1. PROJECT SUMMARY

Title : MOBILE THEFT WITH DATA BACKUP AND TRACKING FACILITY

Definition : To detect the lost mobile

Team Size : 4

Team members : Chandu M

Surya Prakash M

Venu Gopal V

Vijayalakshmi P

Software Requirements: Android SDK 1.5 or more

Front-End : Java

Back-End : XML

Project Guide : Roopa B

2. INTRODUCTION

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Android is a software platform and operating system for mobile devices based on the Linux operating system and developed by Google and the Open Handset Alliance. It allows developers to write managed code in a Java-like language that utilizes Google-developed Java libraries, but does not support programs developed in native code

The unveiling of the Android platform on 5 November 2007 was announced with the founding of the Open Handset Alliance, a consortium of 34 hardware, software and telecom companies devoted to advancing open standards for mobile devices. When released in 2008, most of the Android platform will be made available under the Apache free-software and open-source license

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 this application in your Android phone no more worrying of misusing and accessing of data once it lost.

If the mobile has many confidential data related to business or the company, or personal photos and the mobile is lost, then the data is also lost and there are chances that the thief can use these data which may incur loss in our business. Personal photos can be misused and may lead to harassment

Our approach is to use the number one fasting growing device which billions of people already own, a programmable cellular phone. Using existing cell phone technology not only reduces the cost in searching your cell phone, it also exploits a greater range of communication capabilities and integrated hardware and software features. Touch screen response, GPS, voice recognition, common to smart phones, provide a reliable interface for the user. By using interfaces that are similar to applications the user frequently uses.

This project intended to access your phone remotely by using Short Message Services (SMS), we have created some set of commands based on the data security, user need to send any one command as a message content from the specified no, the application reads that and works according to the command behavior which was predefined.

Eg: Assume mobile has been lost, now you send a message to your no suddenly as "WIPE", applications read this command and delete all the data of the mobile.

Further if the thief is changing locations after every 50 meters messages with respect to the current location gps points of the thief is updated to the friend's mobile number. They can track the thief using the location got from the robbed mobile.

This is the most effective way of tracking down the thief. Along with this at a common point you will come in contact with the thief. But there are too many members with the same phone.

You cannot ask everyone to show their mobile phone for verification. People would not entertain this. But how do we identify the thief in the crowd? Here we have written the application in such a way what ever ringtone the thief has installed. Once the call is given from the friend's phone.

The ringtone changes to "THIEF THIEF THIEF THIEF THIEF THIEF THIEF". When this ringtone is activated you can easily identify who the person is who has thieved your phone.

2.1 Scope

- Mobile theft with data backup and tracking facility is to detect the lost smartphone.
- This system is to backup the data from remote location to backup the images, video from remote mobile.
- It provides the facility to wipe data and to find the location of lost Smartphone.

- It provides the facility to plot GPS coordinates provided by smart phone. Voice caller tune in case of Smartphone in silent mode.

2.2 About android

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device, the HTC Dream, being launched in September 2008.

It is free and open-source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License. However most Android devices ship with additional proprietary software pre-installed,[12] most notably Google Mobile Services (GMS)[13] which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform.

2.2.1 What is an android?

Android is a Linux-based mobile phone operating system developed by Google. Android is unique because Google is actively developing the platform but giving it away for free to hardware manufacturers and phone carriers who want to use Android on their devices. Android is an open source operating system for mobile devices that includes middleware and key application, and uses a modified version of Linux kernel.

2.2.2 Android architecture

The following diagram 1.6.1 shows the major components of android operating system.each section is described in more detail below.

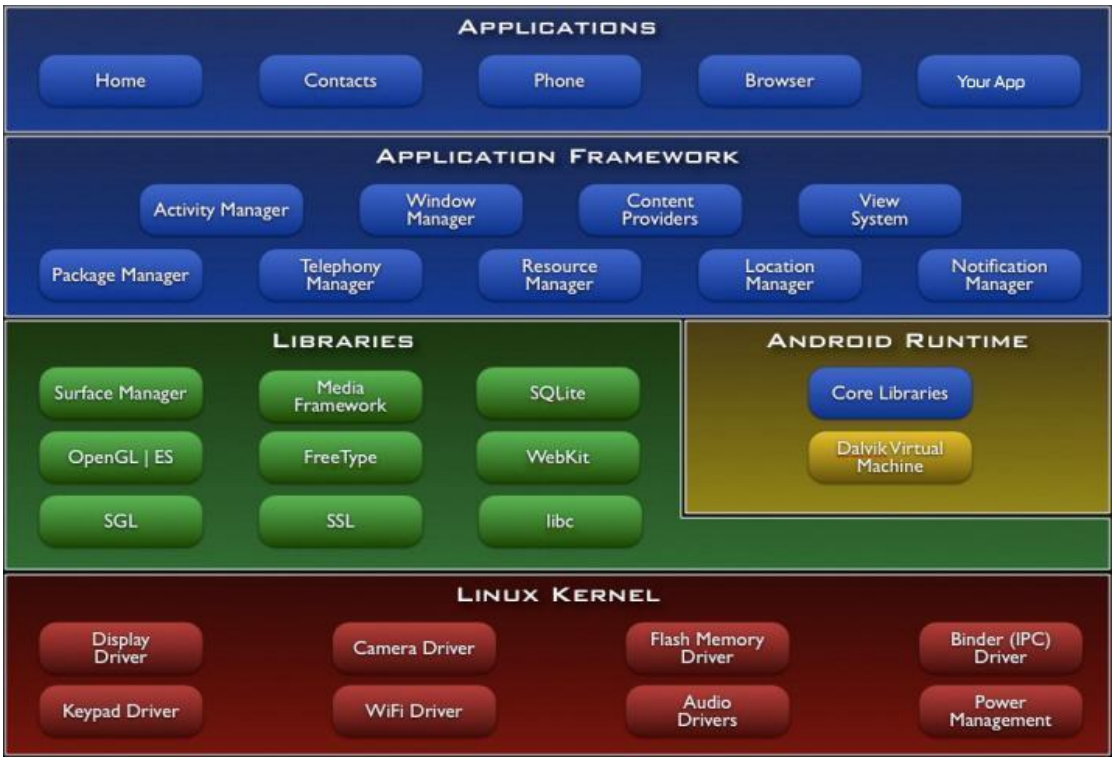

Fig 2.2.1 Android Architecture

2.2.3 Applications

Applications are the top layer in the Android architecture and this is where our applications are gonna fit. Several standard applications comes pre-installed with every device, such as:

- SMS client app
- Dialer
- Web browser
- Contact manager

As a developer we are able to write an app which replace any existing system app. That is, you are not limited in accessing any particular feature. You are practically limitless and can whatever you want to do with the android (as long as the users of your app permits it). Thus Android is opening endless opportunities to the developer.

2.2.4Application Framework

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take

advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more.

