LEARNING FROM PERSONAL DATA COLLECTED ON MOBILE DEVICES

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Ms .RUCHI VERMA

By

ANKITA MITTAL(111280)

to

Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled "Learning from personal data collected on mobile devices.", submitted by Ankita Mittal in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:	**RUCHI VERMA**
15-05-2015	**Professor**

ACKNOWLEDGEMENT

At the very outset, I am highly indebted to Jaypee University of Information Technology for giving me an opportunity to carry out my project on "Learning from personal data collected on mobile devices" at their esteemed organization. I would specially thank to my Project Guide Ms. RUCHI VERMA for giving time and guidance throughout my project development without whom it would have been impossible to attain success.

15.05.2015 ANKITA MITTAL (111280)

Table of Content

List of Figures

Abstract

The modern smart phone is the first device in widespread use that seldom loses sight of its owner. Smart phones collect process and store a considerable amount of information about their owners, such as calendar events, the precise locations visited, the routes taken, the walking and driving speeds, the financial transactions, the images captured with the phone's camera, the phone call logs or the communications via text messages, instant messages or email. The goal of this project is to explore the feasibility of a mobile application, for iOS or Android devices that mines and correlates all these data sources to provide additional insights and assistance to the user. For example, the application could guess the next action that the user is about to perform and assist the user as much as possible with this task. The expected benefits of the proposed application are:

1. **Quick Services:** It gives quick response as you don't need internet connection for the implementation of the project.
2. **Ease to use:** everyone can easily use the application in order to save time as the application is itself going to suggest the user with his next move

CHAPTER-1(INTRODUCTION)

1.1 Objective

The prime objective of the project i.e “Learning from personal data collected on mobile devices” is to create an interface for the user as to manage his/her whole personal data that is stored in his/her mobile .Call logs,gallery,messages,music files,calendars,location etc all are given on the home page so that he or she could able to manage them properly.

.1.2Brief Description

The main objective of the project named “Learning from personal data collected on mobile devices” is to explore the feasibility of mobile application. The application will be developed by using android .

1.3Android

1.3.1Introduction

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc.It contains a linux-based Operating System, middleware and key mobile applications.It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used. There are many code names of android such as Lollipop, Kitkat, Jelly Bean, Ice cream Sandwich, Froyo, Ecliar, Donut etc

1.3.2Features

1) It is open-source.

2) Anyone can customize the Android Platform.

3) There are a lot of mobile applications that can be chosen by the consumer.

4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

1.3.3History

Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003. In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation. The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**. Originally intended for camera but shifted to smart phones later because of low market for camera only.Android is the nick name of Andy Rubin given by coworkers because of his love to robots. In 2007, Google announces the development of android OS. In 2008, HTC launched the first android mobile.

1.3.4Architecture

Android architecture or Android software stack is categorized into five parts:

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.

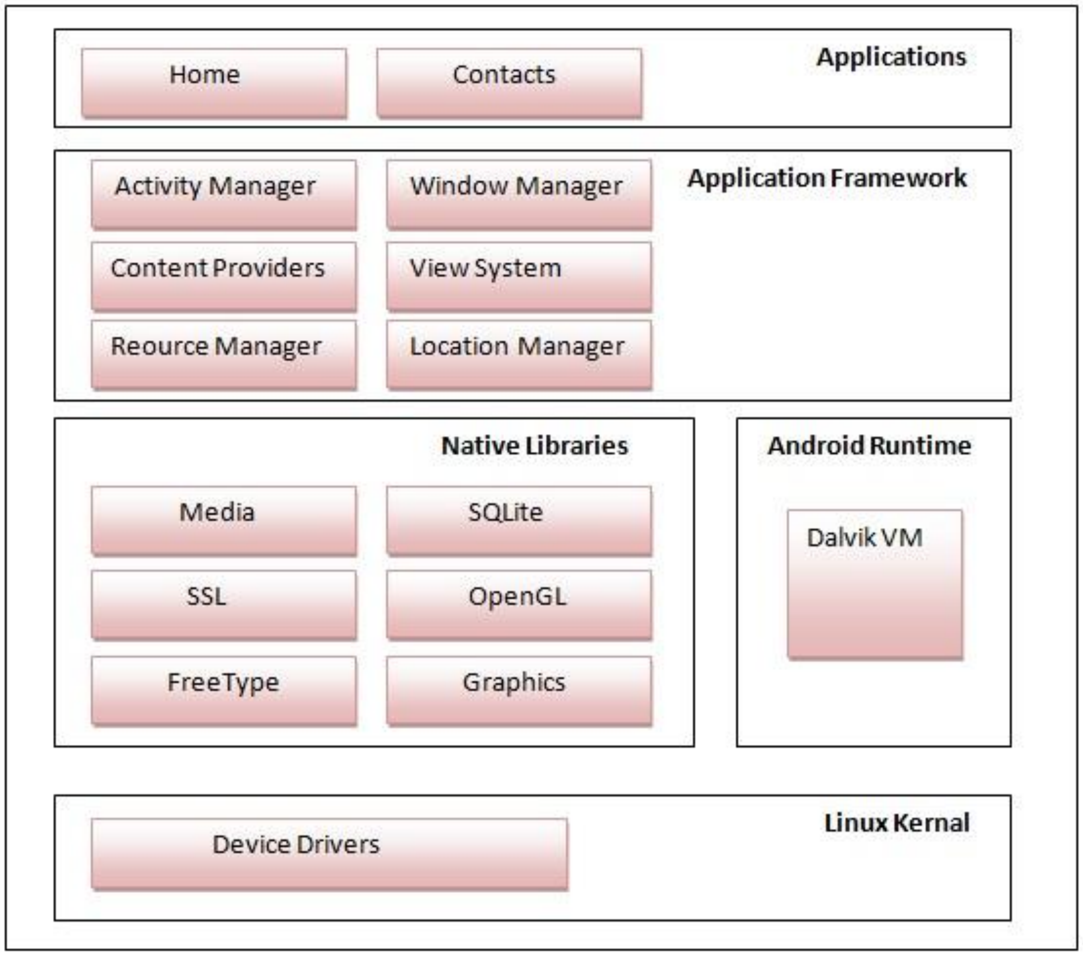


Fig1. Architecture

1) Linux Kernel

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, their are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's**such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using linux kernal.

1.3.5Core Building Blocks

An android **component** is simply a piece of code that has a well defined life cycle e.g. Activity, Receiver, Service etc.

The **core building blocks** or **fundamental components** of android are activities, views, intents, services, content providers, fragments and AndroidManifest.xml.

Activity

An activity is a class that represents a single screen. It is like a Frame in AWT.

View

A view is the UI element such as button, label, text field etc. Anything that you see is a view.

Intent

Intent is used to invoke components. It is mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Service

Service is a background process that can run for a long time.There are two types of services local and remote. Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

Content Provider

Content Providers are used to share data between the applications.

Fragment

Fragments are like parts of activity. An activity can display one or more fragments on the screen at the same time.

AndroidManifest.xml

It contains informations about activities, content providers, permissions etc. It is like the web.xml file in Java EE.

Android Virtual Device (AVD)

It is used to test the android application without the need for mobile or tablet etc. It can be created in different configurations to emulate different types of real devices.

CHAPTER-2

SOFTWARE REQUIREMENT & ANALYSIS

2.1 Purpose of SRS

This document describes the features and components of the project "Learning from personal data collected in mobile devices".This document is intended to serve as a guiding reference for the design and development stages of proposed system.

Thus, the project

1. Is based on android and is developed in java language
2. Includes codes to perform the tasks.

The document is targeted at system designers and the developers as a guiding reference.It will also be useful to end users in their good word.The SRS should fully describe the external behavior of the application or subsystem or subsystem identified. It also describes non functional requirements,design constraints and other factors necessary to provide a complete and comprehensive description of the requirements for the software.

It can also be used by application developers wishing to extend functionality by providing other features for the system.

2.2 System Requirement

1) Technology Used

Programming

i) Core Java

2) Software and Hardware Requirement

a) Software Requirement

i) Window 7 Professional /XP

ii) Eclipse IDE

iii) Android SDK

iv) Eclipse plugin

b) Hardware Requirement

i)Android Device

ii)Computer System

2.3 .Details of Tool Used

The technology selected for implementing the project is Core Java (Jdk 1.6).The development in windows environment.

Java

Java is the latest language in the field of OOP(Object oriented Programming).All the three concepts of OOP namely Abstraction, Inheritance and Encapsulation are accepted by Java. The various characteristics of java are:

1. Simple
2. Object-oriented
3. Compiled And interpreted
4. Portable
5. Dynamic
6. Multithreaded
7. Secure

My Eclipse

MyEclipse IDE refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, Python, PHP, Ruby, C, C++ and much more.

The MyEclipse IDE is written in Java and runs everywhere where a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages.

The MyEclipse Platform allows applications to be developed from a set of modular software components called modules. The MyEclipse IDE provides complete tools for all the latest Java EE 6 standards, including the new Java EE 6 Web Profile, Enterprise Java Beans (EJBs), servlets, Java Persistence API, web services, and annotations.MyEclipse also supports the Java Server Pages (JSP), Hibernate, Spring, and Struts frameworks, and the Java EE 5 and J2EE 1.4 platforms.

Android SDK

Android software development is the process by which new applications are created for the Android operating system. Applications are usually developed in the Java programming language using the Android Software Development Kit, but other development tools are available

CHAPTER-3(DESIGN)

3.1 Design Objective of Project

The various module to be developed in this application are following:

1. Home to open home page.
2. Calendars,Text messages,Location,music files,Gallery,call logs used to store the respective information contained in the mobile device .
3. Settings to reset all the information .

