, some of which are modified versions of open source plug-ins and others built by Unisys, which enable the development of applications specifically for ClearPath and access to ClearPath assets (databases and transactions).

An Application Development Guide to help you get started Equally important, the components of the Unisys Eclipse All-in-One package are:

- Integrated and tested by Unisys engineering
- Packaged and released as a part of the OS 2200 and MCP IOEs
- Supported via Unisys Support at no additional charge.

For more detailed information about the ClearPath OS 2200 IDE for Eclipse, see our Tech Corner article in this issue of ClearPath Connection. (And, look for a similarly in-depth article about the ClearPath MCP IDE for Eclipse in the next issue of this newsletter).

2.4.5 Easy, Integrated, and Innovative

With the Unisys All-in-One packages, it’s never been easier to get started using the Eclipse IDE. Whether you’re an experienced ClearPath programmer or a new college grad just starting out, it’s time to start exploring the technical and business benefits of this powerful toolset. You can take advantage of the many plug-ins available from the open source community and Unisys own plug-ins that are designed to help you get maximum value from your current Clear Path investment.

SYSTEM REQUIREMENTS

3.1 Software Requirements Analysis

A software requirements definition is an abstract description of the services, which the system should provide, and the constraints under which the system must operate. It should only specify only the external behavior of the system and is not concerned with system design characteristics. It is a solution, in a natural language plus diagrams, of what services the system is expected to provide and the constraints under which it must operate.

3.2 Hardware Requirements Analysis

Hardware Requirements Analysis is to define and analyze a complete set of functional, operational, performance, interface, quality factors, and design, criticality and test requirements.3.3 Minimum System Requirements

Hardware requirements:

- Processor: ARM or QUALCOMM Processor (32 bit).
- Ram: 128MB or More.
- Hard Disk: Minimum 200MB.
- Android Mobile Phone.

Software requirements:

- Android SDK 1.5 or More.
- Eclipse IDE.
- Programming language JAVA and XML.
- Operating System Android (Linux Kernel).

ARCHITECTURE

4.1 Problem Definition

The problem is to monitor and control the stolen phone in order to retrieve the important data and wipe it out, and to track the location of the stolen Android device.

4.2 Architecture

In Android, every application runs in its own process, which gives better performance in security, protected memory and other benefits. Therefore, Android is responsible to run and shut down correctly these processes when it is needed.

It is important that application developers understand how different application components (in particular Activity, Service, and BroadcastReceiver) impact the lifetime of the application's process. Not using these components correctly can result in the system killing the application's process while it is doing important work.

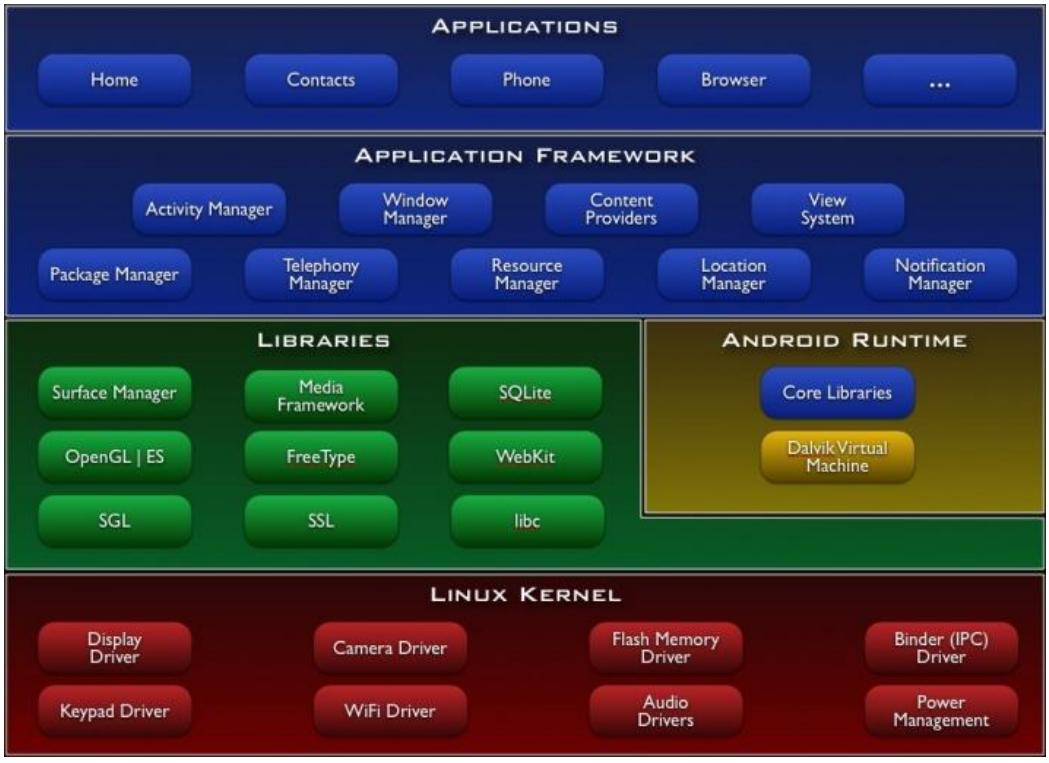

Figure 4.1: ANDROID ARCHITECTURE

To determine which processes should be killed when low on memory, Android places each process into an "importance hierarchy" based on the components running in them and the state of those components.

These process types are (in order of importance):

- A foreground process is one that is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:
    - It is running an Activity at the top of the screen that the user is interacting with (it's on Resume() method has been called).
    - It has a Broadcast Receiver that is currently running (its BroadcastReceiver.onReceive() method is executing).
    - It has a Service that is currently executing code in one of its callbacks (Service.onCreate(), Service.onStart(), or Service.onDestroy()).

There will only ever be a few such processes in the system, and these will only be killed as a last resort if memory is so low that not even these processes can continue to run. Generally, at this point, the device has reached a memory paging state, so this action is required in order to keep the user interface responsive.

- A visible process is one holding an Activity that is visible to the user on-screen but not in the foreground (its onPause() method has been called). This may occur, for example, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it. Such a process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running.
- A service process is one holding a Service that has been started with the startService() method. Though these processes are not directly visible to the user, they are generally doing things that the user cares about (such as background mp3 playback or background network data upload or download), so the system will always keep such processes running unless there is not enough memory to retain all foreground and visible process.