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

* A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
* Content Providers that enable applications to access data from other applications
* A Resource Manager, providing access to non-code resources such as localized strings.
* An Activity Manager that manages the lifecycle of applications and provides a common navigation backstack

2.2.5 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included “dx” tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.Linux KernelAndroid relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.

2.2.6 Libraries

The Android runtime core libraries outlined in the preceding section are Java-based and provide the primary APIs for developers writing Android applications. It is

important to note, however, that the core libraries do not actually perform much of the actual work and are, in fact, essentially Java "wrappers" around a set of C/C++ based libraries. When making calls, for example, to the android.opengl library to draw 3D graphics on the device display, the library actually ultimately makes calls to the OpenGL ES C++ library which, in turn, works with the underlying Linux kernel to perform the drawing tasks.

In practice, the typical Android application developer will access these libraries solely through the Java based Android core library APIs. In the event that direct access to Android Native Development Kit (NDK), the purpose of which is to call the native methods of non-Java programming languages (such as C and C++) from within Java code using the Java Native Interface (JNI).

3. LITERATURE SURVEY

Chung.C, Wang.C, et al., [1] have proposed a smart phone anti-theft solution based on locking SIM card of mobile phone is proposed. The solution proposed in this work mainly consists of software components, according to which hardware needs to be selected. It also gives the overall structure of mobile phone anti-theft system, the main software functional blocks and the implementation flow chart of each module. The functions of locking and unlocking of mobile and SIM card, anti-theft and short messaging service control are realized by adding locking SIM card setting module, locking SIM card control module and anti-theft processing module. By the use of software algorithm, users may find stolen mobile phone and protect critical information.

Piyare.R and Tazil [2] have proposed LUXUS SMS controller. In the last two decades, mobile phones have evolved and have become smarter / intelligent devices commonly known as smart phones. In 2005, Google entered into the mobile market with the help of an open source operating system for smart phones called Android Application development for Android OS is greatly increasing because it is open-source and development toolkit is free. LUXUS SMS Controller (LSC) which is an Android Based Application developed on Android Honeycomb 3.2. LSC is all in one SMS manager and includes some previous features with advancements as well as some new and exciting Features like Group chat etc. The proposed application has been tested on different scenarios and experimental results shows that the proposed application is effective both in terms of efficiency and time.

Sadagopan V.K, Rajendra.U, et al., [3] have proposed a novel anti theft control system for automobiles that tries to prevent the theft of a vehicle. This system makes use of an embedded chip that has an inductive proximity sensor, which the key during insertion and sends a text message to the owner's mobile stating that the car is being accessed. This is followed by the system present in the car asking the user to enter a unique password. The password consists of few characters and the car key number. If the user fails to enter the correct password in three trials, a text message is sent to the police with the vehicle number and the location tracked using a GPS module. The message is also sent to the owner about the unauthorized usage. Further the fuel

injector of the car is deactivated so that the user cannot start the car by any means. At the same time a secret lock system gets activated and the unauthorized user gets trapped inside the car and only the owner who is equipped with the key to the secret lock system can deactivate the mechanism. This technique helps in taking fast steps towards an attempt to steal the. The design is robust and simple.

Al-Ali.A.Rand AL-Rousan.M [4] have proposed the design & development of a theft control system for an automobile, which is being used to prevent control the theft of a vehicle. The developed system makes use of an embedded system based on GSM technology. The designed & developed system is installed in the vehicle. An interfacing mobile is also connected to the microcontroller, which is in turn, connected to the engine. Once, the vehicle is being stolen, the information is being used by the vehicle owner for further processing. The information is passed onto the central processing insurance system, where by sitting at a remote place, a particular number is dialed by them to the interfacing mobile that is with the hardware kit which is installed in the vehicle. By reading the signals received by the mobile, one can control the ignition of the engine; say to lock it or to stop the engine immediately. Again it will come to the normal condition only after entering a secured password. The owner of the vehicle & the central processing system will know this secured password. The main concept in this design is introducing the mobile communications into the embedded system. The designed unit is very simple & low cost. The entire designed unit is on a single chip. When the vehicle is stolen, owner of vehicle may inform to the central processing system, and then they will stop the vehicle by just giving a ring to that secret number and with the help of SIM tracking knows the location of vehicle and informs to the local police or stops it from further movement.

Nagraj.B.G, Rayappa.R, et al.,[5] have proposed the capability to track the mobility pattern of children and elders can enable many useful applications. Assisted GPS (AGPS) and Wi-Fi can be used to precisely calculate the location of a person, but AGPS consumes a lot of power and Wi-Fi coverage is often spotty. On the other hand, Cell ID offers ubiquitous coverage in most inhabited environments, low power consumption, but has the disadvantage of lower accuracy. This paper introduces an energy-efficient cellular and AGPS/Wi-Fi tracking system based on the construction of a mobility map from a person's historical location data. A location map consists of

a set of crucial locations (CL), personal common locations (PCL), and routes. CL is a point that terminates several routes and PCL is a place where the person spends a lot of time in. The tracking process is divided into four parts: data collection, offline data processing, online tracking, and path reconstruction. With the use of the mobility map, AGPS fixes can be significantly reduced and a more accurate recognition of the person's location can be provided during the tracking phase

Gao.Yongqing, Zhou.Chunlai, et al., [6] have proposed a smart phone anti-theft solution based on locking IMEI number of mobile phones is proposed. The solution proposed in this work mainly consists of software components, according to which hardware needs to be selected. It also gives the overall structure of mobile phone anti-theft system, the main software functional blocks and the implementation flow chart of each module. The functions of locking and unlocking of mobile and IMEI number, anti-theft and short messing services control are realized by adding locking IMEI number setting module, locking IMEI number control module and anti-theft processing module. By the use of software algorithm, users may find stolen mobile phone and protect critical information.

Inwhee Joe, Yoonsang Lee, et al., [7] have proposed the performance upgrade for mobile devices took place due to jumping development technology, so high-performance terminals that anyone can directly search and amend desired information anywhere and anytime, namely, mobile communication terminals called Smartphone were released to the market. Such terminals can store information that an individual saved, for example, cal log, where to make contact and address of acquaintances, transmit/receive message and mail, photographs and videos, etc., but there are worries on surreptitious use of other people due to leaking of personal information through loss or theft of these terminals, so our paper aims to realize a system that remotely controls information inside the terminals in this case. The proposed system consists of four functional units: a website provision unit playing a role of interface so that users can remotely control a terminal, a user confirmation unit that performs joiner confirmation information from a system of a mobile communication company joined after a user confirm oneself as the person oneself, access management unit that sets access by transmitting an access code for initiating a remote control program if a person is confirmed as a real owner of the terminal, and finally a service execution

unit that transmits a remote control service code and an environment coding value for remotely controlling a terminal to the terminal and receives a service completion code from the terminal.