.

3.2 USE CASE Diagram

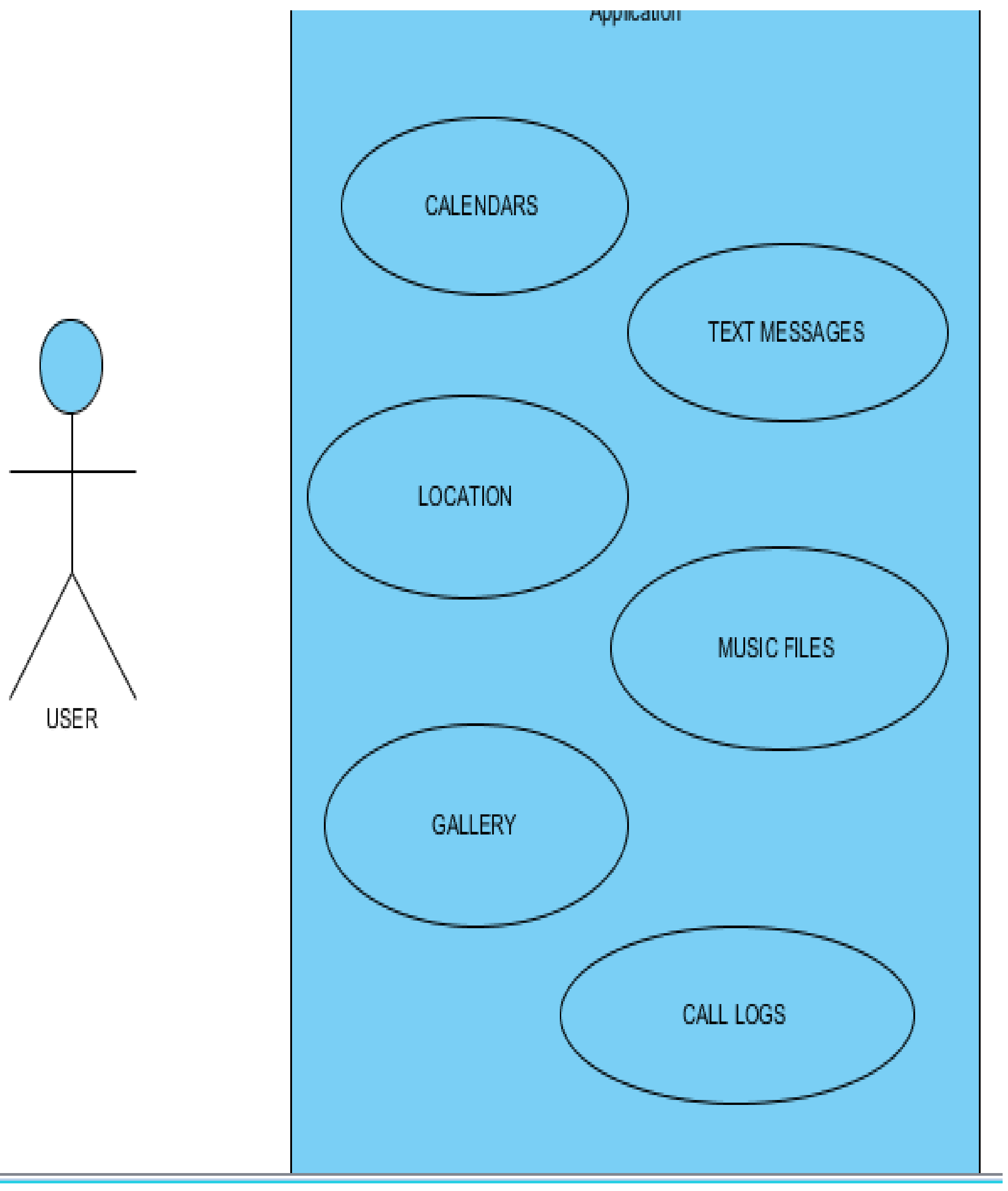


Fig.2 Use Case Diagram

3.3Screenshots

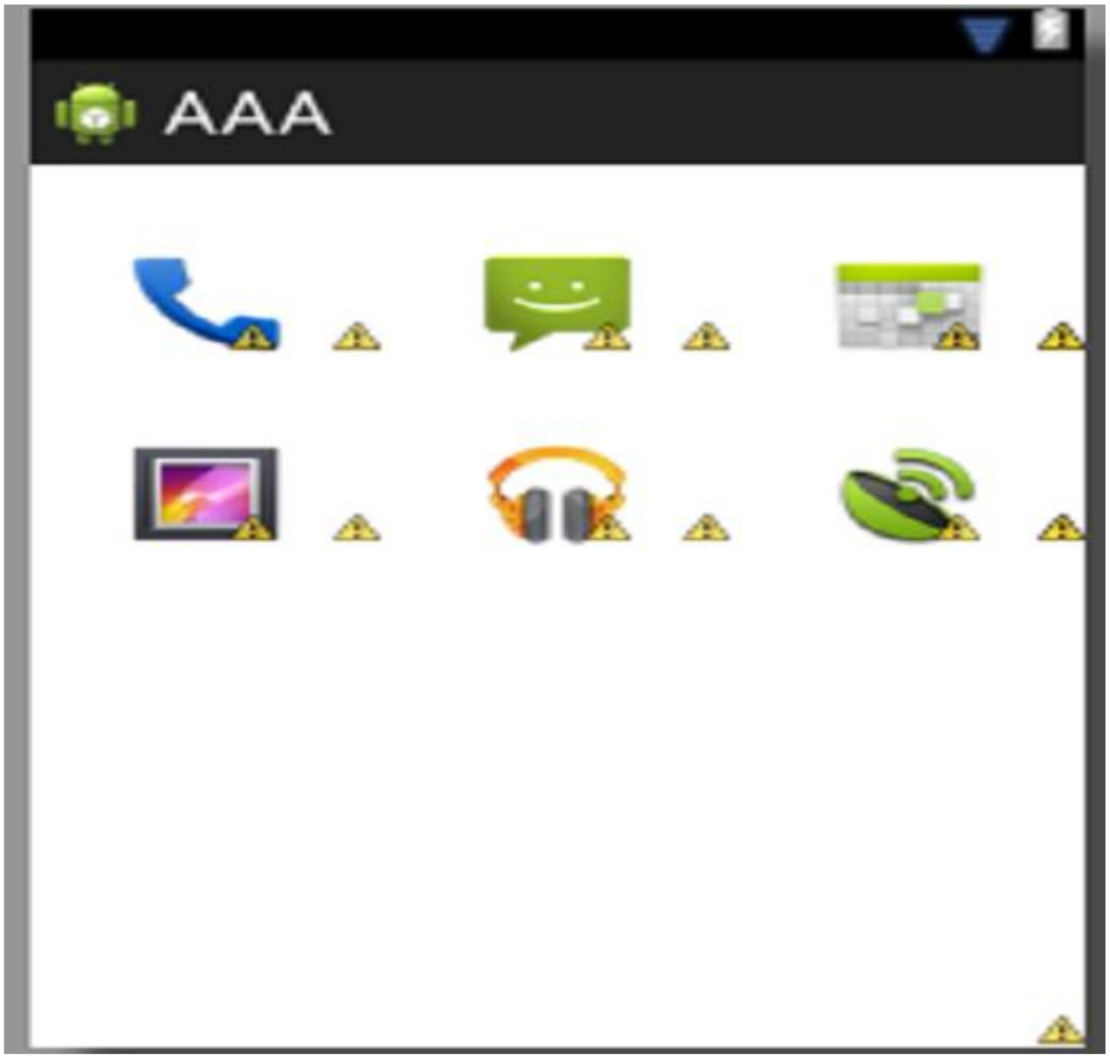


Fig 3.layout for home page

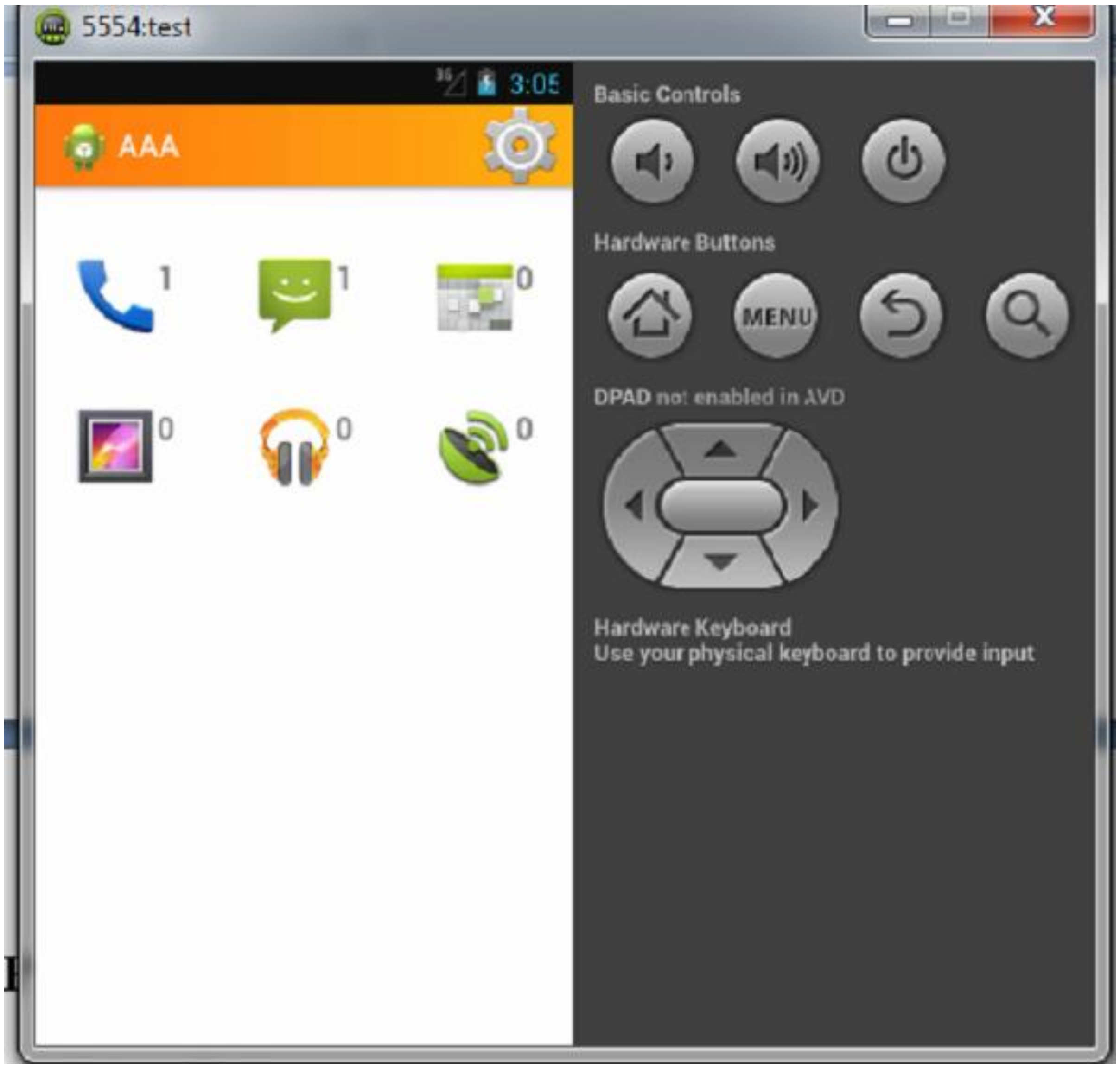
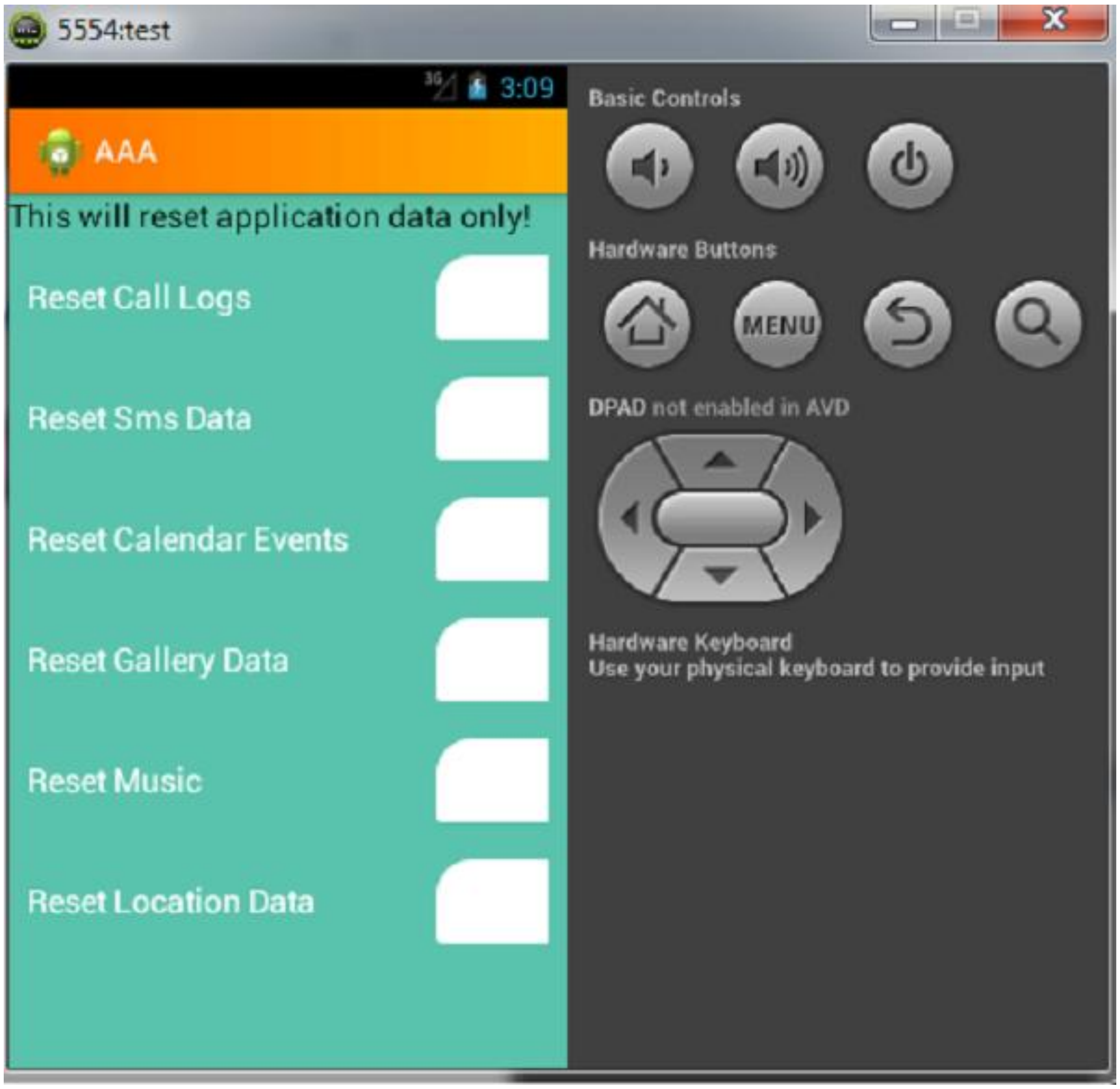


Fig4.home page



This will reset application data only!