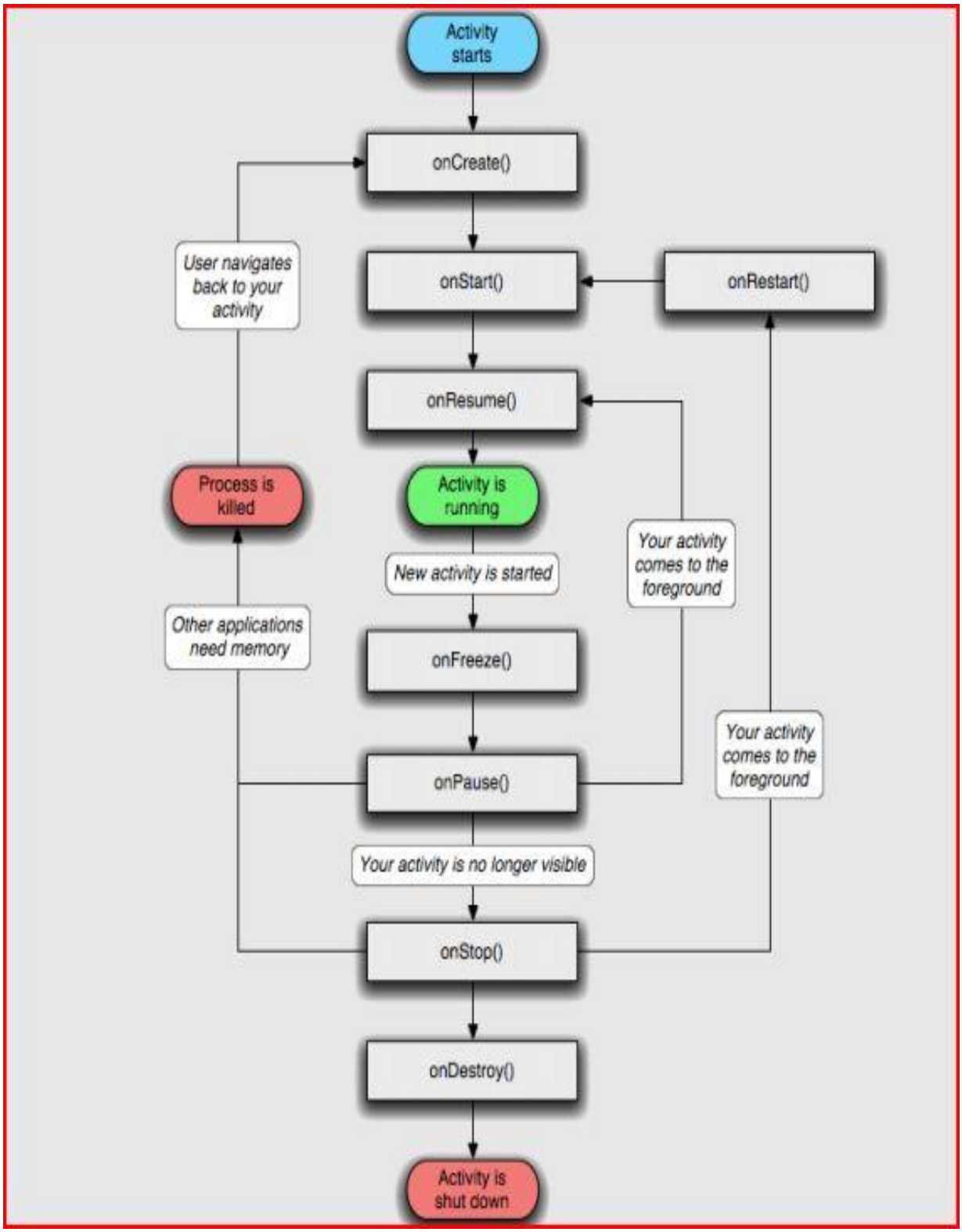

Figure 4.2: Flowchart Showing the Lifecycle of an Application Activity

- A background process is one holding an Activity that is not currently visible to the user (its onStop() method has been called). These processes have no direct impact on the user experience. Provided they implement their Activity life-cycle correctly (see Activity for more details), the system can kill such processes at any time to reclaim memory for one of the three previous processes types. Usually there are many of these processes running, so they are kept in an LRU list to ensure the process that was most recently seen by the user is the last to be killed when running low on memory.
- An empty process is one that doesn't hold any active application components. The only reason to keep such a process around is as a cache to improve startup time the next time a component of its application needs to run. As such, the system will often kill these processes in order to balance overall system resources between these empty cached processes and the underlying kernel caches.

In the following example we will display a process flow from the Android System point of view to get a clear idea how the applications behave. Let assume the

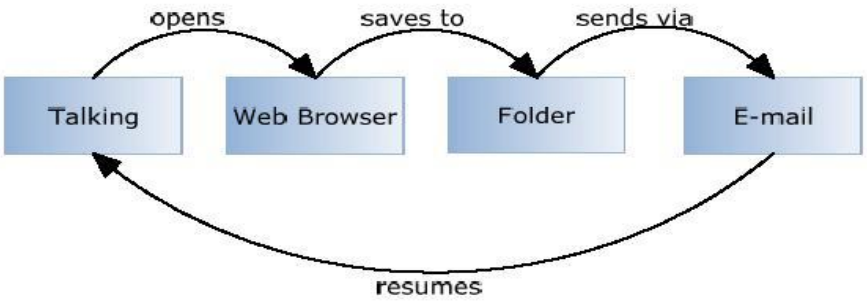

Figure 4.3: Android System point of view of application behavior

Possible scenario: A user talks to his friend via mobile phone and he is asked to browse the internet (a talk is hold for a moment), find a picture of him in his Picasa Album, send it via Email back to his friend and resume a talk.

In this situation, there are 4 different applications and 4 different processes running, but from the user point of view none of them are important, as Android manages CPU work and memory usage by itself. It means the user can travel through the applications forward and back without thinking about how much memory is left or which processes are run at the time. Firstly, as the

user is talking to his friend, a specific Talk application is opened, which contains the activity manager. In the following stack we can see two processes running, the main system process and Talk application process. Moreover, before going to Web Browser application, the system saves a Talk state T in order to remember that process:

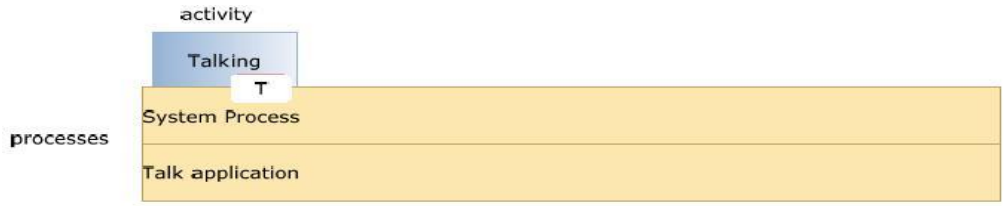

Figure 4.4: Simultaneous running processes

At this point, as a user holds a talk and opens a web browser, the system creates a new process and new web browser activity is launched in it. Again, the state of the last activity is saved (W):

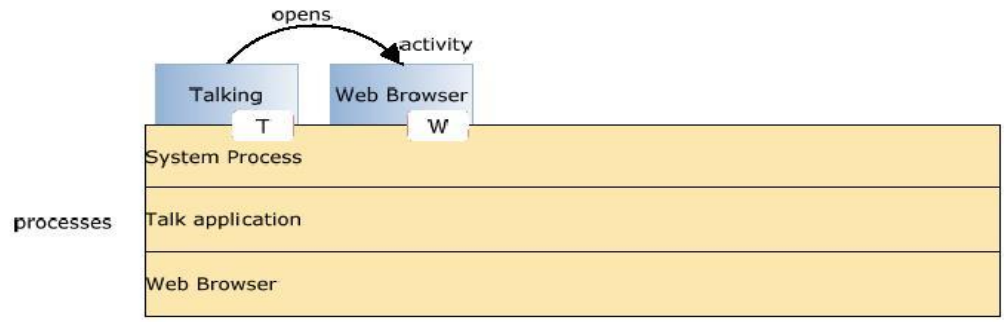

Figure 4.5: Holding one process and launch another

After that, the user browses the internet, finds his picture in Picasa album and saves it to particular folder. He does not close a web browser, instead he opens a folder to find saved picture. The folder activity is launched in particular process:

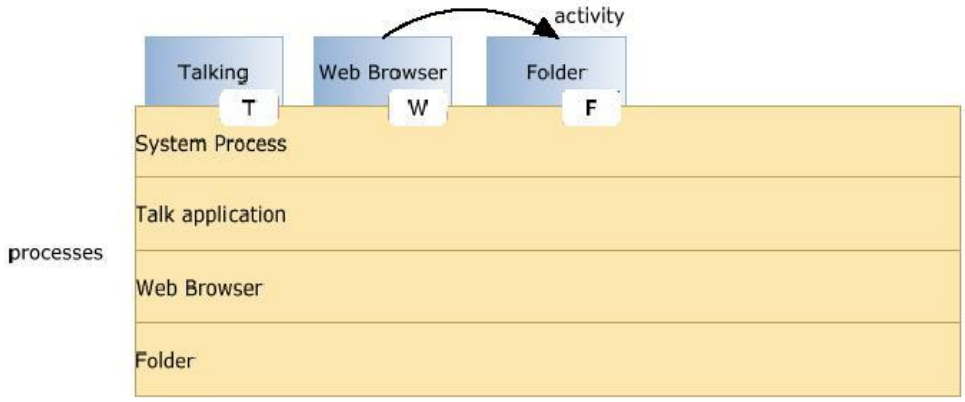

Figure 4.6: Folder Activity

At this point, the user finds his saved picture in the folder and he creates a request to open an Email application. The last state F is saved. Now assume that the mobile phone is out of the memory and there is no room to create a new process for Email application. Therefore, Android looks to kill a process. It cannot destroy Folder process, as it was used previously and could be reused again, so it kills Web Browser process as it is not useful anymore and locates a new Email process instead:

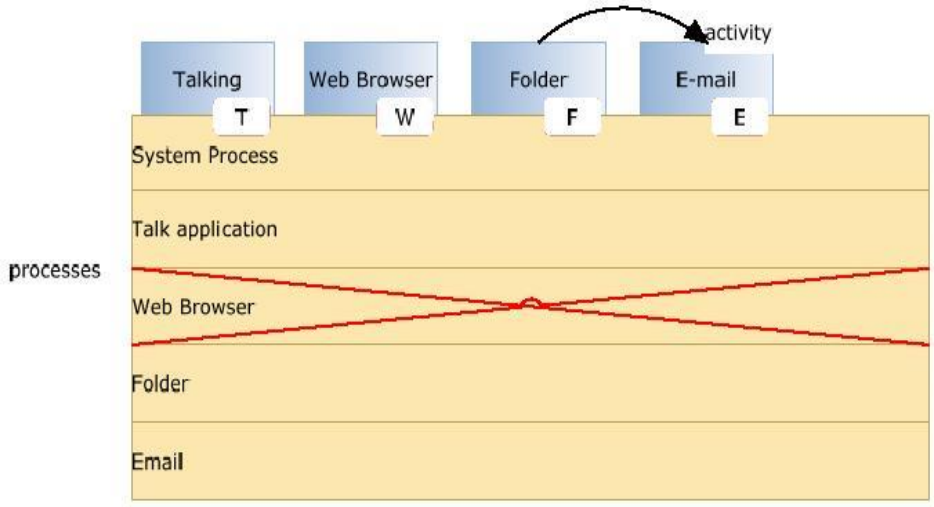

Figure 4.7 Killing an activity and launching a new one

The user opens Email application and sends a picture to his friend via email. Now he wants to go back to the Talk application and to resume a talk to his friend. Because of the previously saved states, this work is done fast and easily. In this example, Email application is popped out and the user sees a previous Folder application:

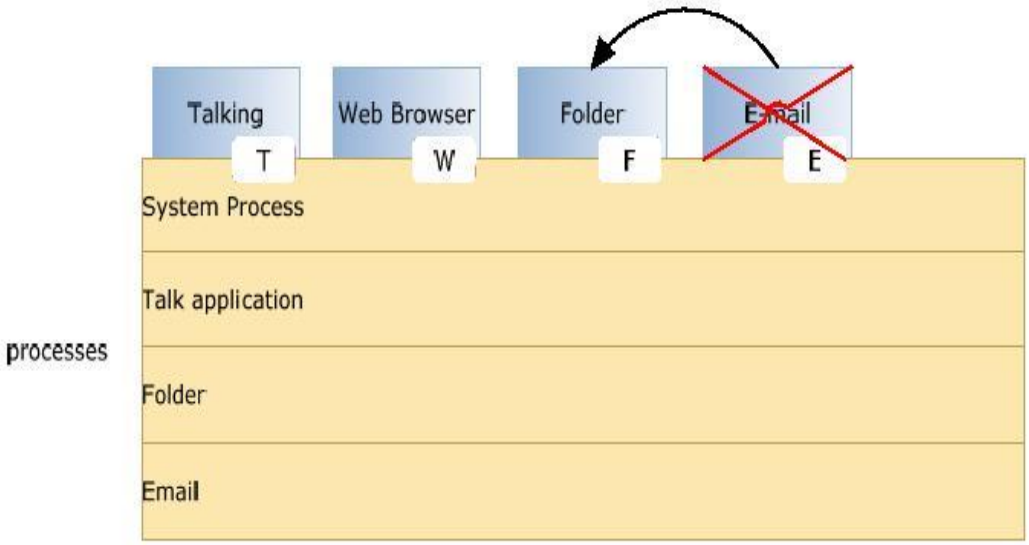

Figure 4.8: Management of Stack Memory

Next, the user goes back to Web Browser application. Unfortunately, web browser process was killed previously so the system has to kill another process (in our case it is Email application process, which is not used anymore) in order to locate Web Browser process and manage the stack memory:

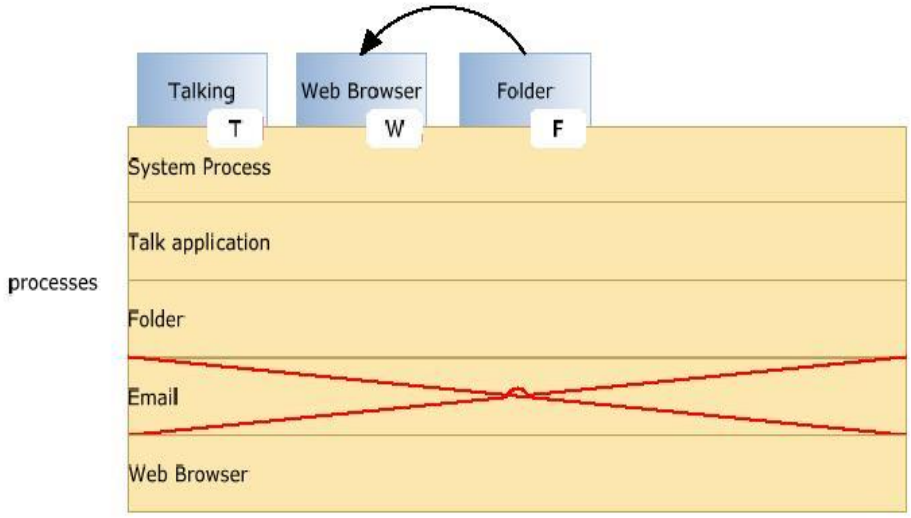

Figure 4.9: Switching between processes

And finally:

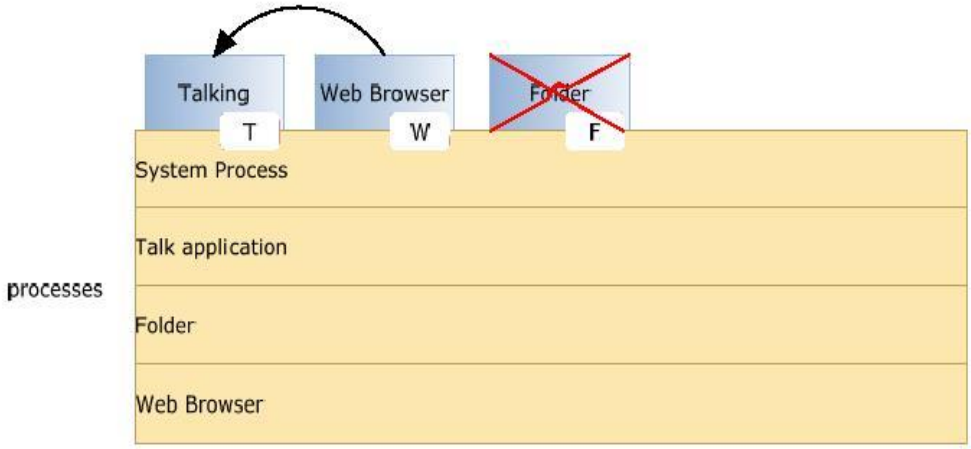

Figure 4.10: Saved states concept

Now the user comes back to the Talk application and resumes his talk with his friend. Because of the saved states, going back procedure is fast and useful, because it remembers previous activities and its views.

This example shows, that it does not matter how many applications and processes are active or how much available memory is left, Android it manages fast and without any user interaction.

SYSTEM DESIGN

The word system is possibly the most overused and abused term in the technical lexicon. System can be defined as the "a set of fact, principles, rules etc., classified and arranged in an orderly form so as to show a logical plan linking the various parts" here the system design defines the computer based information system. The primary objective is to identify user requirements and to build a system that satisfies these requirements.

Design is much more creative process than analysis. Design is the first step in the development of any system or product. Design can be defined as "the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization".

It involves four major steps they are:

1. Understanding how the system is working now;
2. Finding out what the system does now;
3. Understanding what the new system will do; and
4. Understanding how the new system will work.

So as to avoid these difficulties, a new system was designed to keep these requirements in mind. Therefore the manual process operation has been changed into GUI based environment, such that the user can retrieve the records in a user-friendly manner and it is very easy to navigate to the corresponding information.

5.1 Input Design

Input design is the bridge between users and information system. It specifies the manner in which data enters the system for processing it can ensure the reliability of the system and produce reports from accurate data or it may results in output of error information.

5.2 Output Design

Outputs from the computer system are rewired primary to communicate the results of processing to the uses. They also used to provide a permanent copy of these results for later consultation / verification. The main points on designing an output are deciding the

media, designing layout and report to be printed. The outputs are designed from the system, are simple to read and interpret.

5.3 Data Flow Diagram

A DFD is a logical model of the system. The model does not depend on the hardware, software and data structures of file organization. It tends to be easy for even non-technical users to understand and thus serves as an excellent communication tool.

DFD can be used to suggest automatic boundaries for proposed system at pa very high level; the entire system is shown as a single logical process clearly identifying the sources and destination of data. This is often referred to as zero level DFD.

Then the processing is exploded into major processes and the same is depicted as level one DFD.

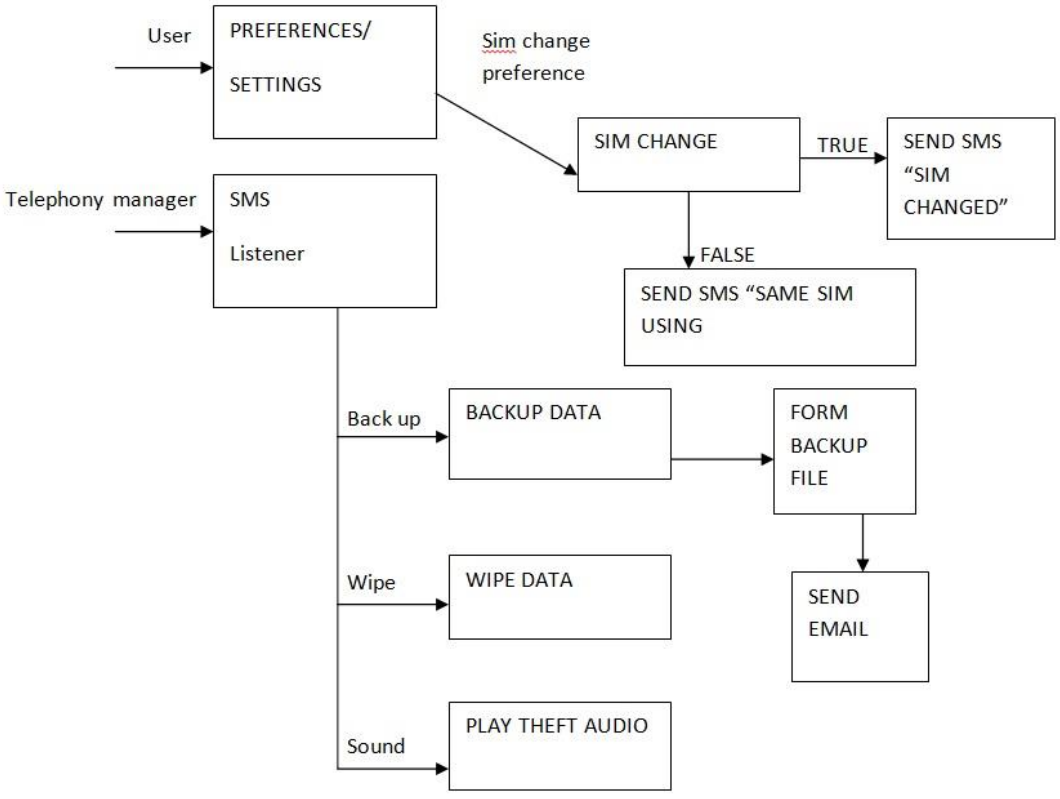

Figure 5.1: Data Flow Diagram

5.4 Flow Chart

Step 1: User installs the application on his Android phone and the SIM serial number will get registered immediately.