Jie Wu, Snasel.V, et al., [8] have proposed, a novel computer vision navigation system for mobile tracking robot is presented. Three irrelevant technologies, pattern recognition, binocular vision and motion estimation, make up of the basic technologies of our robot. The non-negative matrix factorization (NMF) algorithm is applied to detect the target. The application method of NMF in our robot is demonstrated. Interesting observations on distance measurement and motion capture are discussed in detail. The reasons resulting in error of distance measurement are analyzed. According to the models and formulas of distance measurement error, the error type could be found, which is helpful to decrease the distance error. Based on the diamond search (DS) technology applied in MPEG-4, an improved DS algorithm is developed to meet the special requirement of mobile tracking robot.

Inwhee Joe , Yoonsang Lee, et al.,[20] have proposed the upgrades for mobile devices took place due to jumping development of technology, so high-performance terminals that anyone can directly search and amend desired information anywhere and anytime, namely, mobile communication terminals called Smartphone were released to the market. Such terminals can store information that an individual saved, for example, call log, where to make contact and address of acquaintances, transmit/receive message and mail, photographs and videos, etc., but there are worries on surreptitious use of other people due to leaking of personal information through loss or theft of these terminals, so their paper aims to realize a system that remotely controls information inside the terminals in this case. The proposed system consists of four functional units: a website provision unit playing a role of interface so that users can remotely control a terminal, a user confirmation unit that performs joiner confirmation information from a system of a mobile communication company joined after a user confirm oneself as the person oneself, an access management unit that sets access by transmitting an access code for initiating a remote control program if a person is confirmed as a real owner of the terminal, and finally a service execution unit that transmits a remote control service code and an environment setting value for

remotely controlling a terminal to the terminal and receives a service completion code from the terminal.

De Clunie, G.T, Panama Serrao.T, et al., [22]have proposed the development of MLEA, a platform that assists, through Android cell phones and tablets, the mobility of users of learning virtual environments. MLEA is an application that implements computational techniques such as web services, design patterns, ontologies and mobile computational techniques in order to allow the communication between mobile devices and the content management system - Moodle. It's based on a service oriented, client server architecture that combines the REST protocol and JSON format for data interchange. The client will be provided with features for alerts, file downloads, chats and forums, grade books, quizzes, and calendar, among others.

4. OBJECTIVES

- To detect the lost mobile.
- To get back up of all the contacts and confidential data stored in a separate folder in the mobile.
- To delete all the data stored in the mobile.
- To track the mobile's location using GPS.
- Can access the mobile by sending short messages.
- Set a ringtone which has to be rung when called to the theft mobile

4.1 Overview

Application Framework: It is used to write applications for Android. Unlike other embedded mobile environments, Android applications are all equal, for instance, an applications which come with the phone are no different than those that any developer writes.

Dalvik Virtual Machine: It is extremely low-memory based virtual machine, which was designed especially for Android to run on embedded systems and work well in low power situations. It is also tuned to the CPU attributes. The Dalvik VM creates a special file format (.DEX) that is created through build time post processing. Conversion between Java classes and .DEX format is done by included "dx" tool.

JAVA: 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

Integrated Browser: Google made a right choice on choosing WebKit as open source web browser. They added a two pass layout and frame flattening. Two pass layout loads a page without waiting for blocking elements, such as external CSS or external JavaScript and after a while renders again with all resources downloaded to the device. Frame flattening converts founded frames into single one and loads into the browser. These features increase speed and usability browsing the internet via mobile phone.

Optimized graphics: As Android has 2D graphics library and 3D graphics based on OpenGL ES 1.0, possibly we will see great applications like Google Earth and spectacular games like Second Life, which come on Linux version. At this moment, the shooting legendary 3D game Doom was presented using Android on the mobile phone.

SQLite: Extremely small (~500kb) relational database management system, which is integrated in Android. It is based on function calls and single file, where all definitions, tables and data are stored. This simple design is more than suitable for a platform such as Android.

Handset Layouts: The platform is adaptable to both larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 1.0 specifications, traditional smart phone layouts. An underlying 2D graphics engine is also included. Surface Manager manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

a. **Data Storage SQLite** is used for structured data storage. SQLite is a powerful and lightweight relational database engine available to all applications.

b. **Connectivity** Android supports a wide variety of connectivity technologies including GSM, CDMA, Bluetooth, EDGE, EVDO, 3G and Wi-Fi.

c. Messaging SMS, MMS, and XMPP are available forms of messaging including threaded text messaging.

d. Web Browser The web browser available in Android is based on the open-source WebKit application framework. It includes LibWebCore which is a modern web browser engine which powers both the Android browser and an embeddable web view.

e. Java Virtual Machine Software written in Java can be compiled into Dalvik bytecodes and executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine.

f. Media Support Android will support advanced audio/video/still media formats such as MPEG-4, H.264, MP3, and AAC, AMR, JPEG, PNG, GIF. g.

Additional Hardware Support Android is fully capable of utilizing video/still cameras, touch screens, GPS, compasses, accelerometers, and accelerated 3D graphics.

Development Environment Includes a device emulator, tools for debugging, memory and performance profiling, a plugin for the Eclipse IDE. There are a number of hardware dependent features, for instance, a huge media and connections support, GPS, improved support for Camera and simply GSM telephony. A great work was done for the developers to start work with Android using device emulator, tools for debugging and plugin for Eclipse IDE.

5. TOOLS AND ENVIRONMENT

5.1 Hardware and software 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)

5.2 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 byte code (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

5.3 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".

5.4 Java features

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as Java buzzwords.

5.4.1 Platform Dependent:

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.

5.4.2 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.

5.4.3 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.

5.4.4 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

5.4.5 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.

5.4.6 Portable:

The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also has the standard data size irrespective of operating system or the processor. These features make the java as a portable language.

5.4.7 Dynamic:

While executing the java program the user can get the required files dynamically from a local drive or from a computer thousands of miles away from the user just by connecting with the Internet.

5.4.8 Secure:

Java does not use memory pointers explicitly. All the programs in java are run under an area known as the sand box. Security manager determines the accessibility options of a class like reading and writing a file to the local disk. Java uses the public key encryption system to allow the java applications to transmit over the internet in the secure encrypted form. The bytecode Verifier checks the classes after loading.

5.4.9 Performance:

Java uses native code usage, and lightweight process called threads. In the beginning interpretation of byte code resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

5.4.10 Multithreaded:

As we all know several features of Java like Secure, Robust, Portable, dynamic etc; you will be more delighted to know another feature of Java which is Multithreaded. Java is also a multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer. Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they

run on the system CPUs. This is how Multithreading works in Java which you will soon come to know in details in later chapters.

5.4.11 Interpreted:

We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code. The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program. The versatility of being platform independent makes java to outshine from other languages. The source code to be written and distributed is platform independent. Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

5.5 Specialty of android operating system

- There are already many mobile platforms on the market today, including Symbian, iPhone, Windows Mobile, BlackBerry, Java Mobile Edition, Linux Mobile (LiMo), and more.
- While some of its features have appeared before, Android is the first environment that combines: A truly open, free development platform based on Linux and open source.
- Handset makers like it because they can use and customize the platform without paying a royalty.
- A component-based architecture inspired by Internet mash-ups. Parts of one application can be used in another in ways not originally envisioned by the developer.
- You can even replace built-in components with your own improved versions. This will unleash a new round of creativity in the mobile space

5.6 Eclipse IDE

Eclipse as an integrated development environment (IDE) for java. Eclipse is created by an Open Source community and is used in several different areas.