Reset Call Logs

Reset Sms Data

Reset Calendar Events

Reset Gallery Data

Reset Music

Reset Location Data

Fig 5 Settings to reset all data

CHAPTER-4

CODING AND SNAPHOTS

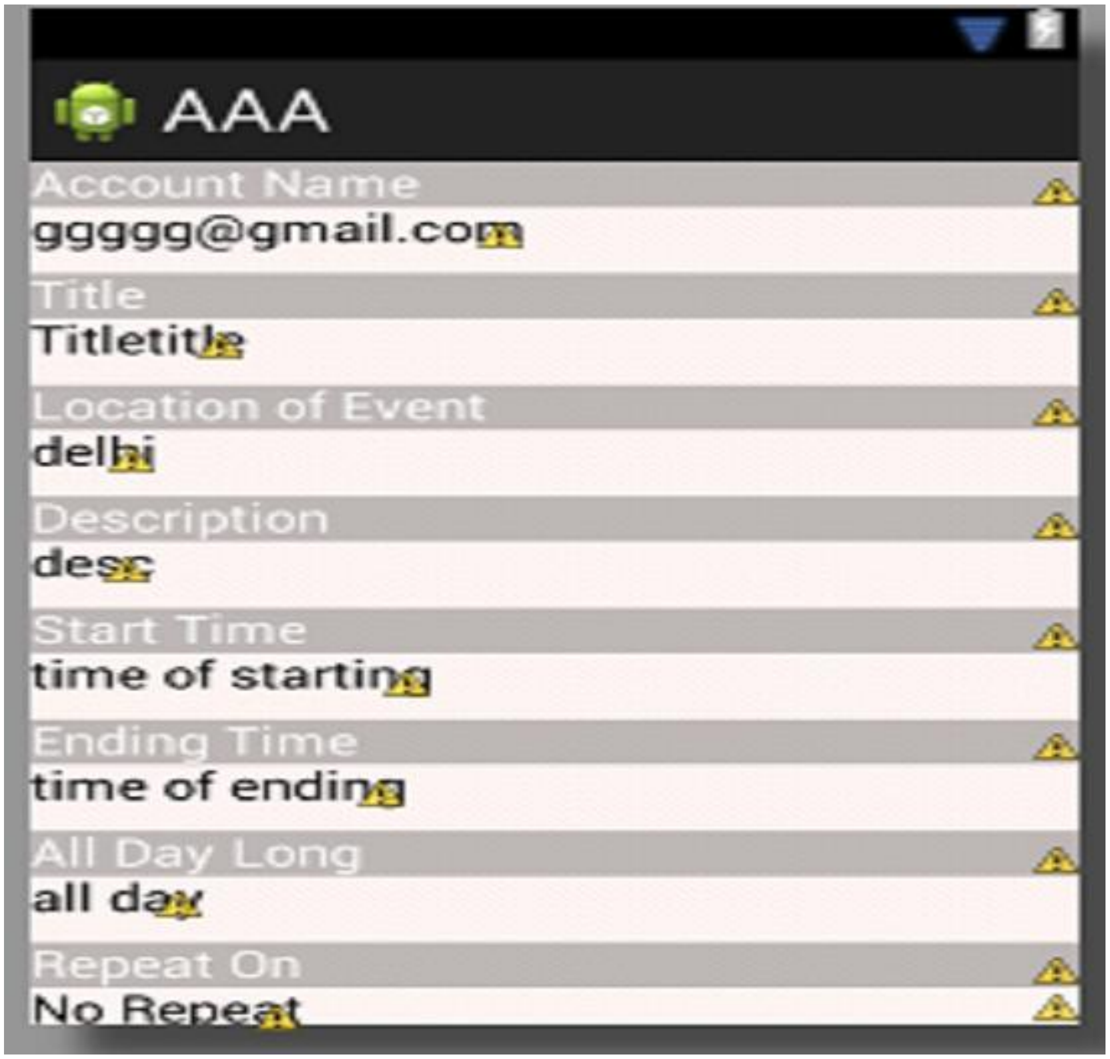
4.1Calendars

Android developers have been longing for an official Calendar app and content provider since Android has been released. With the release of Ice Cream Sandwich Google has finally added this feature to our tools list. Now we developers can use the Calendar app from within our Activities using Intents or we can access the data by making use of the new CalendarContract content provider. We obviously have calendars. The plural is no accident, because each user can manage multiple calendars on her device. For example her personal calendar, her work calendar, a calendar of her colleague and one for her sports team. Android uses colors to distinguish between different calendars in the Calendar app. For each calendar you can have multiple events. But each event belongs to exactly one calendar only. If the event is visible twice (e.g. in the private and in the business account of the user), the event is stored twice, each belonging to their respective calendars.Many events are recurring. So in addition to events there are also event instances. An instance represents the specific event that takes place at just this one point in time. For a single event Android creates one instances entry, for recurring events Android creates as many instance entries as there are occurrences of this event.Events can of course have multiple attendees as well as multiple reminders.To use the Calendar content provider you need to declare the necessary permissions within your manifest file first:

```
<uses-permission   android:name="android.permission.READ_CALENDAR"/>

<uses-permission   android:name="android.permission.WRITE_CALENDAR"/>
```

Keep in mind that the Calendar is part of the official API only from Ice Cream Sandwich onwards. The constants, defined in CalendarContract and its many inner classes are not available prior to API version 14. All code samples shown here need an API-level of 14 or higher as build target and the value for the android:minSdkVersion attribute of the <uses-sdk> element in the manifest file.

4.1.1Screenshots