Figure 5.2: Application Installation on Android device

Step 2: Every time the phone is rebooted, or in an occurrence of a theft, the following activities will automatically occur:

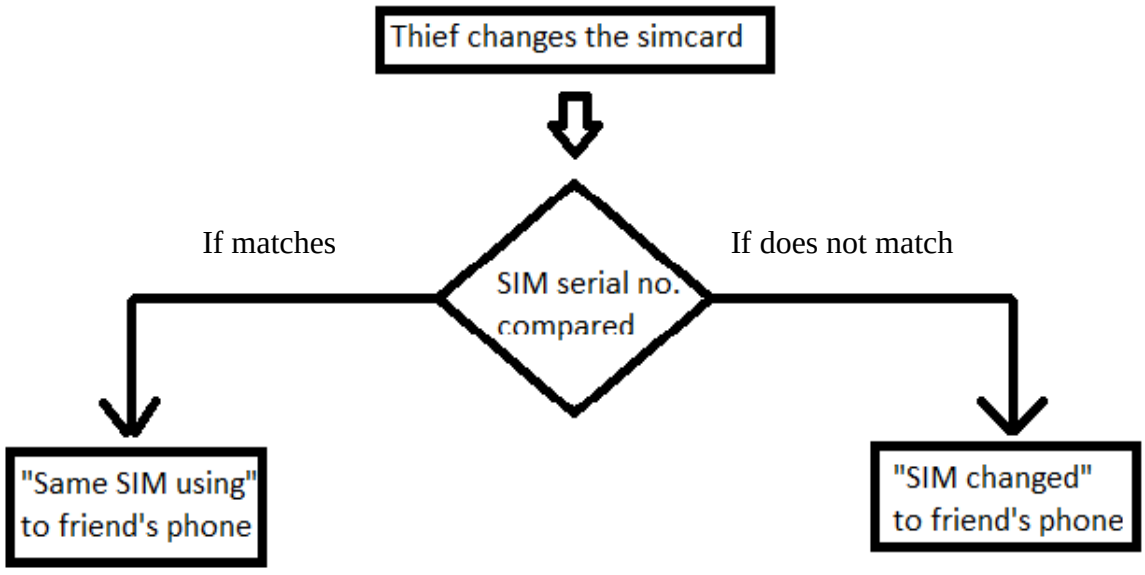

Figure 5.3: Comparison of SIM card serial numbers.

5.5 Sequence Diagrams

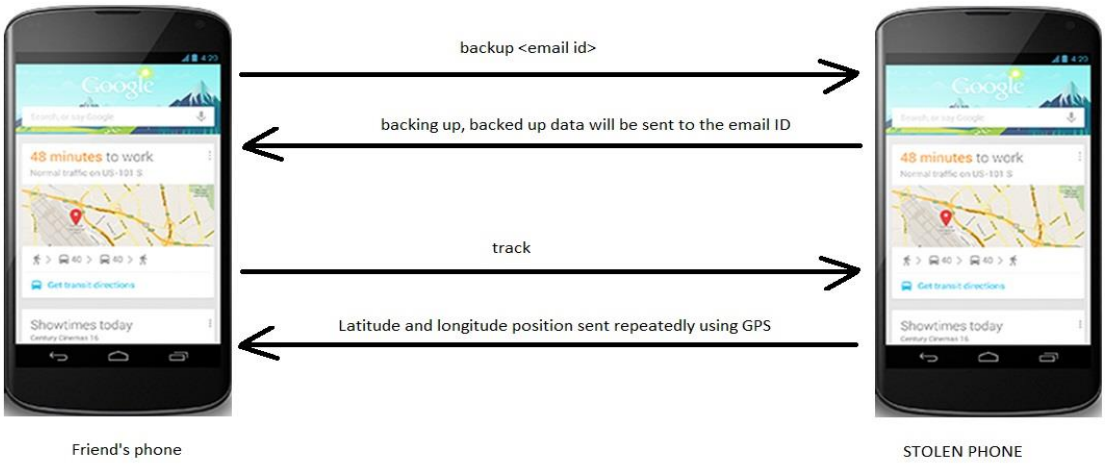

Figure 5.4: Message Activity Sequences 1

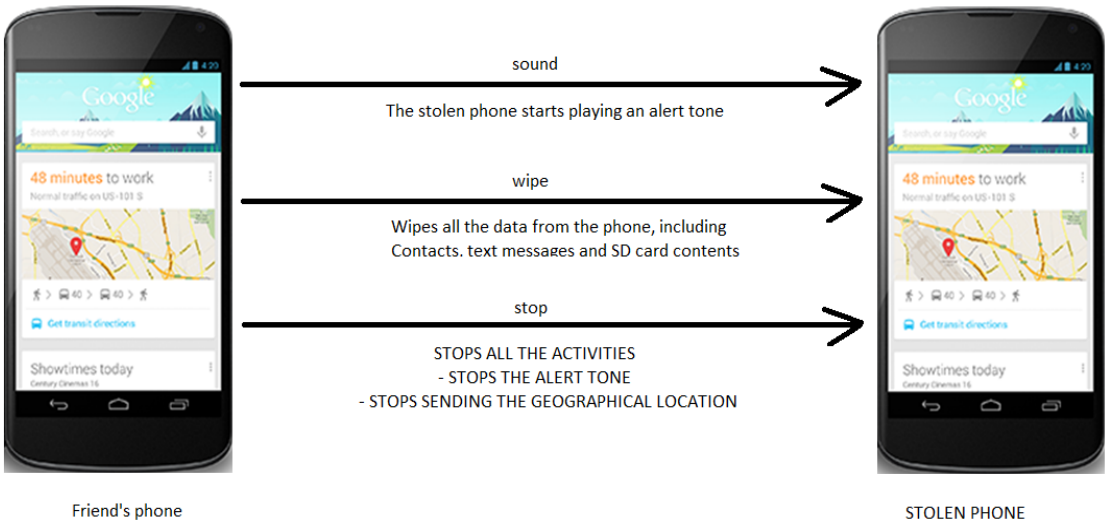

Figure 5.5: Message Activity Sequences 2

IMPLEMENTATION

Once the system has been designed, the next step is to convert the designed one in to actual code, so as to satisfy the user requirements as excepted. If the system is approved to be error free it can be implemented.

When the initial design was done for the system, the department was consulted for acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about working of the system. The aim of the system illustration was to identify any malfunctioning of the system.

Implementation includes proper training to end-users. The implemented software should be maintained for prolonged running of the software.

Initially the system was run parallel with manual system. The system has been tested with data and has proved to be error-free and user-friendly. Training was given to end -user about the software and its features.

In the coding part of our implementation, we first start with the file AndroidManifest.xml, which is the main and initial file to be created for an Android application.

6.1 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.sms.filesharing"
android:versionCode="1"
android:versionName="1.0"
android:installLocation="internalOnly"  >
<uses-sdk android:minSdkVersion="8" />
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.SEND_SMS">
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
 <uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.WRITE_CONTACTS" />
   <uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.WRITE_SMS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION">
</uses-permission>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"></usespermission>
 <uses-permission android:name="android.permission.MODIFY_PHONE_STATE"/>
 <uses-permission android:name="android.permission.CAMERA"></uses-permission>
 <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
 <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED"></uses-permission>
<application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
        android:name=".SMSFileSharingActivity"
        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="About"
          android:label="@string/app_name" />
        </activity>
```

```
        <activity
          android:name="OneShotPreviewActivity"
          android:label="@string/app_name" />
         <receiver android:name=".smsReceiver">
        <intent-filter android:priority="5000">
        </activity>
        <action
         android:name="android.provider.Telephony.SMS_RECEIVED" />
         </intent-filter>
         </receiver>
         <activity
         android:name=".Reminder"
         android:label="@string/app_name" />
        <service
         android:name=".MyAlarmService"
          />
         <activity
         android:name="History"
         android:label="@string/app_name" />
         <receiver android:name="MyStartupIntentReceiver">
         <intent-filter>
         <action
         android:name="android.intent.action.BOOT_COMPLETED" />
         <category android:name="android.intent.category.DEFAULT" />
         </intent-filter>
         </receiver>
</application>
</manifest>
```

6.2 SMSFileSharingActivity.java

```
package com.sms.filesharing;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
```