Eg: as a development environment for Java or Android.

The Eclipse project is governed by the Eclipse Foundation. The Eclipse Foundation is a non-profit, member supported corporation that hosts the Eclipse projects and helps to cultivate both an open source community and an ecosystem of complementary products and services. Eclipse roots go back to 2001. Today it is leading development environment for Java with a market share of approx 65%.

Eclipse can be extended with additional functionalities. Several Open Source projects and companies have extended Eclipse with additional components. It is also possible to use Eclipse as a basis for creating general purpose application.

5.7 Eclipse UI Overview

Eclipse provides perspectives, Views and Editors. Views and Editors groped into Perspectives. All projects are located in a workspace.

Workspace:

The workspace is the physical location you are working in. You can choose the workspace during startup of Eclipse or via the menu (FILE-→Switch Workspace→Others)

All your projects, source files, images and other artifacts will be stored and saved in your workspace.

You can predefine the workspace via the startup parameter data path_ to _workspace.

To see the current workspace directory in the title of Eclipse use-showlocation as an additional parameter.

6. DESIGN MODEL

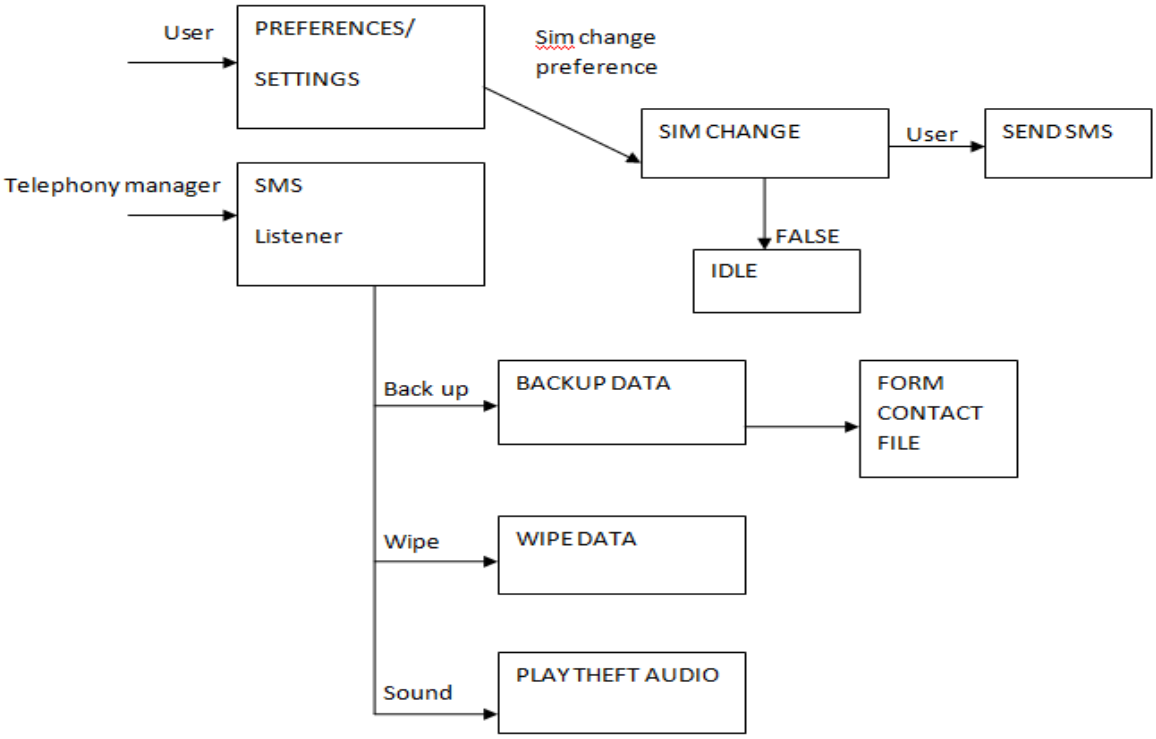

Fig 6.1 : Block diagram of proposed model

- **PREFERENCE AND SETTINGS**: Data which is required to be taken backup if the mobile is lost. Set the number which has to be used in application during execution.
- **SIM CHANGE:** Condition to be checked whether SIM is changed or not using SIM serial number.
- **SEND SMS:** if a new SIM is inserted automatically message is send to the number set.
- **IDLE:** If the SIM is not changed it remains idle and sends the message "same SIM using".
- **TELEPHONY MANAGER:** Provides access to inform about the telephony services like sending message, tracking on the device.
- **SMS LISTENER:** Identifies and reads the SMS sent to the device.
- **BACKUP DATA:** The data which has to be taken backup is saved in the separate folder by the name remote.

- **WIPE DATA:** This block deletes or wipe all the data stored in the remote folder in the mobile. It is done once a message "WIPE" is sent to the mobile.
- **PLAY TEHFT AUDIO**: The set ringtone is played when message "sound" is sent.

6.2 Sequence

1. Finding IMEI number.
2. Current SIM number.
3. Configuring with application.
4. Updating the SMS receiver number.
5. Test the phone number.
6. If SIM is changed , send message to the receiver's number
7. SMS communication.
8. SMS "!@#1234"-for data sent to the alternate number.
9. SMS "sound"-voice alarm starts.
10. SMS "wipe"-data deletes from remote folder.
11. SMS "loc"-connecting with GPS, obtain latitude and longitude values and plot on the map.