AAA

Account Name

ggggg@gmail.com

Title

Titletitle

Location of Event

delhi

Description

desc

Start Time

time of starting

Ending Time

time of ending

All Day Long

all day

Repeat On

No Repeat

Fig6 layout for calendar

4.2Text Messages

- Using the Content Resolver, you can read both inbox and sent items. If you want to read the inbox or sent items alone then you specify it in content resolver.
- Adding Permissions : For reading your sms, the first thing you need to do is add the appropriate permissions to the Android manifest.xml.

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

IncomingSms.java pseudocode :

```
    1. Created class IncomingSms with extends BroadcastReceiver class
   public class IncomingSms extends BroadcastReceiver
2. Get the object of SmsManager to find out received sms details
  // Get the object of SmsManager
       final SmsManager sms = SmsManager.getDefault();
 3.  Create method receiver()
  public void onReceive(Context context, Intent intent)
  4. Get / Read current Incomming SMS data
 // Retrieves a map of extended data from the intent.
       final Bundle bundle = intent.getExtras();
   try {
          if (bundle != null) {
               final Object[] pdusObj = (Object[]) bundle.get("pdus");
               for (int i = 0; i < pdusObj.length; i++) {
                    SmsMessage currentMessage = SmsMessage.createFromPdu((byte[])
pdusObj[i]);
                   String phoneNumber = currentMessage.getDisplayOriginatingAddress();
                  String senderNum = phoneNumber;
                  String message = currentMessage.getDisplayMessageBody();
Log.i("SmsReceiver", "senderNum: "+ senderNum + "; message: " + message);
                   // Show alert
                   int duration = Toast.LENGTH_LONG;
                   Toast toast = Toast.makeText(context, "senderNum: "+ senderNum + ",
message: " + message, duration);
                   toast.show();

               } // end for loop
             } // bundle is null

        } catch (Exception e) {
            Log.e("SmsReceiver", "Exception smsReceiver" +e);
             }
```

4.2.2Screenshots

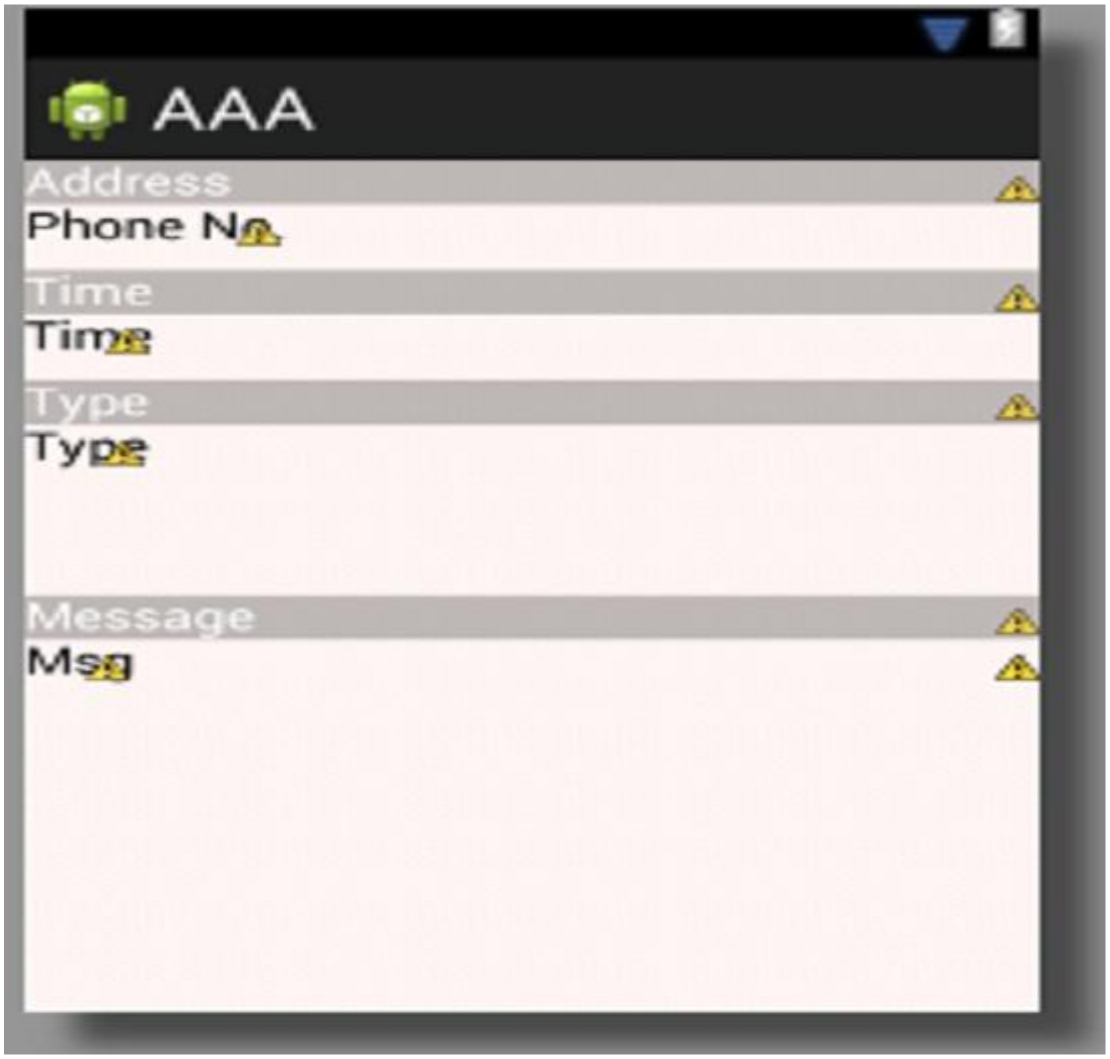
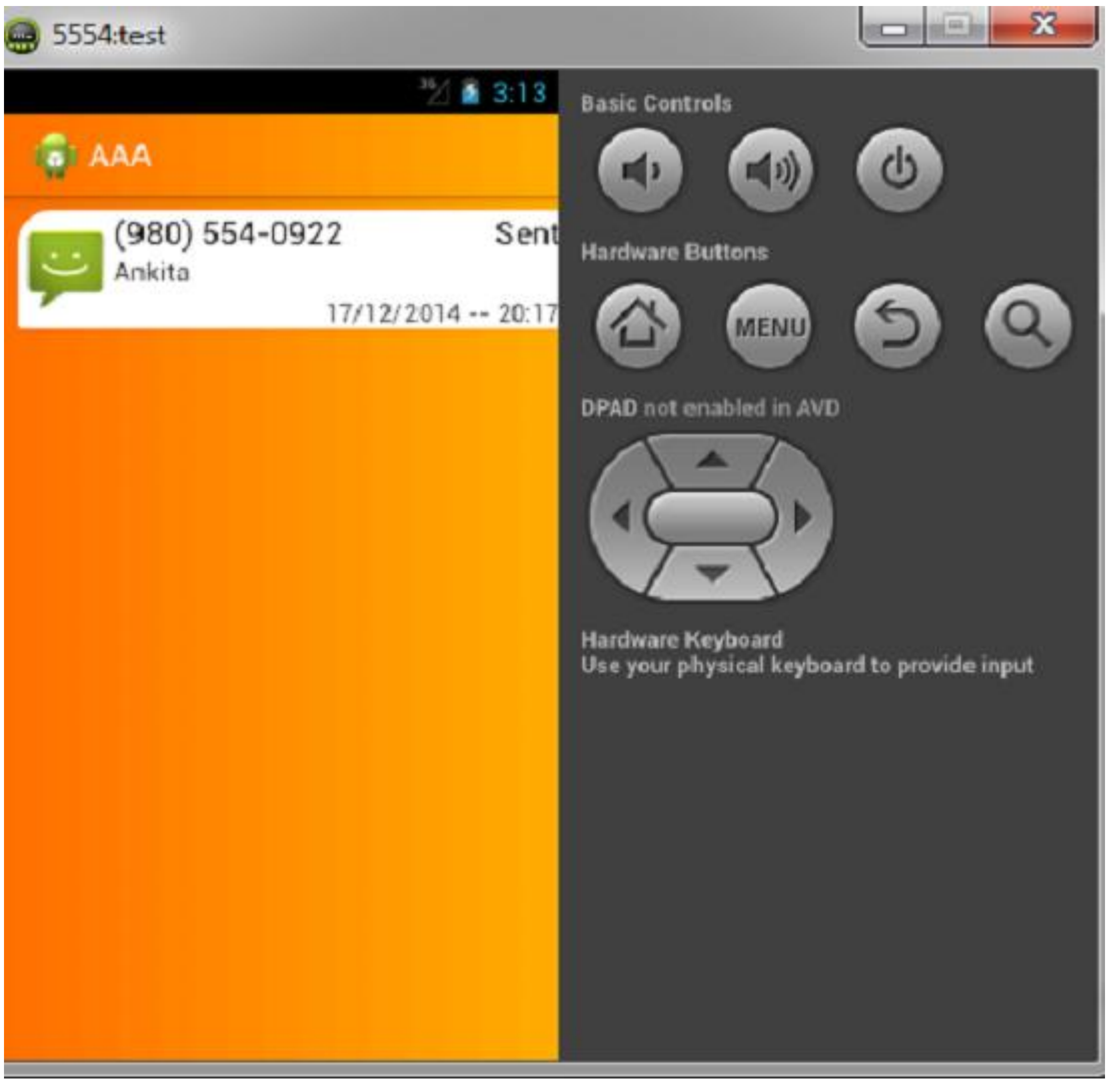
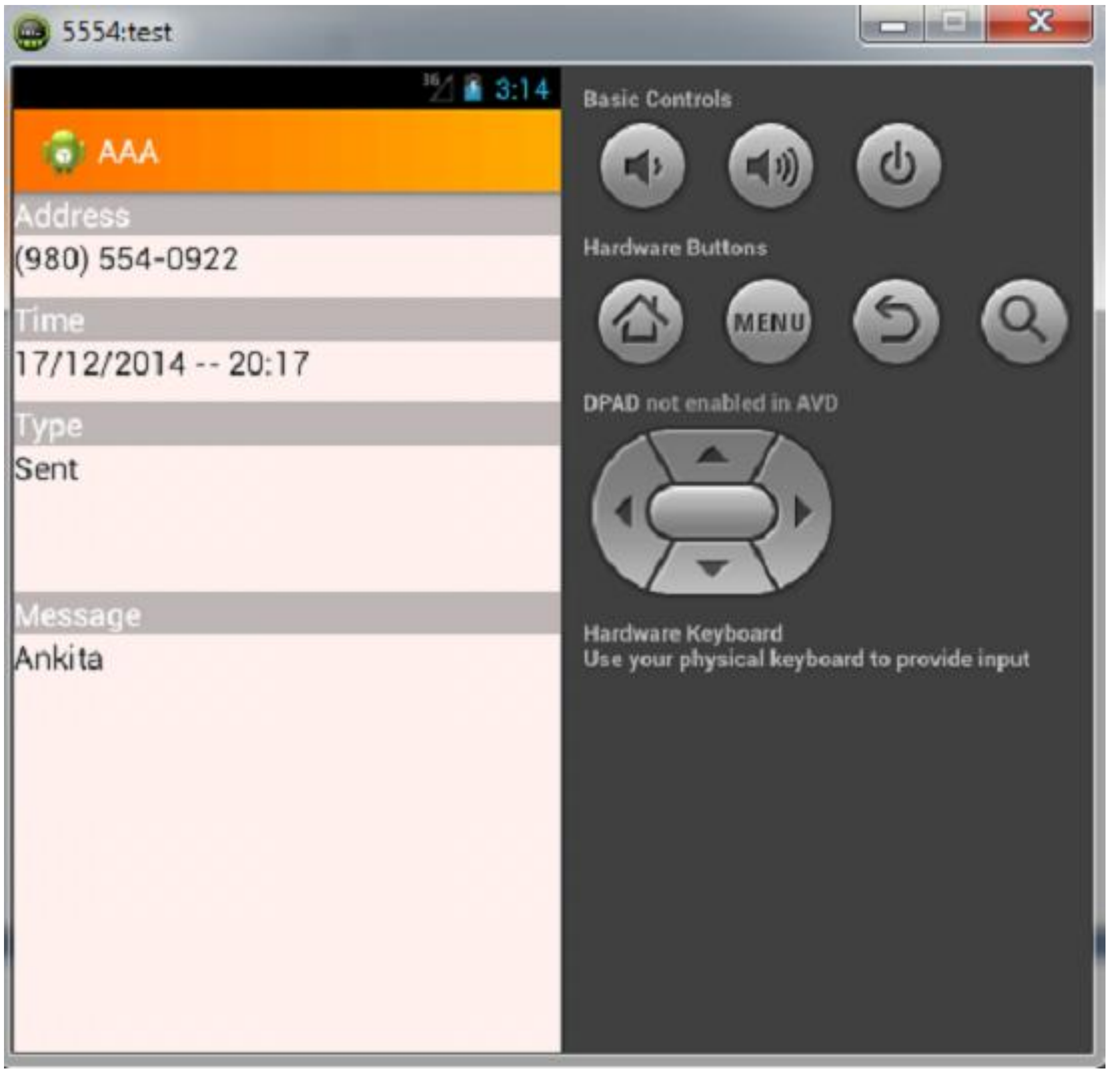