```
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class SMSFileSharingActivity extends Activity
{
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState)
       {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      TelephonyManager mTelephonyMgr =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
      final String no = mTelephonyMgr.getSimSerialNumber();
      SharedPreferences dbevent=getSharedPreferences("pwd", MODE_PRIVATE);
                Editor editor=dbevent.edit();
                editor.putString("password", no);
                editor.commit();
                Toast.makeText(this, "serial no stored "+no, 600).show();
          final EditText et1 = (EditText) findViewById(R.id.editText1);
          final EditText et2 = (EditText) findViewById(R.id.editText2);
          Button btn = (Button) findViewById(R.id.btn);
          btn.setOnClickListener(new OnClickListener() {
                        @Override
              public void onClick(View v) {
                                // TODO Auto-generated method stub
                                // store pwd in app memory
         SharedPreferences dbevent=getSharedPreferences("pwd", MODE_PRIVATE);
                Editor editor=dbevent.edit();
              editor.putString("password", no);
              editor.commit();
                SharedPreferences dbevent1=getSharedPreferences("EmailId",
MODE_PRIVATE);
                Editor editor1=dbevent1.edit();
                editor1.putString("emailId", et1.getText().toString());
                                editor1.commit();
                SharedPreferences dbevent2=getSharedPreferences("EmailIdpwd",
MODE_PRIVATE);
                Editor editor2=dbevent2.edit();
                editor2.putString("emailIdpwd", et2.getText().toString());
                editor2.commit();

            Toast.makeText(getApplicationContext(), "updated", 6000).show();
                        }});
```

```
      List<File> match=searchFullSDCard("dad21.jpg");
      Toast.makeText(getApplicationContext(), "THe count is "+match.size(), 6000).show();
      }

    public boolean onCreateOptionsMenu(Menu m)
     {
     return true;
     }
   public boolean onOptionsItemSelected(MenuItem itm)
    {
     return true;
    }
   public static List<File> searchFullSDCard(String match)
    {
      List<File> matches = new ArrayList<File>();
      File f = new File("/sdcard/");
      findMatch(matches,f,match);
      return matches;
    }
   public static List<File>searchFromDirectory(String match,File dir)
     {
       List<File> matches = new ArrayList<File>();
       findMatch(matches,dir,match);
       return matches;

private static void findMatch(List<File>matches,File curDir,String match)
   {
       File[]files = curDir.listFiles();
       if(files==null)
       return;
       for(File f: files)
        {
          if(f.isDirectory())
          findMatch(matches,f,match);
          else
           {
            if(f.getName().toLowerCase().contains(match.toLowerCase()))
                          matches.add(f);
                }
          }
   }
}
```

6.3 MyStartUpIntentReceiver.java

```
package com.sms.filesharing;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;
import android.telephony.TelephonyManager;
import android.telephony.gsm.SmsManager;
import android.widget.Toast;

public class MyStartupIntentReceiver extends BroadcastReceiver{
@Override
public void onReceive(Context context, Intent intent) {

        System.out.println("**inside onRecevier");
        Toast.makeText(context, "sending message...", 600000).show();
        SmsManager sm = SmsManager.getDefault();
        SharedPreferences dbevent=context.getSharedPreferences("pwd",
context.MODE_PRIVATE);
        String pwd = dbevent.getString("password", "1234");
        Toast.makeText(context, "the stored values is "+pwd, 600).show();
        TelephonyManager mTelephonyMgr =
(TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
        String id = mTelephonyMgr.getSimSerialNumber();
        Toast.makeText(context, "executing theft", 600).show();
        SharedPreferences dbevent3=context.getSharedPreferences("EmailId",
context.MODE_PRIVATE);
        if(id.equals(pwd)){
                Toast.makeText(context, "sim no equals", 600).show();
     sm.sendTextMessage(dbevent3.getString("emailId", "9916683186"),null,"same sim
using",null,null);
                sm.sendTextMessage(dbevent3.getString("emailId", "8884059819"),null,"same
sim using",null,null);
        }
else{
    Toast.makeText(context, "sim no not equals", 600).show();
    sm.sendTextMessage(dbevent3.getString("emailId", "9916683186"),null,"sim has been
changed",null,null);
    sm.sendTextMessage(dbevent3.getString("emailId", "8884059819"),null,"same sim
using",null,null);
        }
        // String pno=LbsGeocodingActivity.phoneno;
  }
}
```

6.4 GmailSenderActivity.java

```
package com.sms.filesharing;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.Security;
import java.util.List;
import java.util.Properties;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.activation.FileDataSource;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.Multipart;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

public class GMailSender extends javax.mail.Authenticator {
   private String mailhost = "smtp.gmail.com";
   private String user;
   private String password;
   private Session session;
   private Multipart _multipart=new MimeMultipart();

   static {
      Security.addProvider(new com.sms.filesharing.JSSEProvider());
   }
   public GMailSender(String user, String password) {
      this.user = user;
      this.password = password;
      Properties props = new Properties();
      props.setProperty("mail.transport.protocol", "smtp");
      props.setProperty("mail.host", mailhost);
      props.put("mail.smtp.auth", "true");
      props.put("mail.smtp.port", "465");
      props.put("mail.smtp.socketFactory.port", "465");
      props.put("mail.smtp.socketFactory.class",
            "javax.net.ssl.SSLSocketFactory");
      props.put("mail.smtp.socketFactory.fallback", "false");
```

```
        props.setProperty("mail.smtp.quitwait", "false");

        session = Session.getDefaultInstance(props, this);
    }

        protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(user, password);
    }

        public void addAttachment(String filename,String subject) throws Exception {
        BodyPart messageBodyPart = new MimeBodyPart();
        DataSource source = new FileDataSource(filename);
        messageBodyPart.setDataHandler(new DataHandler(source));
        messageBodyPart.setFileName(filename);
        multipart.addBodyPart(messageBodyPart);
        BodyPart messageBodyPart2 = new MimeBodyPart();
        messageBodyPart2.setText(subject);
        multipart.addBodyPart(messageBodyPart2);
    }

        public synchronized void sendMail(String subject, String body, String sender, String
recipients,List<File> filepath,String fileloc) throws Exception {
        try{

        MimeMessage message = new MimeMessage(session);
        DataHandler handler = new DataHandler(new
ByteArrayDataSource(body.getBytes(), "text/plain"));
        message.setSender(new InternetAddress(sender));
        message.setSubject(subject);
        message.setDataHandler(handler);
        if (recipients.indexOf(',') > 0)
        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(recipients));
        else
        message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipients));
        if(fileloc.equals("")){
         for(File f: filepath){
         addAttachment(f.getAbsolutePath(), subject);
           }
       }
else {
        addAttachment(fileloc, subject);
        }

        message.setContent(_multipart);
        Transport.send(message);
```

```
        }
        catch(Exception e){
        System.out.println("the error is "+e.getMessage());
        }
    }
        public class ByteArrayDataSource implements DataSource {
        private byte[] data;
        private String type;
        public ByteArrayDataSource(byte[] data, String type) {
        super();
        this.data = data;
        this.type = type;
         }
        public ByteArrayDataSource(byte[] data) {
        super();
        this.data = data;
        }
        public void setType(String type) {
          this.type = type;
        }

        public String getContentType() {
            if (type == null)
                return "application/octet-stream";
            else
                return type;
        }

        public InputStream getInputStream() throws IOException {
            return new ByteArrayInputStream(data);
        }

        public String getName() {
            return "ByteArrayDataSource";
        }

        public OutputStream getOutputStream() throws IOException {
            throw new IOException("Not Supported");
        }
    }
}
```

6.5 JSSEProvider.java

```
package com.sms.filesharing;
import java.security.AccessController;
import java.security.Provider;
public final class JSSEProvider extends Provider {
public JSSEProvider() {
      super("HarmonyJSSE", 1.0, "Harmony JSSE Provider");
      AccessController.doPrivileged(new java.security.PrivilegedAction<Void>() {
      public Void run() {
      put("SSLContext.TLS",
      "org.apache.harmony.xnet.provider.jsse.SSLContextImpl");
       put("Alg.Alias.SSLContext.TLSv1", "TLS");
       put("KeyManagerFactory.X509",
       "org.apache.harmony.xnet.provider.jsse.KeyManagerFactoryImpl");
       put("TrustManagerFactory.X509",
       "org.apache.harmony.xnet.provider.jsse.TrustManagerFactoryImpl");
       return null;
          }
       });
   }
}
```

6.6 SMSReceiver.java

```
package com.sms.filesharing;
import java.io.File;
import java.io.IOException;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.Cursor;
import android.location.Address;
import android.location.Criteria;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.media.MediaPlayer;
import android.net.ConnectivityManager;
import android.net.Uri;
import android.os.Bundle;
```

```
import android.provider.ContactsContract;
import android.telephony.SmsManager;
import android.telephony.gsm.SmsMessage;
import android.util.Log;
import android.widget.Toast;

public class smsReceiver extends BroadcastReceiver{

        String incomingno,body;
        static LocationManager locationManager;
        SmsManager sm;
        Context cntxt;
        static MyLocationListener locationListener;
        String pwd,emailid,emailidpwd;
        List<File> sample;
        GMailSender sender;
        String reqtime;
        long time;
        @Override
        public void onReceive(Context context, Intent intent) {
                // TODO Auto-generated method stub
                Bundle bundle = intent.getExtras();
                cntxt=context;
                 Toast.makeText(context, "message received", 600000).show();
      SmsMessage[] msgs = null;
      String str = "";
      SharedPreferences dbevent=context.getSharedPreferences("pwd",
context.MODE_PRIVATE);
      SharedPreferences dbevent3=context.getSharedPreferences("EmailId",
context.MODE_PRIVATE);
      SharedPreferences dbevent4=context.getSharedPreferences("EmailIdpwd",
context.MODE_PRIVATE);

      pwd = dbevent.getString("password", "1234");

      emailid = dbevent3.getString("emailId", "dayanand.lokesh@gmail.com");
      emailidpwd = dbevent4.getString("emailIdpwd", "achievement123!@#$%^");

      if (bundle != null)
            {
                //---retrieve the SMS message received---
                Object[] pdus = (Object[]) bundle.get("pdus");
                msgs = new SmsMessage[pdus.length];
                abortBroadcast();
                for (int i=0; i<msgs.length; i++){
                        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
```