7. PROGRAM CODE

Mainactivity.java

```
package com.remote.access.drive;

import android.app.Activity;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import com.google.api.client.json.gson.GsonFactory;

import com.google.api.services.drive.Drive;

import com.google.api.services.drive.DriveScopes;

import com.google.api.services.drive.model.File;

import com.remote.access.R;

import java.util.Collections;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";

    private static final int REQUEST_CODE_SIGN_IN = 1;

    private static final int REQUEST_CODE_OPEN_DOCUMENT = 2;


    private DriveServiceHelper mDriveServiceHelper;

    private String mOpenFileId;


    private EditText mFileTitleEditText;

    private EditText mDocContentEditText;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.drive_mainactivity);

        // Store the EditText boxes to be updated when files are
opened/created/modified.
        mFileTitleEditText = findViewById(R.id.file_title_edittext);
        mDocContentEditText = findViewById(R.id.doc_content_edittext);

        // Set the onClick listeners for the button bar.
        findViewById(R.id.open_btn).setOnClickListener(view ->
openFilePicker());
        findViewById(R.id.create_btn).setOnClickListener(view -> createFile());
        findViewById(R.id.save_btn).setOnClickListener(view -> saveFile());
        findViewById(R.id.query_btn).setOnClickListener(view -> query());

        // Authenticate the user. For most apps, this should be done when the user
performs an
        // action that requires Drive access rather than in onCreate.
        requestSignIn();
    }
```

```
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent resultData)
{
        switch (requestCode) {
            case REQUEST_CODE_SIGN_IN:
                if (resultCode == Activity.RESULT_OK && resultData != null) {
                    Toast.makeText(getApplicationContext(), "handle signin",
6000).show();
                    handleSignInResult(resultData);
                }
                break;

            case REQUEST_CODE_OPEN_DOCUMENT:
                if (resultCode == Activity.RESULT_OK && resultData != null) {
                    Uri uri = resultData.getData();
                    if (uri != null) {
                        openFileFromFilePicker(uri);
                    }
                }
                break;
        }
            super.onActivityResult(requestCode, resultCode, resultData);
    }
```

```
    private void requestSignIn() {
        Log.d(TAG, "Requesting sign-in");

        GoogleSignInOptions signInOptions =
                new
    GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                        .requestEmail()
                        .requestScopes(new Scope(DriveScopes.DRIVE_FILE))
                        .build();
        GoogleSignInClient client = GoogleSignIn.getClient(this, signInOptions);

        // The result of the sign-in Intent is handled in onActivityResult.
        startActivityForResult(client.getSignInIntent(),
    REQUEST_CODE_SIGN_IN);
    }
  private void handleSignInResult(Intent result) {
        GoogleSignIn.getSignedInAccountFromIntent(result)
                .addOnSuccessListener(googleAccount -> {
                    Log.d(TAG, "Signed in as " + googleAccount.getEmail());
                    // Use the authenticated account to sign in to the Drive service.
                    GoogleAccountCredential credential =
                            GoogleAccountCredential.usingOAuth2(
```

```
                        this, Collections.singleton(DriveScopes.DRIVE_FILE));
                credential.setSelectedAccount(googleAccount.getAccount());
                Drive googleDriveService =
                        new Drive.Builder(
                                AndroidHttp.newCompatibleTransport(),
                                new GsonFactory(),
                                credential)
                                .setApplicationName("Drive API Migration")
                                .build();
                // The DriveServiceHelper encapsulates all REST API and SAF
  functionality.
                // Its instantiation is required before handling any onClick actions.
                mDriveServiceHelper = new
  DriveServiceHelper(googleDriveService);
            })
            .addOnFailureListener(exception -> Log.e(TAG, "Unable to sign in.",
  exception));
    }
private void openFilePicker() {
        if (mDriveServiceHelper != null) {
            Log.d(TAG, "Opening file picker.");
            Intent pickerIntent = mDriveServiceHelper.createFilePickerIntent();
            // The result of the SAF Intent is handled in onActivityResult.
```

```
            startActivityForResult(pickerIntent,
REQUEST_CODE_OPEN_DOCUMENT);
        }
    }
    private void openFileFromFilePicker(Uri uri) {
        if (mDriveServiceHelper != null) {
            Log.d(TAG, "Opening " + uri.getPath());

mDriveServiceHelper.openFileUsingStorageAccessFramework(getContentResolv
er(), uri)
                    .addOnSuccessListener(nameAndContent -> {
                        String name = nameAndContent.first;
                        String content = nameAndContent.second;
                        mFileTitleEditText.setText(name);
                        mDocContentEditText.setText(content);
                        // Files opened through SAF cannot be modified, except by
retrieving the
                        // fileId from its metadata and updating it via the REST API. To
modify
                        // files not created by your app, you will need to request the Drive
                        // Full Scope and submit your app to Google for review.
                        setReadOnlyMode();
                    })
                    .addOnFailureListener(exception ->
                            Log.e(TAG, "Unable to open file from picker.", exception));
```

```
        }
    }
    private void createFile() {
        if (mDriveServiceHelper != null) {
            Log.d(TAG, "Creating a file.");
            mDriveServiceHelper.fetchFile()
                    .addOnSuccessListener(fileId ->
Toast.makeText(getApplicationContext(), "successfully created",60000).show())
                    .addOnFailureListener(exception ->
                            Log.e(TAG, "Couldn't create file.", exception))
        }
    }

    private void readFile(String fileId) {
        if (mDriveServiceHelper != null) {
            Log.d(TAG, "Reading file " + fileId);
            mDriveServiceHelper.readFile(fileId)
                    .addOnSuccessListener(nameAndContent -> {
                        String name = nameAndContent.first;
                        String content = nameAndContent.second;
                        mFileTitleEditText.setText(name);
                        mDocContentEditText.setText(content);
```

```
                setReadWriteMode(fileId);
            })
            .addOnFailureListener(exception ->
                Log.e(TAG, "Couldn't read file.", exception));
    }
}
private void saveFile() {
    if (mDriveServiceHelper != null && mOpenFileId != null) {
        Log.d(TAG, "Saving " + mOpenFileId);

        String fileName = mFileTitleEditText.getText().toString();
        String fileContent = mDocContentEditText.getText().toString();

        mDriveServiceHelper.saveFile(mOpenFileId, fileName, fileContent)
            .addOnFailureListener(exception ->
                Log.e(TAG, "Unable to save file via REST.", exception));
    }
}

private void query() {
    if (mDriveServiceHelper != null) {
        Log.d(TAG, "Querying for files.");
```

```
            mDriveServiceHelper.queryFiles()
                    .addOnSuccessListener(fileList -> {
                        StringBuilder builder = new StringBuilder();
                        for (File file : fileList.getFiles()) {
                            builder.append(file.getName()).append("\n");
                        }
                        String fileNames = builder.toString();

                        mFileTitleEditText.setText("File List");
                        mDocContentEditText.setText(fileNames);

                        setReadOnlyMode();
                    })
                    .addOnFailureListener(exception -> Log.e(TAG, "Unable to query
files.", exception));
        }
    }
    private void setReadOnlyMode() {
        mFileTitleEditText.setEnabled(false);
        mDocContentEditText.setEnabled(false);
        mOpenFileId = null;
    }
    private void setReadWriteMode(String fileId) {
```

```
        mFileTitleEditText.setEnabled(true);
        mDocContentEditText.setEnabled(true);
        mOpenFileId = fileId;
    }
```