Fig7 layout for text messages



(980) 554-0922 Sent
Ankita
17/12/2014 -- 20:17

Fig 8 text messages



5554:test

AAA

Address

(980) 554-0922

Time

17/12/2014 -- 20:17

Type

Sent

Message

Ankita

Fig 9 second layout for text messages

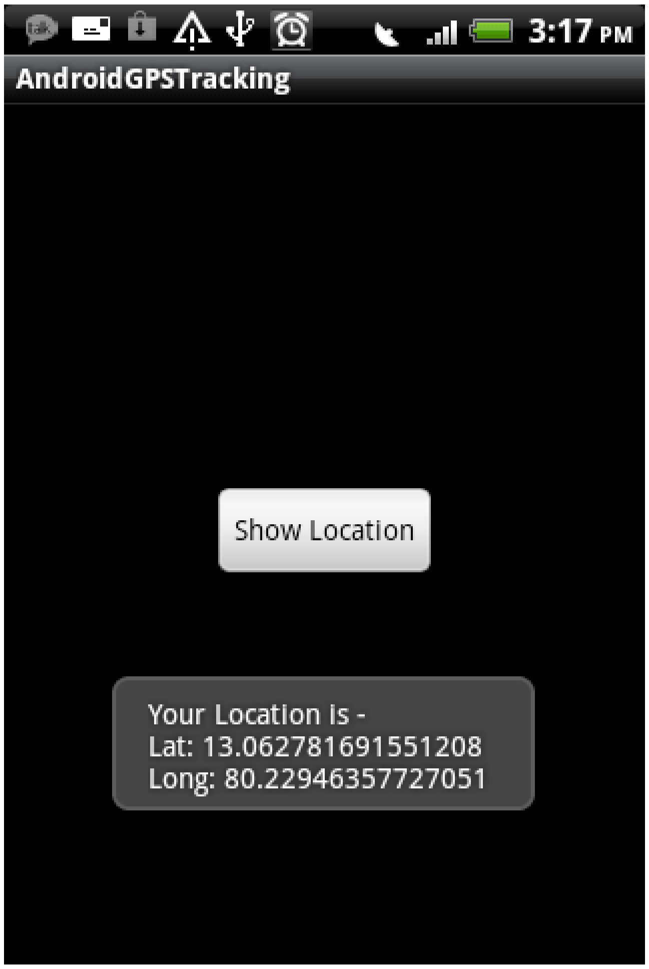
4.3Locations

- Android contains the android.location package which provides the API to determine the current geo position.
- The LocationManager class provides access to the Android location service. This services allows to access location providers, to register location update listeners and proximity alerts and more.
- The LocationProvider class is the super class of the different location providers which deliver the information about the current location. This information is stored in the Location class. The Android device might have several LocationProvider available and you can select which one you want to use. In most cases you have the following LocationProvider available i.e network ,gps **.**
- Selecting LocationProvider via Criteria **:** For a flexible selection of the best location provider use a Criteria object, in which you can define how the provider should be selected. You can register a Location Listener object with the LocationManager class to receive periodic updates about the geoposition.
- Proximity Alert **:** You can also register an Intent which allows to define a proximity alert, this alert will be triggered if the device enters an area given by a longitude, latitude and radius (proximity alert).
- Forward and reverse Geocoding **:** The Geocoder class allows to determine the geo-coordinates (longitude, laditude) for a given address and possible addresses for given geo-coordinates.
- Security **:** If you want to access the GPS sensor, you need the ACCESS_FINE_LOCATION permission. Otherwise you need theACCESS_COARSE_LOCATION permission.
- Prompt the user to Enabled GPS **:** The user can decide if the GPS is enabled or not. You can find out, if a LocationManager is enabled via the isProviderEnabled() method. If its not enabled you can send the user to the

settings via an intent with the Settings.ACTION_LOCATION_SOURCE_SETTINGS action for the android.provider.Settings class.