```
                incomingno = msgs[i].getOriginatingAddress();
                body= msgs[i].getMessageBody().toString();
                sm = SmsManager.getDefault();
                time = msgs[i].getTimestampMillis();

                  if(body.equalsIgnoreCase("sound")){

                      MediaPlayer mp = new MediaPlayer();
                          try {

                                    mp.setDataSource("/sdcard/voice.3ga");
        mp.prepare();
             mp.start();

        } catch (IllegalArgumentException e) {
       // TODO Auto-generated catch block
        e.printStackTrace();

       } catch (IllegalStateException e) {
      // TODO Auto-generated catch block
       e.printStackTrace();
       } catch (IOException e) {
       // TODO Auto-generated catch block
      e.printStackTrace();
                                  }

        }else if(body.contains("backup"))

{
        String[] temp = body.split(" ");

                setMobileDataEnabled(cntxt,true);
                File dir=new File("/sdcard/theftcontrol/");
                        File[] files=dir.listFiles();
                        String[] path=new String[files.length];
                        if(files.length==0){

        }
else
{
 for(int j=0;j<files.length;j++)
{
path[j]=files[j].getAbsolutePath();
 }
List<File> e;
Compress c = new Compress(path, "/sdcard/zipname.zip");
```

```
            c.zip();
            File file = new File("/sdcard/zipname.zip");
            List<File> sample = new ArrayList<File>();
            sample.add(file);
            SmsManager sm = SmsManager.getDefault();
            sm.sendTextMessage(incomingno, null, "Backing up", null, null);
             try
                 {
 GMailSender sender = new GMailSender("niteshdoc71@gmail.com", "vikasgoyal")
sender.sendMail("from android app",
              "Test mail from android app",
                 "niteshdoc71@gmail.com",
                 temp[1],sample,"/sdcard/zipname.zip");
                       }
catch (Exception e1)
        {
                    Log.e("SendMail", e1.getMessage(), e1);
             }

                    Toast.makeText(cntxt, "backup did ", 600).show();
         }

}
else if(body.equalsIgnoreCase("wipe"))
     {
            File file=new File("/sdcard/theftcontrol/");
            //deleteDirectory(file);
             File[] files = file.listFiles();
         for(int t=0;t<files.length;t++){
            files[t].delete();
          }
                wipesms();
                contactwipe();

}
else if(body.equalsIgnoreCase("track"))
{
     final Criteria criteria = new Criteria();
     locationManager = (LocationManager)
cntxt.getSystemService(Context.LOCATION_SERVICE);

     final String bestProvider = locationManager.getBestProvider(criteria, true);

     locationListener=new MyLocationListener();
     locationManager.requestLocationUpdates(
     bestProvider, 30000, 0, locationListener);
```

```
 }
else if(body.equalsIgnoreCase("stop"))
{
          sm.sendTextMessage(incomingno, null, "stoping gps", null, null);

          if(locationManager!=null){
      locationManager.removeUpdates(locationListener);
          locationManager =  null;
          locationListener = null;

          }
       }
    }
  }
}

long lid;
String phoneNumber;
public void DeleteSms()
{
                    Uri deleteUri = Uri.parse("content://sms");
                    cntxt.getContentResolver().delete(deleteUri, "address=?", new String[]
{incomingno});
                    Toast.makeText(cntxt, "Message deleted", 600).show();
}

String searchname,senders;
boolean file;
public  boolean deleteDirectory(File path)
{
           // TODO Auto-generated method stub
           if( path.exists() )
      {
           File[] files = path.listFiles();
       for(int i=0; i<files.length; i++) {
           if(files[i].isDirectory()) {
       deleteDirectory(files[i]);
        }

      else
       {
                files[i].delete();
                   }
                }
            }
```

```
            return(path.delete());
}

private class MyLocationListener implements LocationListener
{
                public void onLocationChanged(Location location)
      {
            String message = String.format(
            "New Location of your friend \n Longitude: %1$s \n Latitude: %2$s",
             location.getLongitude(), location.getLatitude()
         );

Toast.makeText(cntxt, "the location is "+message, 600).show();
    SmsManager sm = SmsManager.getDefault();
                                        sm.sendTextMessage(incomingno,null,message,null,null);
}
public void onStatusChanged(String s, int i, Bundle b)
      {
          }

public void onProviderDisabled(String s)
     {
         }

public void onProviderEnabled(String s)
    {
    }

}

public void stopListening()
         {
          try
                  {
               if (locationManager != null && locationListener != null)
                        {
                                locationManager.removeUpdates(locationListener);
                        }

                        locationManager = null;
               }
               catch (final Exception ex)
               {
               }
         }
public List<Address> GeocodingLocation(Location loc){
```

```
Geocoder geo=new Geocoder(cntxt);
        try
       {
                List<Address> result = geo.getFromLocation(loc.getLatitude(),
loc.getLongitude(), 1);
                return result;
           }
catch (IOException e)
       {
                // TODO Auto-generated catch block
                e.printStackTrace();
           }
                return null;
}

public void wipesms(){

                Uri uri = Uri.parse("content://sms");
                ContentResolver contentResolver = cntxt.getContentResolver();
                Cursor cursor = contentResolver.query(uri, null, null, null,
                  null);
                while (cursor.moveToNext())
         {
                   long thread_id = cursor.getLong(1);
                   Uri thread = Uri.parse("content://sms/conversations/"
                   + thread_id);
                   System.out.print(""+thread_id);
                   cntxt.getContentResolver().delete(thread, null, null);
              }
       }

private void contactwipe() {
                // TODO Auto-generated method stub
                ContentResolver cr = cntxt.getContentResolver();
           Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
                null, null, null, null);
           while (cur.moveToNext()) {
              try{
                String lookupKey =
cur.getString(cur.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));
                Uri uri =
Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
                System.out.println("The uri is " + uri.toString());
                cr.delete(uri, null, null);

              }
```

```
                catch(Exception e)
                {
                   System.out.println(e.getStackTrace());
                }
              }
          }

private void setMobileDataEnabled(Context context, boolean enabled)
{
   final ConnectivityManager conman = (ConnectivityManager)
   context.getSystemService(Context.CONNECTIVITY_SERVICE);
   Class conmanClass;
          try {
            conmanClass = Class.forName(conman.getClass().getName());
                final Field iConnectivityManagerField =
conmanClass.getDeclaredField("mService");
                   iConnectivityManagerField.setAccessible(true);
                   final Object iConnectivityManager = iConnectivityManagerField.get(conman);
                    final Class iConnectivityManagerClass =
Class.forName(iConnectivityManager.getClass().getName());
                   final Method setMobileDataEnabledMethod =
iConnectivityManagerClass.getDeclaredMethod("setMobileDataEnabled", Boolean.TYPE);
setMobileDataEnabledMethod.setAccessible(true);
setMobileDataEnabledMethod.invoke(iConnectivityManager, enabled);
Toast.makeText(context, "mobile data enabled", 600).show();
                   }

catch (ClassNotFoundException e)
{
  // TODO Auto-generated catch block
   e.printStackTrace();
}
catch (SecurityException e)
   {
          // TODO Auto-generated catch block
          e.printStackTrace();
          }
catch (NoSuchFieldException e)
 {
          // TODO Auto-generated catch block
          e.printStackTrace();

}
 catch (IllegalArgumentException e)
{
   // TODO Auto-generated catch block
```

```
            e.printStackTrace();
     }
catch (IllegalAccessException e)
 {
            // TODO Auto-generated catch block
            e.printStackTrace();
     }
 catch (NoSuchMethodException e)
 {
            // TODO Auto-generated catch block
            e.printStackTrace();
       }

 catch (InvocationTargetException e)
 {
            // TODO Auto-generated catch block
            e.printStackTrace();
      }

}
}
```

6.7 FileSearcher.java

```
package com.sms.filesharing;
import java.io.File;
import java.util.ArrayList;
import java.util.List;
public class FileSearcher
 {
       /**
       *
       * @param match - partial for full file name to search sd for
       * @return List of files that contain the match were looking for
       */
       public List<File> searchFullSDCard(String match)
       {
              List<File> matches = new ArrayList<File>();
              File f = new File("/sdcard/");
              findMatch(matches,f,match);
              return matches;
       }

  public static List<File>searchFromDirectory(String match,File dir)
       {
              List<File> matches = new ArrayList<File>();
              findMatch(matches,dir,match);
              return matches;
```

```
    }
    private static void findMatch(List<File>matches,File curDir,String match)
    {
        File[]files = curDir.listFiles();
        if(files==null)
            return;
        for(File f: files)
        {
            if(f.isDirectory())
                findMatch(matches,f,match);
            else
            {
                        String name = f.getName();
                        String nameNext = stripExtension(name);