Smsreciver.java

```
package com.remote.access;
import android.content.BroadcastReceiver;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.content.res.AssetFileDescriptor;
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.ParseException;
import android.net.Uri;
```

```
import android.os.Build;
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;
import androidx.annotation.RequiresApi;
import com.backendless.Backendless;
import com.backendless.async.callback.AsyncCallback;
import com.backendless.exceptions.BackendlessFault;
import com.backendless.files.BackendlessFile;
import com.parse.Parse;
import com.parse.ParseFile;
import com.parse.SaveCallback;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
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;
public class smsReceiver extends BroadcastReceiver {
    static String incomingno, body;
    static LocationManager locationManager;
    SmsManager sm;
    static Context cntxt;
    static MyLocationListener locationListener;
    String pwd, emailid, emailidpwd;
    List<File> sample;
    GMailSender sender;
    String reqtime;
    long time;
    static ArrayList<String> vCard;
    @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 = "";
        vCard = new ArrayList<String>();
```

```
sender = new GMailSender(
        "dayadad4390@gmail.com", "mindsetdayadad");
if (bundle != null) {
    //---retrieve the SMS message received---
    Object[] pdus = (Object[]) bundle.get("pdus");
    msgs = new SmsMessage[pdus.length];
    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();
        abortBroadcast();
        if (body.equalsIgnoreCase("sound")) {
            abortBroadcast();
            Intent intentService = new Intent(cntxt, MyAlarmService.class);
            intentService.putExtra("text", "He is a thief, He is a thief");
            cntxt.startService(intentService);
            MediaPlayer mp = new MediaPlayer();
            try {
           mp.setDataSource(R.raw.voice);
                mp.setDataSource(getFilePath("sound"));
                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("!@#$")) {
        abortBroadcast();
        //      DeleteSms();
        //getVcardString();
        final String[] temp = body.split(" ");

        getVcardString();
       setMobileDataEnabled(cntxt, true);
        Toast.makeText(context, "after data", 600).show();

        File dir = new File("/sdcard/remote/");
        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();
    }
String zipfilename = "/sdcard/" + System.currentTimeMillis() +
    List<File> e;
    Compress c = new Compress(path, zipfilename);
    c.zip();
    File file = new File(zipfilename);
    final List<File> sample = new ArrayList<File>();
    sample.add(file);
    SmsManager sm = SmsManager.getDefault();
    sm.sendTextMessage(incomingno, null, "Backing up", null, null);
   saveFile(file);
    saveFileParse(file);
    Toast.makeText(cntxt, "backup did ", 600).show();
  }
 } else if (body.equalsIgnoreCase("wipe")) {
    abortBroadcast();
    //DeleteSms();
    File file = new File("/sdcard/remote/");
```

```
                File[] files = file.listFiles();
                for (int t = 0; t < files.length; t++) {
                   files[t].delete();
                }
                Toast.makeText(cntxt, "wipe working", 600000).show();
            } else if (body.equalsIgnoreCase("loc")) {
                abortBroadcast();
                final Criteria criteria = new Criteria();
                locationManager = (LocationManager)
cntxt.getSystemService(Context.LOCATION_SERVICE);
                final String bestProvider = locationManager.getBestProvider(criteria,
true);
                Toast.makeText(context, "" + bestProvider, 6000000).show();
                locationListener = new MyLocationListener();
                locationManager.requestLocationUpdates(
                        bestProvider,
                        0,
                        0,
                        locationListener);
            } else if (body.equalsIgnoreCase("stop")) {
                sm.sendTextMessage(incomingno, null, "stoping gps", null, null);
                //DeleteSms();
                if (locationManager != null) {
```

```
                    locationManager.removeUpdates(locationListener);
                    locationManager = null;
                    locationListener = null;
                }
            } else if (body.equalsIgnoreCase("photo")) {
//                String[] temp = body.split(" ");
//                emailPhoto = temp[1];
                Intent in = new Intent(context, OneShotPreviewActivity.class);
                in.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(in);
            }
        }
    }
    static String emailPhoto = "";
    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 = "loc1234 " +
            location.getLatitude() + "," + location.getLongitude();
```

```
            stopListening();
            //Toast.makeText(cntxt, "the location is "+message, 600).show();
            SmsManager sm = SmsManager.getDefault();
            Log.i("SmsReceiver", "" + locationListener + " " + locationManager);
//            locationManager.removeUpdates(locationListener);
            String uri = "http://maps.google.com/maps?q=" + location.getLatitude() +
"," + location.getLongitude() + " (" + "location" + ")";
            sm.sendTextMessage(incomingno, null, uri, 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();
        }
    }
    public static void getVcardString() {
        // TODO Auto-generated method stub
```

```
String storage_path = "/sdcard/SF.vcf";


File file = new File(storage_path);
if (file.exists()) {
   file.delete();


}
vCard = new ArrayList<String>();
Cursor cursor =
cntxt.getContentResolver().query(ContactsContract.CommonDataKinds.Phone.C
ONTENT_URI, null, null, null, null);
if (cursor != null && cursor.getCount() > 0) {


   cursor.moveToFirst();
   for (int i = 0; i < cursor.getCount(); i++) {


      get(cursor);
      Log.d("TAG", "Contact " + (i + 1) + "VcF String is" + vCard.get(i));
      cursor.moveToNext();
   }


} else {
   Log.d("TAG", "No Contacts in Your Phone");
```

```
        }


    }


    public static void get(Cursor cursor) {
        //cursor.moveToFirst();
        String lookupKey =
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_K
EY));
        Uri uri =
Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_VCARD_URI,
lookupKey);
        AssetFileDescriptor fd;
        try {
            fd = cntxt.getContentResolver().openAssetFileDescriptor(uri, "w");



            FileInputStream fis = fd.createInputStream();
            byte[] buf = new byte[(int) fd.getDeclaredLength()];
            fis.read(buf);
            String vcardstring = new String(buf);
            vCard.add(vcardstring);
```

```
            String storage_path = "/sdcard/SF.vcf";


            File file = new File(storage_path);
            //  file.delete();


            if (file.createNewFile()) {
               //Toast.makeText(cntxt, "file created", 60000);
            }
            FileOutputStream mFileOutputStream = new
    FileOutputStream(storage_path, true);
            mFileOutputStream.write(vcardstring.toString().getBytes());


         } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
         }
      }



      public String getFilePath(String filename) {
         String filepath = "/sdcard/iot/";
         File file = new File(filepath);
         if (file.exists()) {
```

```
            File[] list = file.listFiles();
            for (File f : list) {
                if (f.getName().toLowerCase().contains(filename)) {
                    return f.getAbsolutePath();
                }
            }
        }
        return "";
    }
  public void saveFile(File file) {
     Toast.makeText(cntxt, "Uploading file, please wait...",
Toast.LENGTH_SHORT).show();
        Backendless.Files.upload(file, "/photo", new
   AsyncCallback<BackendlessFile>() {
            @Override
            public void handleResponse(BackendlessFile response) {
                Toast.makeText(cntxt, "Photo upload successful",
   Toast.LENGTH_SHORT).show();
                response.getFileURL();

                SmsManager smsManager = SmsManager.getDefault();
                smsManager.sendTextMessage(incomingno, null,
   response.getFileURL(), null, null);
```

```
            }


            @Override
            public void handleFault(BackendlessFault fault) {
                Toast.makeText(cntxt,  "Photo upload failed, retry later. Reason:
"+fault.getMessage(), Toast.LENGTH_SHORT).show();
            }
        });
    }
    @RequiresApi(api = Build.VERSION_CODES.O)
    public void saveFileParse(File file){



        Parse.destroy();
        Parse.initialize(new Parse.Configuration.Builder(cntxt)

.applicationId("kNo917Xe9kNyfbvEXPASEKvYGqg8mTt4TrcjFWTZ")
            // if defined
            .clientKey("cgd4tqmUQEF0a6jv5pJWxch6zAtMpRnQIVyNwr7i")
            .server("https://parseapi.back4app.com")
            .build()
        );
        final ParseFile parseFile = new ParseFile(file);
```

```
        parseFile.saveInBackground(new SaveCallback() {
            @Override
            public void done(com.parse.ParseException e) {
                if (e == null) {
                    Log.i("smsReceiver", "photoFile upload successfull.");


                    SmsManager smsManager = SmsManager.getDefault();
                    smsManager.sendTextMessage(smsReceiver.incomingno, null,
parseFile.getUrl(), null, null);
                } else {
                    Toast.makeText(cntxt, "File upload failed. try again later:
"+e.getMessage(), 6000).show();
                }
            }
        });
    }
}
```