Fig10

Android Launching GPS Setting Options AndroidHive



Your Location is -
Lat: 13.062781691551208
Long: 80.22946357727051

4.4Call Logs

If you want to act on an **incoming** phone call and do something with the incoming number ,this can be achieved by implementing a broadcast Receiver and listening for a TelephonyManager.ACTION_PHONE_STATE_CHANGED action.

If you want to do something when the phone rings you have to implement a *broadcast receiver* which listens for
the TelephonyManager.ACTION_PHONE_STATE_CHANGED intent action. This is a broadcast intent action indicating that the call state (cellular) on the device has changed.
1. Create a class IncomingCallInterceptor which extends BroadcastReceiver.
2. Override the onReceive method to handle incoming broadcast messages.
3. The EXTRA_STATE intent extra in this case indicates the new call state.
4. If (and only if) the new state is RINGING, a second intent
extra EXTRA_INCOMING_NUMBER provides the incoming phone number as a String.
5. We extract the number information from the EXTRA_INCOMING_NUMBER intent extra.
6. We have to register our IncomingCallInterceptor as a <receiver> within
the <application> element in the AndroidManifest.xml file.
7. We register an <intent-filter> …
8. and an <action value which registers our receiver to listen
for TelephonyManager.ACTION_PHONE_STATE_CHANGED broadcast messages.
9. Finally we have to register a <uses-permission> so we are allowed to listen to phone state changes.

In case if two receivers listen for phone state changes

In general, a *broadcast message* is just that, a message which is sent out to many receivers at the same time. This is the case for *normal broadcast* messages which is used to send out the ACTION_PHONE_STATE_CHANGED intent as well. All receivers of the broadcast are run in an undefined order, often at the same time and for that reason *order* is not applicable.

If you want to detect outgoing phone call event,steps for doing so are

1) create OutgoingCallBroadcastReceiver

```
public class OutgoingCallReceiver extends BroadcastReceiver {

public void onReceive(Context context,Intent intent) {

 Log.d(OutgoingCallReceiver.class.getSimpleName(),intent.toString());

Toast.makeText(context,”Outgoing call catched!”,Toast.LENGTH_LONG).show();

 }}
```

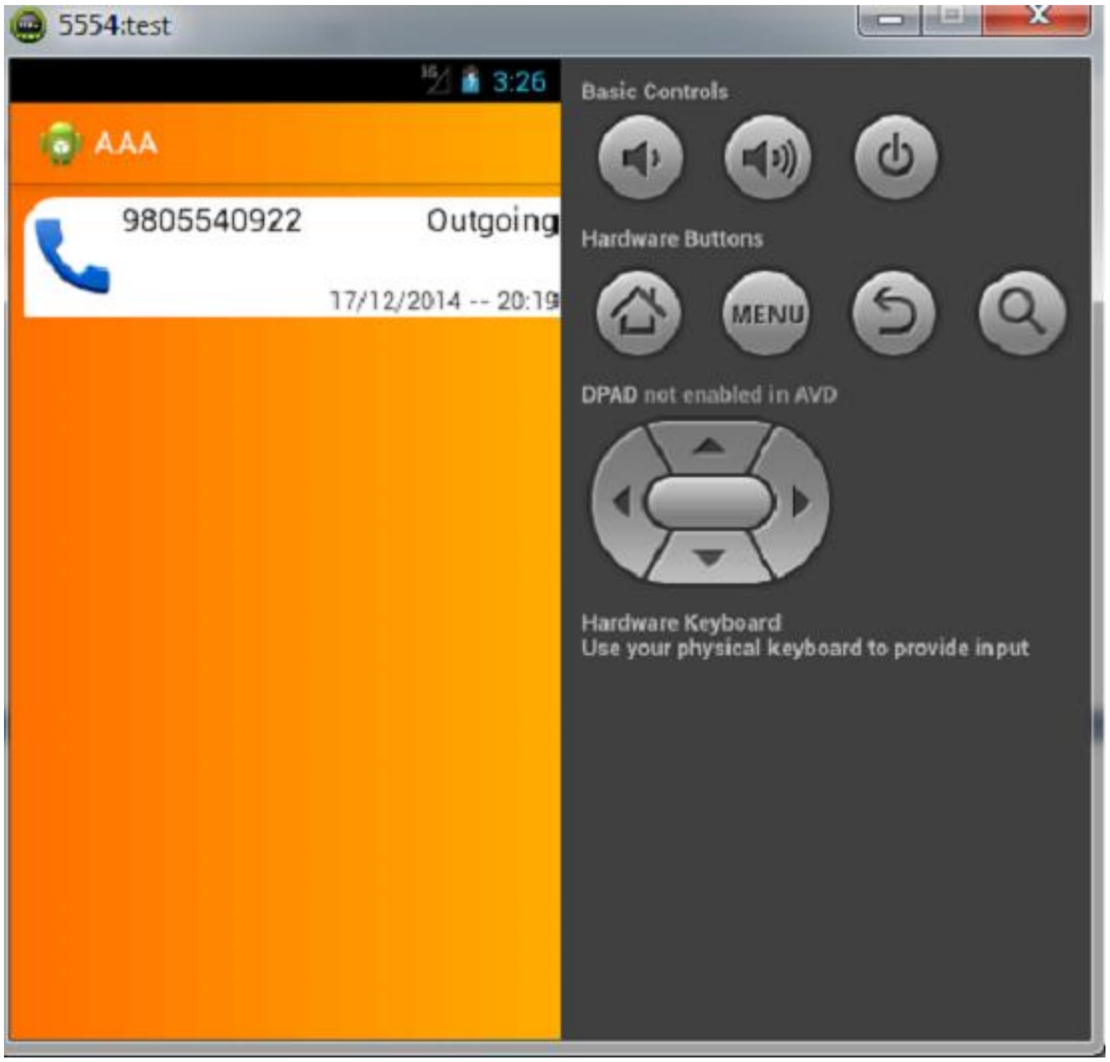
2)Register OutgoingCallBroadcastReceiver in AndroidManifest.xml

```
<receiver android:name=”.OutgoingCallReceiver”> <intent-filter>

<action android:name=”android.intent.action.NEW_OUTGOING_CALL” /></intent-filter>

</receiver>
```

3)Add permission in AndroidManifest.xml

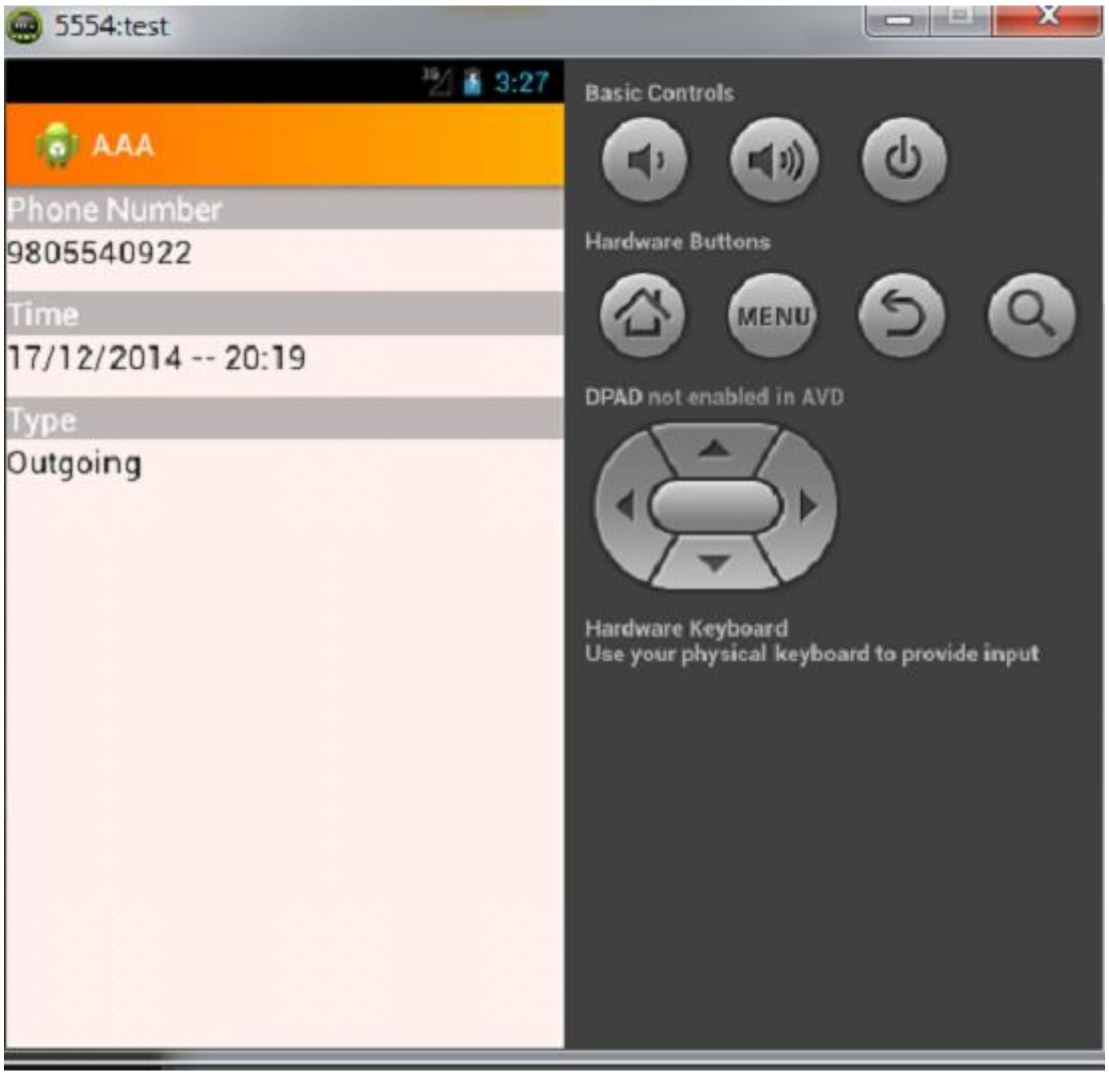
```
            <uses-permission
android:name=“android.permission.PROCESS_OUTGOING_CALLS”/>
```

4.4.1Screenshots



9805540922 Outgoing
17/12/2014 -- 20:19

Fig 11 call logs page


5554:test

3:27

AAA

Phone Number

9805540922

Time

17/12/2014 -- 20:19

Type

Outgoing

Fig12 second page for call logs

4.5Music Files

- Building the music player will involve using the ContentResolver and Cursor classes to retrieve tracks on the device.
- MediaPlayer class to play audio and the MediaController class to control playback.
- Service instance is used to play audio when the user is not directly interacting with the app.
- Adapter instance is used to present the songs in a list view,starting playback when the user taps an item from the list.
- Adding Permission: add the appropriate permission to the Android manifest.xml.