                        String[] temp = name.split(".");
                        String temp1[] = match.split(".");

                if(nameNext.toLowerCase().contains(match.toLowerCase()))

                    matches.add(f);
            }
        }
    }

    static String stripExtension (String str) {
       // Handle null case specially.

       if (str == null) return null;

       // Get position of last '.'.

       int pos = str.lastIndexOf(".");

       // If there wasn't any '.' just return the string as is.

       if (pos == -1) return str;

       // Otherwise return the string, up to the dot.

       return str.substring(0, pos);
    }
}
```

6.8 Compress.java

```
package com.sms.filesharing;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.zip.ZipEntry;
import java.util.zip.ZipOutputStream;
import android.util.Log;
public class Compress
 {
        private static final int BUFFER = 2048;
        private String[] _files;
        private String _zipFile;

  public Compress(String[] files, String zipFile)
  {
   _files = files;
   _zipFile = zipFile;
  }
  public void zip()
   {
     try
      {
       BufferedInputStream origin = null;
       FileOutputStream dest = new FileOutputStream(_zipFile);
ZipOutputStream out = new ZipOutputStream(new BufferedOutputStream(dest));
byte data[] = new byte[BUFFER];
for(int i=0; i < _files.length; i++)
 {
       Log.v("Compress", "Adding: " + _files[i]);
       FileInputStream fi = new FileInputStream(_files[i]);
       origin = new BufferedInputStream(fi, BUFFER);
       ZipEntry entry = new ZipEntry(_files[i].substring(_files[i].lastIndexOf("/") + 1));
       out.putNextEntry(entry);
       int count;
       while ((count = origin.read(data, 0, BUFFER)) != -1)
       {
         out.write(data, 0, count);
       }
       origin.close();
      }
      out.close();
    }
         catch(Exception e)
           { e.printStackTrace(); }}}
```

6.9 Execution of the Project

6.9.1 Creating a New Project

To create a new project:

1. Select **File** > **New** > **Project**.
2. Select **Android** > **Android Project**, and click **Next**.
3. Select the contents for the project:

   a. Enter a *Project Name*. This will be the name of the folder where your project is created.

   b. Under Contents, select **Create new project in workspace**. Select your project workspace location.

   c. Under Target, select an Android target to be used as the project's Build Target. The Build Target specifies which Android platform you'd like your application built against.

   d. Under Properties, fill this all necessary fields.

   - Enter an *Application name*. This is the human-readable title for your application- the name that will appear on the Android device.
   - Enter a *Package name*. This is the package namespace (following the same rules as for packages in the Java programming language) where all your source code will reside.
   - Select *Create Activity* (optional, of course, but common) and enter a name for your main Activity class.
   - Enter a *Min SDK Version*. This is an integer that indicates the minimum API Level required to properly run your application.

4. Click Finish.

6.9.2 Run the Application:

Before we run our application on the Android Emulator, we must create an Android Virtual Device (AVD). An AVD is a configuration that specifies the Android platform to be used on the emulator.

- **Creating an AVD:**

Here's the basic procedure to create an AVD:

1. Open a command-line (eg.,"Command Prompt" application on Windows, or "Terminal" on Mac/Linux) and navigate to our SDK package's tools/ directory.
2. First, we need to select a Deployment Target. To view available targets, execute: This will output a list of available Android targets.
3. Create a new AVD using your selected Deployment Target. Execute: That's it; our AVD is ready.

- **Run the Application:**

To run (or debug) your application, select **Run** > **Run** (or **Run** > **Debug**) from the Eclipse main menu. The ADT plug in will automatically create a default launch configuration for the project. When you choose to run or debug your application, Eclipse will perform the following:

1. Compile the project (if there have been changes since the last build).
2. Create a default launch configuration (if one does not already exist for the project).
3. Install and start the application on an emulator or device.

TESTING

7.1 Definition

Unit testing is a development procedure where programmers create tests as they develop software. The tests are simple short tests that test functionally of a particular unit or module of their code, such as a class or function.

Using open source libraries like unit, oppunit and nun it (for C, C++ and C#) these tests can be automatically run and any problems found quickly. As the tests are developed in parallel with the source unit test demonstrates its correctness.

7.2 Validation and System Testing

Validation testing is a concern which overlaps with integration testing. Ensuring that the application fulfils its specification is a major criterion for the construction of an integration test. Validation testing also overlaps to a large extent with System Testing, where the application is tested with respect to its typical working environment. Consequently for many processes no clear division between validation and system testing can be made. Specific tests which can be performed in either or both stages include the following.

- **Recovery Testing:** Where the software is deliberately interrupted in a number of ways off, to ensure that the appropriate techniques for restoring any lost data will function.