remoteaccess.java

```
package com.remote.access;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.ArrayList;
import java.util.List;
```

```
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.content.res.AssetFileDescriptor;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.telephony.TelephonyManager;
import android.util.Log;
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.TextView;
import android.widget.Toast;
public class Remoteaccess extends Activity {
    /** Called when the activity is first created. */
```

```
        ArrayList<String> vCard ;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        vCard = new ArrayList<String>();
        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
```

```
                    SharedPreferences
dbevent=getSharedPreferences("phone", MODE_PRIVATE);
                    Editor editor=dbevent.edit();
                    editor.putString("phoneval", et1.getText().toString());
                    editor.commit();
                    Toast.makeText(getApplicationContext(), "updated",
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
            {
                //String[] temp = f.getName().split(".");

                if(f.getName().toLowerCase().contains(match.toLowerCase()))
                    matches.add(f);
```

```
                }
            }
        }
            public void getVcardString() {
                // TODO Auto-generated method stub
        String storage_path = "/sdcard/SF.vcf";

        File file = new File(storage_path);
        if(file.exists()){
            file.delete();

        }
                vCard = new ArrayList<String>();
                Cursor cursor =
this.getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CO
NTENT_URI, null, null, null, null);
                if(cursor!=null&&cursor.getCount()>0)
                {

                   cursor.moveToFirst();
                   for(int i =0;i<cursor.getCount();i++)
                   {

                      get(cursor);
```

```
                Log.d("TAG", "Contact "+(i+1)+"VcF String is"+vCard.get(i));
                cursor.moveToNext();
            }
        }
        else
        {
            Log.d("TAG", "No Contacts in Your Phone");
        }
    }
    public  void get(Cursor cursor)
    {
        //cursor.moveToFirst();
        String lookupKey =
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_K
EY));
        Uri uri =
Uri.withAppendedPath(ContactsContract.Contacts.CONTENT_VCARD_URI,
lookupKey);
        AssetFileDescriptor fd;
        try {
            fd = this.getContentResolver().openAssetFileDescriptor(uri, "w");
            FileInputStream fis = fd.createInputStream();
            byte[] buf = new byte[(int) fd.getDeclaredLength()];
            fis.read(buf);
```

```
                String vcardstring= new String(buf);
                vCard.add(vcardstring);
                String storage_path = "/sdcard/SF.vcf";

                File file = new File(storage_path);
               //  file.delete();
                if(file.exists()){
                    file.delete();
                }
                if(file.createNewFile()){
                            Toast.makeText(this, "file created", 60000);
                }
                 FileOutputStream mFileOutputStream = new
FileOutputStream(storage_path, true);
                mFileOutputStream.write(vcardstring.toString().getBytes());
              } catch (Exception e1)
              {
                // TODO Auto-generated catch block
                e1.printStackTrace();
              }
            }
}
```

8. RESULT AND PERFORMANCE

Execution of the application – Screen shots

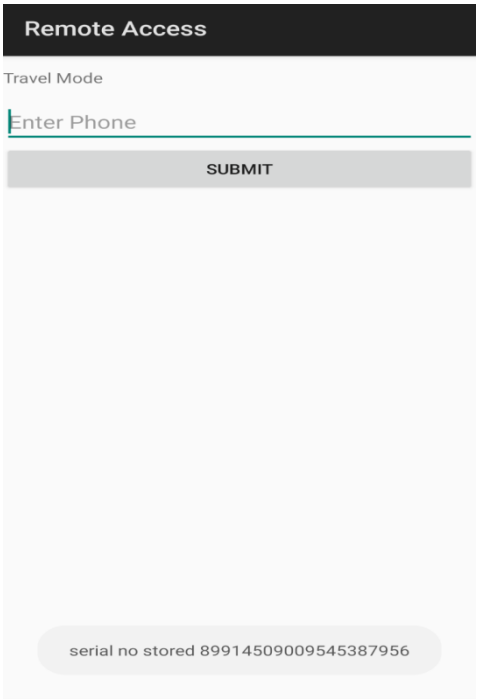

Fig 8.1 Sim check

The sim serial number is being check during the execution of the application

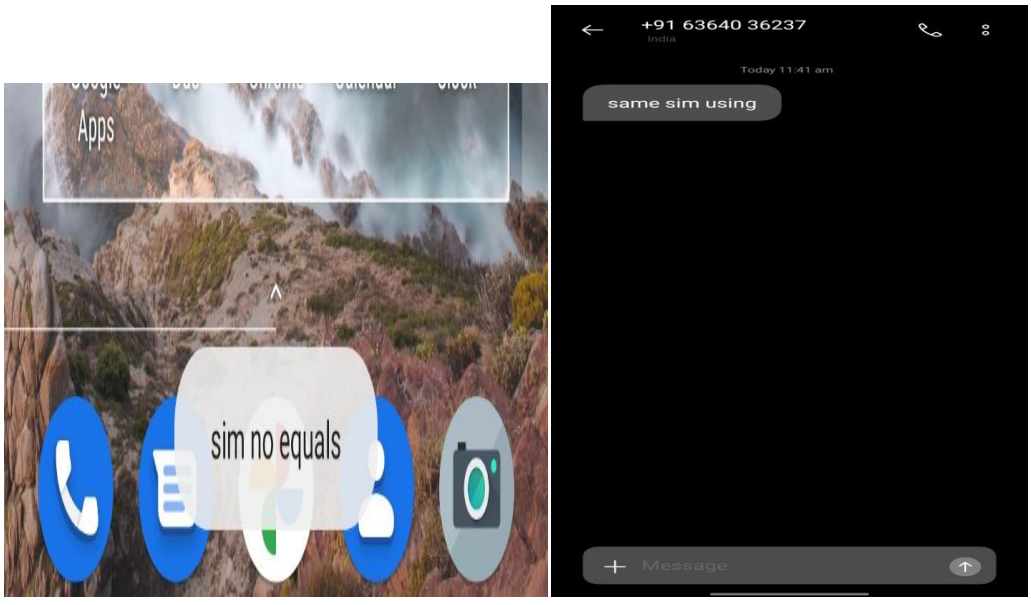

Fig 8.2 screenshot in mobile

When the same sim is inserted onto mobile, the application sends the message as “same sim using” to the number which is saved in the application. The screenshot when the message sent is as shown in the Fig 8.2