<uses-permission android:name="android.permission.WAKE_LOCK"/>

This permission will let music playback continue when the user's device become idle.

a music player on Android can be created using the MediaPlayer and MediaController classes. We presented the list of songs on the user device and specified a method to execute when the user makes a selection. Also we will implement a Service class to execute music playback continuously, even when the user is not directly interacting with the application.

4.5.1Screenshots

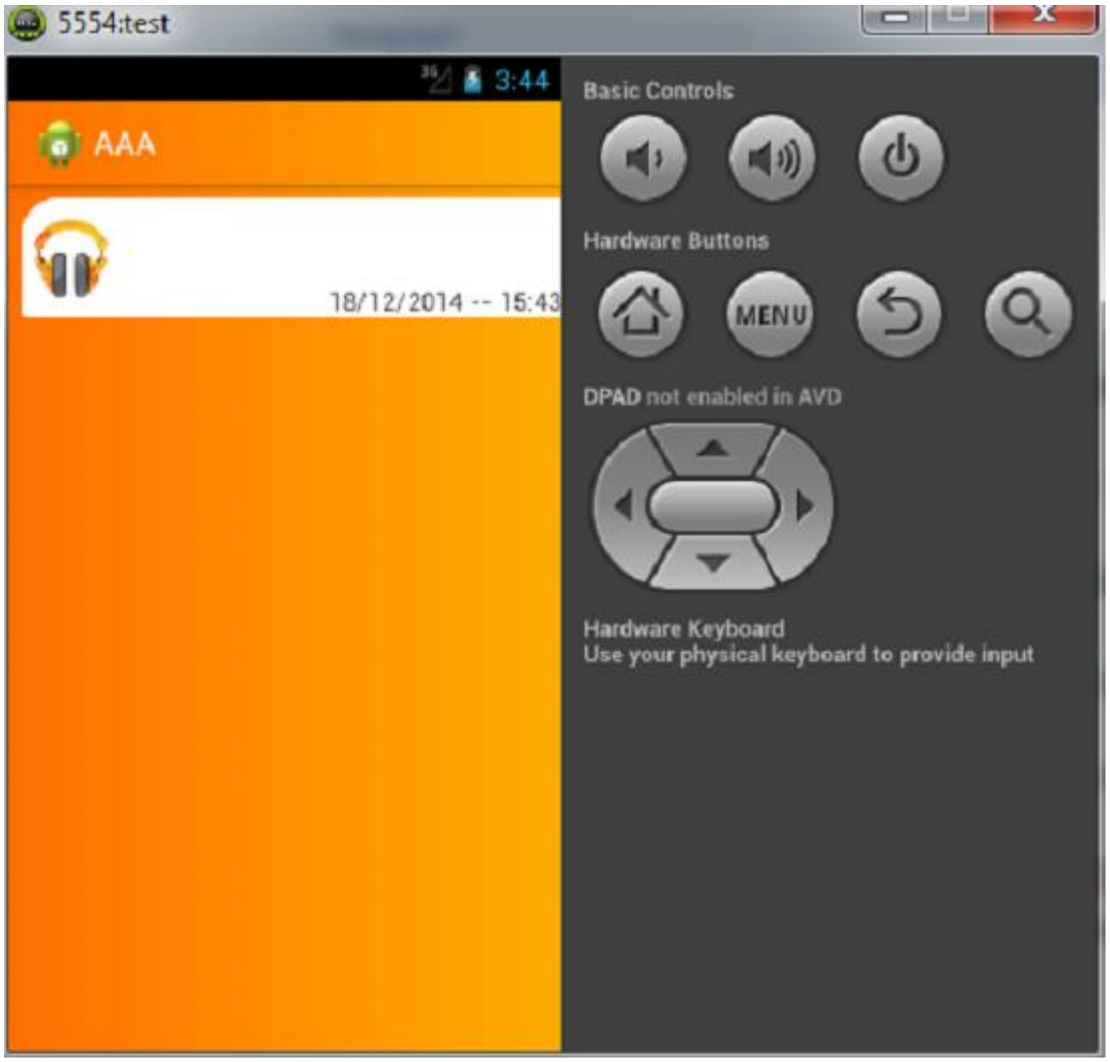
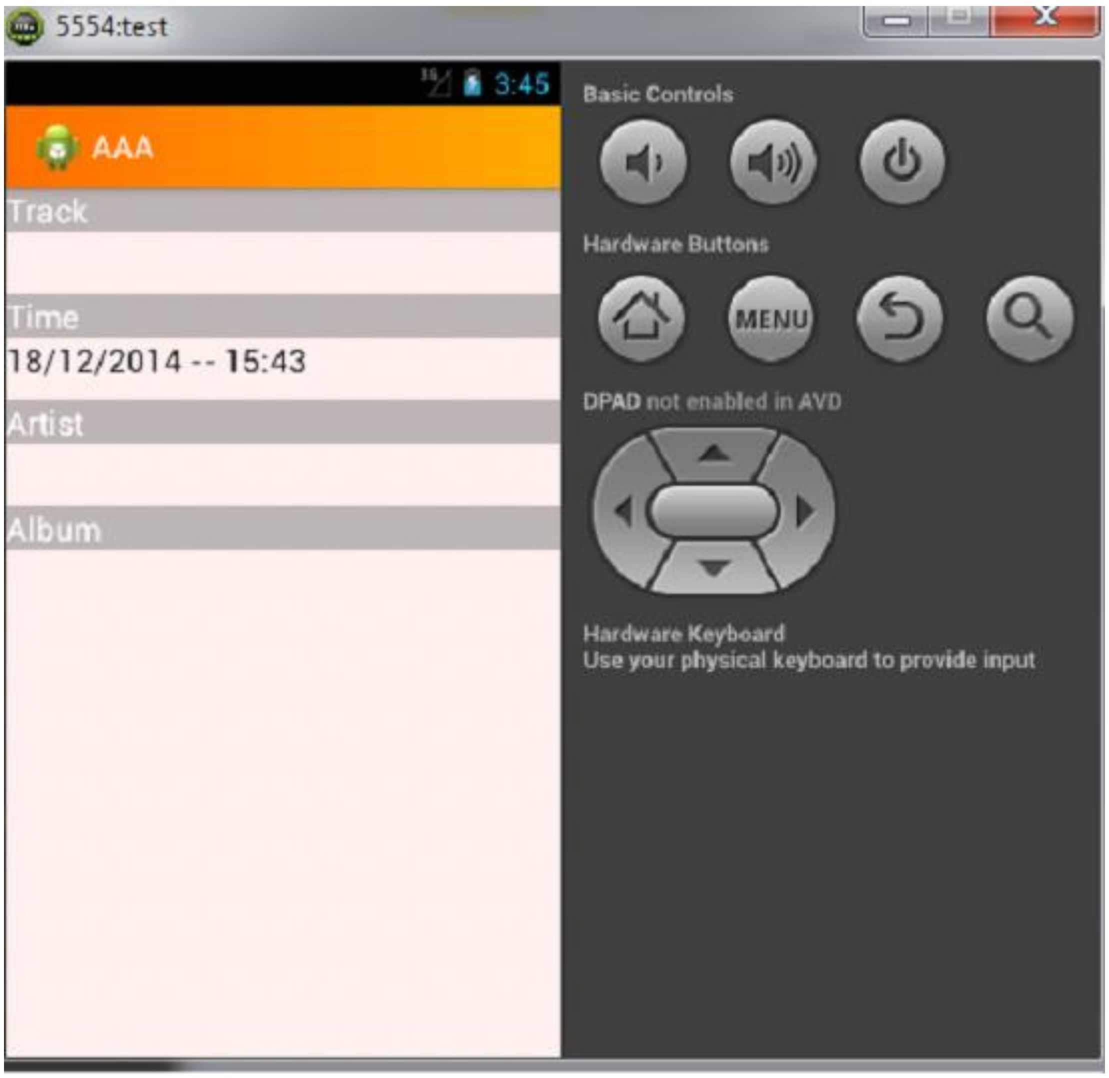