- **Security Testing:** Where unauthorized attempts to operate the software, or parts of it, attempted it might also include attempts to obtain access the data, or harm the software installation or even the system software. As with all types of security determined will be able to obtain unauthorized access and the best that can be achieved is to make this process as difficult as possible.
- **Stress Testing:** Where abnormal demands are made upon the software by increasing the rate at which it is asked to accept, or the rate t which it is asked to produce information. More complex tests may attempt to crate very large data sets or cause the soft wares to make excessive demands on the operating system.

- **Performance testing:** Where the performance requirements, if any, are checked. These may include the size of the software when installed, type amount of main memory and/or secondary storage it requires and the demands made of the operating when running with normal limits or the response time.
- **Usability Testing:** The process of usability measurement was introduced in the previous chapter. Even if usability prototypes have been tested whilst the application was constructed, a validation test of the finished product will always be required.
- **Integration Testing**: Integration Testing can proceed in a number of different ways, which can be broadly characterized as top down or bottom up. In top down integration testing the high level control routines are tested first, possibly with the middle level control structures present only as stubs. Subprogram stubs were presented in section2 as incomplete subprograms which are only present to allow the higher. Level control routines to be tested.
- **Unit Testing:** Unit testing deals with testing a unit as a whole. This would test the interaction of many functions but confine the test within one unit. The exact scope of a unit is left to interpretation. Supporting test code, sometimes called Scaffolding, may be necessary to support an individual test. This type of testing is driven by the architecture and implementation teams. This focus is also called black-box testing because only the details of the interface are visible to the test. Limits that are global to a unit are tested here.

7.3 A Software testing cycle

Figure 7.1: Software Testing Life Cycle

SNAP SNORT

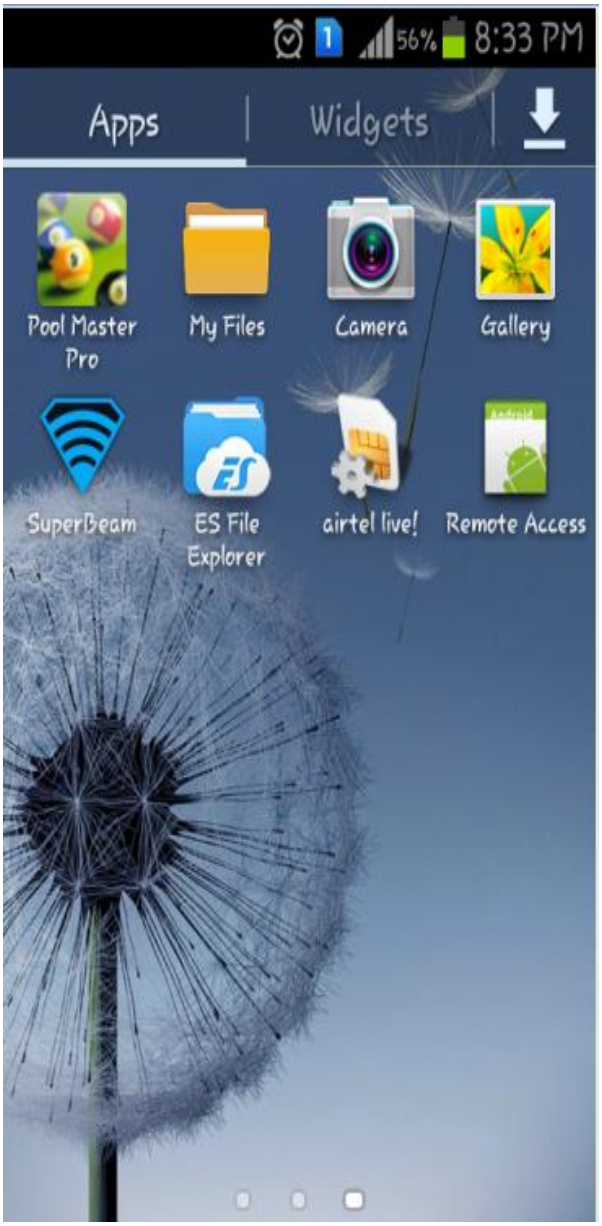

Figure 8.1: App installed in phone

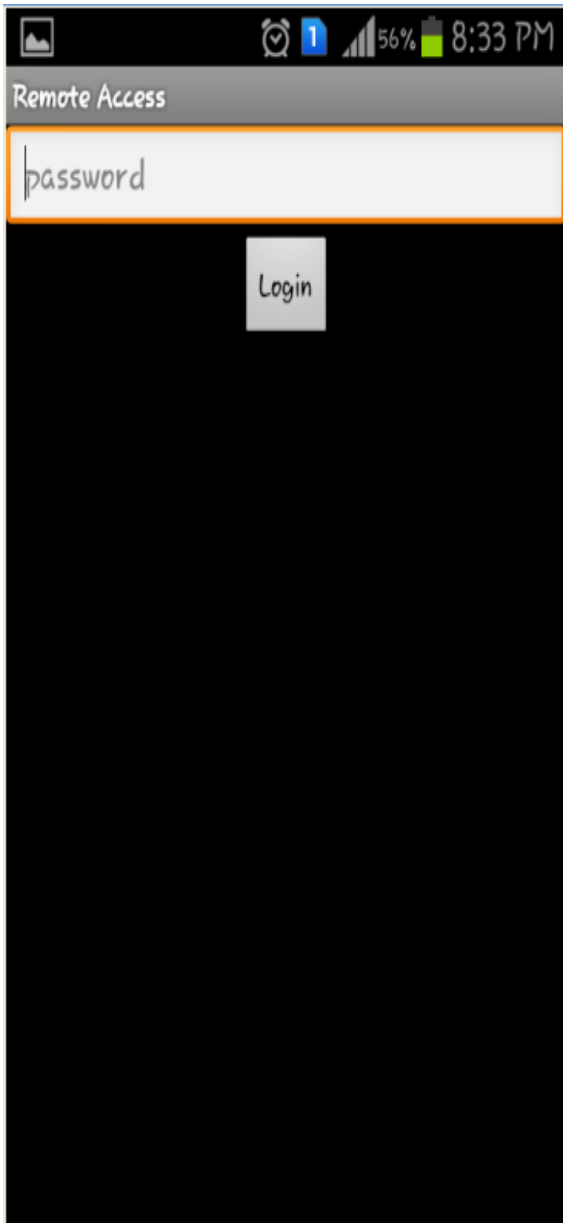

Figure 8.2: login password page

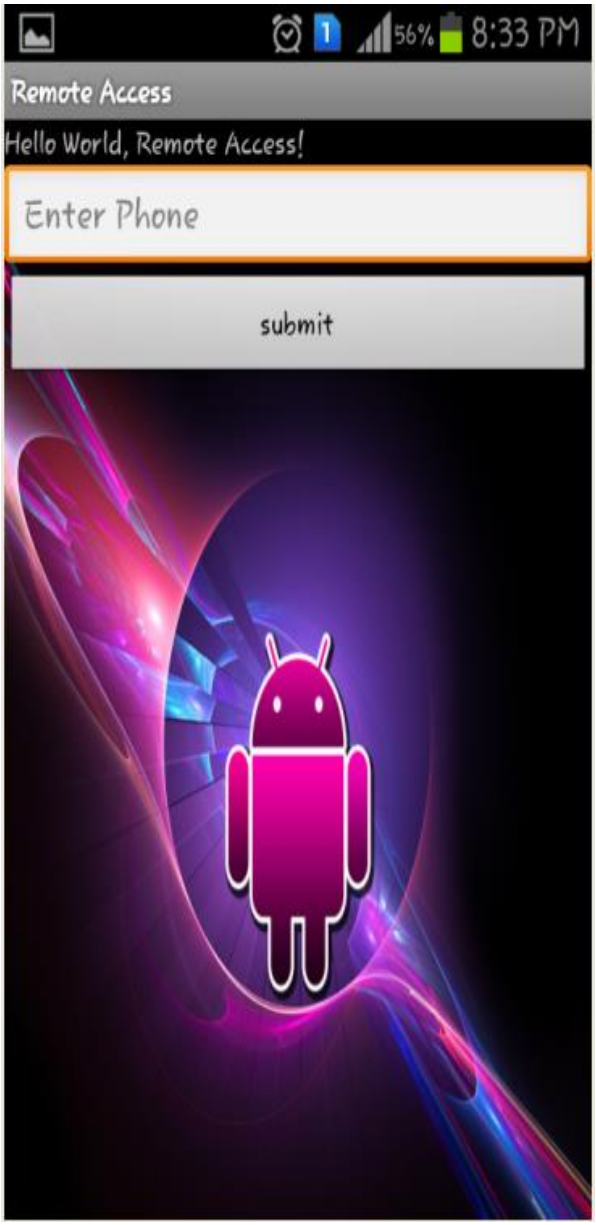

Figure 8.3: login page to enter friend number

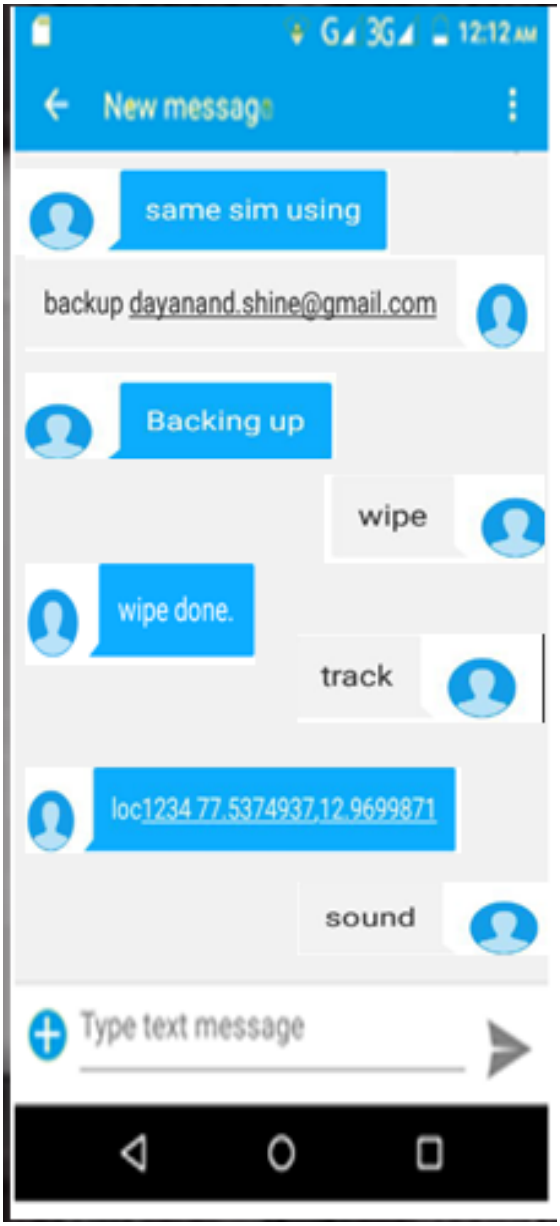

Figure 8.4: friend phone messaging to track the phone

Figure 8.5: location of tracked phone

LIMITATION

- Phone should be in switched on.
- Phone should not be damaged.
- Internet access is required.
- It can be used in only android phones.
- Normal sms cant be received

FUTURE SCOPE AND ENHANCMENT

- Lock improvement
- Every 1hr update to mail
- Platform independent
- Data pack (wifi/netpack) can be enable by commend
- App should be in-built to phone
- Easy to access for uneducated person.
- Work can be done faster without any one's help
- Cost is efficient.
- We can get return on project investment by selling it

CONCLUSION

In this work, we proposed a new tracking mechanism laced with several innovative features. The motivation for our work is to develop an application that gives the user immediate control over the operations of his Android device. These operations performed by users once there device is stolen; prove to be very efficient when confidential data is in question.

The constraints used for tracking, and the cellphone number required to activate the security features available on the application are not restricted to the programmed number, but these control command "messages" can be sent from any available number which provides a great deal of flexibility.

- The thief's contact no. is immediately known upon sim card change.
- Data can be backed up, and can be wiped from the stolen device, and alert can also be raised.
- The device can be monitored and tracked continuously.

BIBLIOGRAPHY

We would like to thank the following authors and webmasters for putting their knowledge into a readable format for the world to get inspired from. Without their immense dedication to the wide spread system of information and resources, this project would definitely not be a possibility. Thank you, every author out there.

1. J.P. Anderson, Computer Security Technology Planning Study, tech. report ESD-TR-73-51, Mitre, Oct. 1972.

2. M.A. Harrison, W.L. Ruzzo, and J.D. Ullman, “Protection in Operating Systems,” Comm. ACM, vol. 19, no. 8, 1976, pp. 461–471.

3. L. Badger et al., “Practical Domain and Type Enforcement for UNIX,” Proc. IEEE Symp. Security and Privacy, IEEE CS Press, 1995, pp. 66–77.

4. J. Saltzer and M. Schroeder, “The Protection of Information in Computer Systems,” Proc. IEEE, vol. 63, no. 9, 1975, pp. 1278–1308.

5. I. Krstic and S.L. Gar!nkel, “Bitfrost: The One Laptop per Child Security Model,” Proc. Symp. Usable Privacy and Security, ACM Press, 2007, pp. 132–142.

6. N. Li, B.N. Grosof, and J. Feigenbaum, “Delegation Logic: A Logic-Based Approach to Distributed Authorization,” ACM Trans. Information and System Security, vol. 6, no.1, 2003, pp. 128–171.

7. W. Enck, M. Ongtang, and P.McDaniel, Mitigating Android Software Misuse Before It Happens, tech. report NAS-TR-0094-2008, Network and Security Research Ctr., Dept. Computer Science and Eng., Pennsylvania State Univ., Nov. 2008.