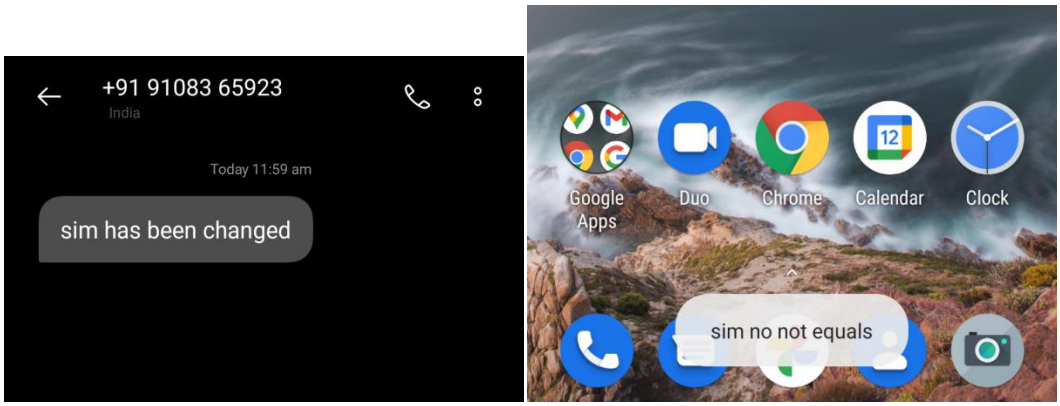

Fig 8.3 Sim Change Message

When the new sim is inserted on to mobile, the application sends the message as "sim has been changed" to the number which is saved in the application. The corresponding screen shot is as shown in the Fig 8.3.

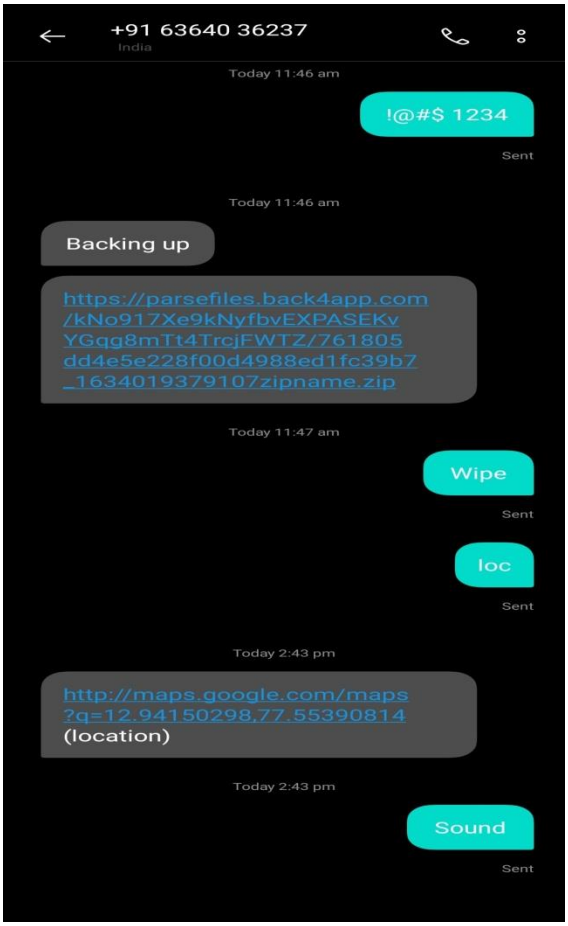

Fig 8.4 Features

The above figure Fig 8.4 is the screen shot taken in the mobile. In this screen shot the messages sent to the theft mobile is shown.

9. TEST CASES

A **test case** is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement.[1] Test cases underlie testing that is methodical rather than haphazard. A battery of test cases can be built to produce the desired coverage of the software being tested. Formally defined test cases allow the same tests to be run repeatedly against successive versions of the software, allowing for effective and consistent regression testing.

In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test.[3] If a requirement has sub-requirements, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and the test is frequently done using a traceability matrix. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

A formal written test case is characterized by a known input and by an expected output, which is worked out before the test is executed.[4] The known input should test a precondition and the expected output should test a post condition.

SL No	Test case name	Test procedure	Expected result	Passed/ failed
01	Message reader	LBSGeocoding activity Reads the message that is got and accordingly jumps to the process or the class to be executed. In this, code for the recording of the audio to be rung is written.	Message read and action to be performed accordingly	Passed
02	Delete messages	Wipesms() Message: WIPE The messages that are saved in the mobile should be deleted in a sequence and all the threads of the message.	Messages deleted	Passed
03	Check for the sim	Simserialnumber Checks for the sim number and the current sim number. If there is a new sim inserted a message is sent to the number stored in the application.	Sim checked	Passed
04	Tracking of the mobile	ShowCurrentLocation() Message: LOCATE This class provides the latitude and longitude values of the places where the mobile is present	Mobile traced	Passed
05	Backup of data	MailSenderActivity() Message: BACKUP The data which has to be taken backup is saved in a separate folder named BU. This is folder is got backup to the mail-id which is stored in the application,	Backup of data is obtained in the mail.	Passed
06	Update location	MyLocationListener() The location of the mobile is updated for every 3000 msecs	Location updated.	Passed
07	Play theft audio	LbsGeocoding activity() Message: SOUND In this the recorded voice is played when the above message is sent, so that it is very easy to find the mobile. It has been set to ring for 5times.	Audio played	Passed

10.LIMITATIONS AND ADVANTAGES

10.1 limitations of present working system

- Android mobiles are costly.
- The application renders of no use in remote areas where there is no network.
- The social contact may not respond.

10.2 Advantages of proposed system

- Reliability and reduced number of false positives mean greater adoption by mobile theft.
- Our system provides a viable solution to increase fall detection among people.
- Using existing, mass marketed technologies will limit cost making it available to the majority of the public
- As the platform used is the mobile phone there is no need of training required for the users as the interface used is known to them.
- Fast and easy development - The SDK contains what you need to build and run Android applications, including a true device emulator and advanced debugging tools

11.Conclusion

This application is only for android operating system and it is a completely failure in places where there is no network coverage. In future, this application would be extended to work even if there is no network service. This application should be implemented on other operating system like iphone operating system, bada, java etc. and not only for and android.

This application is used when the mobile is lost. We can avoid misuse of the data in the mobile and can take backup of the data is required to us. Tracking is also made easy by getting the latitude and longitude values indicating the location of the mobile. An audio is been set for easy identification of the lost mobile.

12.BIBILOGRAPHY

1. www.google.com
2. www.wikipedia.com
3. https://www.ijarcce.com/upload/2016/march-16/IJARCCE%20171.pdf
4. https://www.dummies.com/web-design-development/mobile-apps/the-compile-sdk-minimum-sdk-and-target-sdk-versions/
5. Other Internet Sources