Fig13 page for music files



Track

Time

18/12/2014 -- 15:43

Artist

Album

Fig 14 second page for music files

4.6Gallery

- It will be done using the ContentResolver and Cursor classes to retrieve pictures on the device.
- Register your broadcast receiver in AndroidManifest

```
< receiver
    android:name=“.Class_name”
    android:enabled=“true”>
```

and then finally create a class and extend it using BroadcastReceiver

4.6.1Screenshots

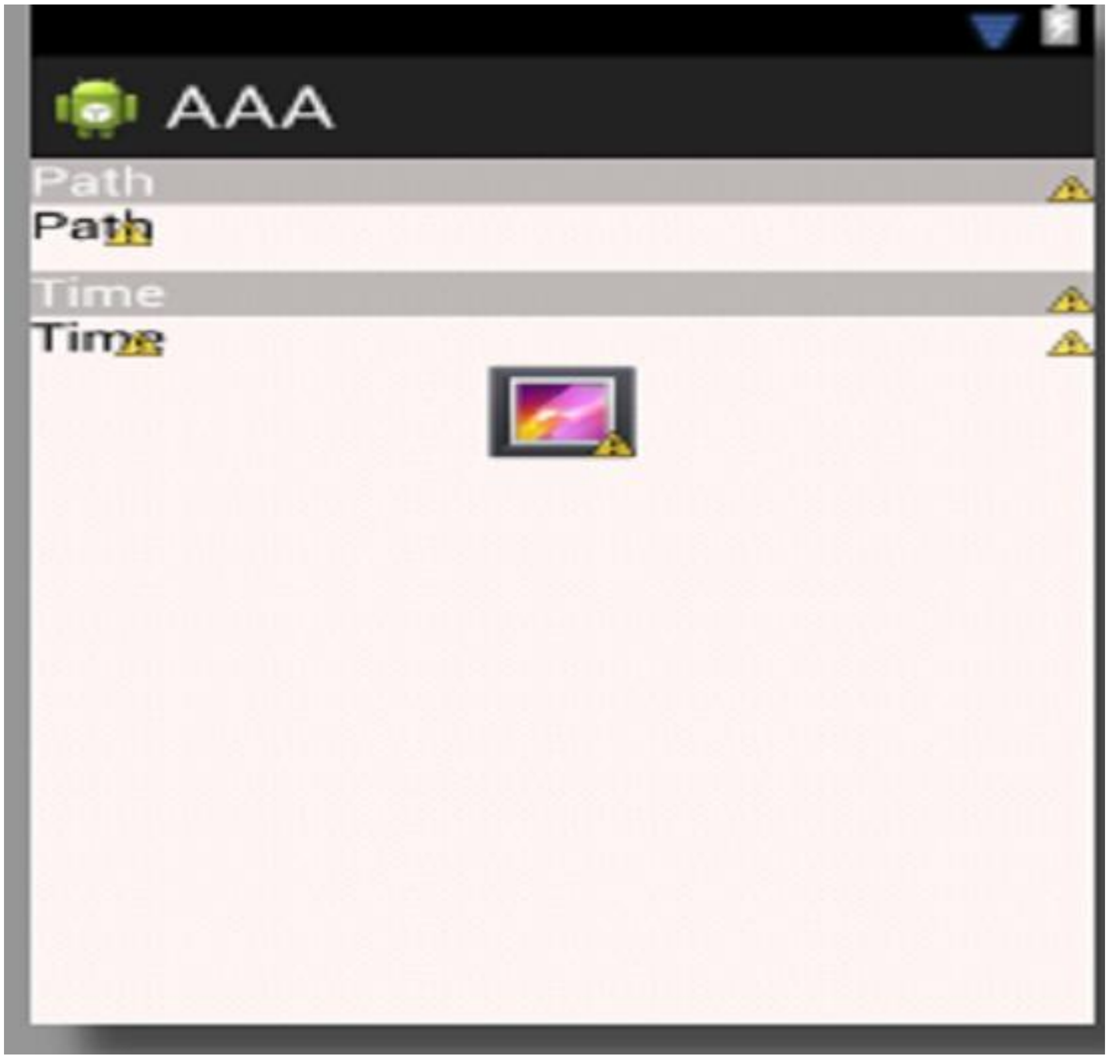


Fig15 page for gallery

CHAPTER-5

DATA MINING

5.1 Abstract

Smartphones can collect considerable context data about the user, ranging from apps used to places visited. Frequent user patterns discovered from longitudinal, multi-modal context data could help personalize and improve overall user experience. Our long term goal is to develop novel middleware and algorithms to efficiently mine user behavior patterns entirely on the phone by utilizing idle processor cycles. Mining patterns on the mobile device provides better privacy guarantees to users, and reduces dependency on cloud connectivity .As an important step in this direction, we develop a novel general-purpose service that runs on the phone and discovers frequent co-occurrence patterns indicating which context events frequently occur together. Further, we find interesting behavior patterns for individual users and across users, ranging from calling patterns to place visitation patterns. Finally, we show how our co-occurrence patterns can be used by developers to improve the phone UI for launching apps or calling contacts.

5.2 Introduction

Smartphones can collect and infer rich contextual data about users and how they use their phones. Using public APIs and apps on most smartphone platforms, we can easily log raw contextual data about the user such as her location, application usage, online activity, call and SMS behavior, charging behavior, and battery usage.

User behavior patterns can be expressed in a number of different ways. For example, sequence mining algorithms may uncover sequences of user contexts that occur frequently, while statistical correlation functions may express interesting relationships between numerical context data such as activity level and sleep quality. Our longterm vision is to use longitudinal smartphone context to infer diverse frequent patterns that capture different aspects of the user's behavior, and explore the utility of each type of pattern in improving overall user experience. A key aspect of our vision is to leverage the computing potential of modern smartphones to perform the pattern mining entirely on the device.

5.3 Why mine co-occurrence patterns

For example, the first pattern could be expressed in an association rule as {Morning, Breakf ast, AtHome} → {ReadNews}, and indicates that the user typically reads news apps on the phone whenever she has breakfast at home in the morning. Such a pattern could be used preload news content and news apps to memory ahead of time to reduce loading delays and improve user experience. we can provide convenient UI shortcuts to typical user tasks performed at work in the afternoon, such as calling Alex, or opening a project folder. Finally, we can even try to expose and alter frequent patterns that are not beneficial to the user; for example, a smart reminder to charge the phone before the user typically goes to sleep with a half-empty battery. A key benefit of using association rules is that they can be easily read and understood by both end users and developers, and even be used in simple if-this-then-that style mobile recipes, compared to potentially more accurate but less readable classification models.

5.4 Why Mine Patterns on the Mobile Device?

Modern smartphones have powerful quad-core processors and are also typically unused for a majority of time such as at night when the user is sleeping and the phone ischarging. Compute-intensive pattern mining algorithms may be run on the phone periodically during this idle time with little or no impact on the end user. The average idle time per day on weekdays and weekends for 6 users over 2 months of smartphone usage. We define the phone to be idle whenever it is charging, there are no foreground applications, and the battery level is at least 80%.that users have between 1-10 hours of idle time each day, which may be aggregated across multiple days for mining algorithms that do not need to refresh their patterns every day. In fact, we show later in the paper that multiple instances of mining algorithms may be run simultaneously on a quadcore smartphone with little or no impact to user experience, thus increasing the mining capacity during the idle time. Another key reason to run mining algorithms on the phone is to provide better privacy guarantees to the user by ensuring that personal context data is not transmitted to the cloud for mining. Our approach is complementary to parallel efforts to improve the privacy guarantees of sharing personal context data with the cloud. Finally, mining patterns on the device may also provide benefits to users with lower-end phones in developing or remote regions, where cloud connectivity and data plans may be limited or absent.

5.5 Coding and snapshots

```
package u.aaa;

import java.text.DateFormatSymbols;

import java.text.FieldPosition;

import java.text.Format;

import java.text.NumberFormat;

import java.text.ParseException;

import java.text.ParsePosition;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Date;

import java.util.Iterator;

import java.util.List;

import u.calendarpack.CalendarHelper;

import u.calendarpack.CalendarObject;

import u.callpack.CallHelper;

import u.callpack.CallObject;

import u.gall.GalleryHelper;

import u.gall.GalleryObject;

import u.music.MusicHelper;
import u.music.MusicObject;

import u.smspack.SmsHelper;

import u.smspack.SmsObject;

import android.annotation.SuppressLint;

import android.app.Activity;

import android.content.Intent;

import android.graphics.Color;

import android.graphics.Paint;

import android.graphics.PointF;

import android.os.Bundle;

import android.util.Log;

import android.util.Pair;

import android.view.MotionEvent;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemSelectedListener;

import android.widget.ArrayAdapter;

import android.widget.CheckBox;

import android.widget.CompoundButton;

import android.widget.SeekBar;

import android.widget.Spinner;
import android.widget.Toast;

import com.androidplot.LineRegion;

import com.androidplot.ui.AnchorPosition;

import com.androidplot.ui.SeriesRenderer;

import com.androidplot.ui.SizeLayoutType;

import com.androidplot.ui.SizeMetrics;

import com.androidplot.ui.TextOrientationType;

import com.androidplot.ui.widget.TextLabelWidget;

import com.androidplot.util.PixelUtils;

import com.androidplot.xy.*;

import com.androidplot.xy.BarRenderer.BarRenderStyle;

import com.androidplot.ui.XLayoutStyle;

import com.androidplot.ui.YLayoutStyle;


/**

 * The simplest possible example of using AndroidPlot to plot some data.

 */

@SuppressLint("SimpleDateFormat")

public class GraphActivity extends Activity

{	String[] hours = {"0","1","2","3","4","5","6","7","8","9","10","11"

			,"12","13","14","15","16","17","18","19","20","21","22","23"}
```

```
int num;
    private XYPlot plot;
    private XYSeries series2;
    List<Integer> data;
        String[] xLabels;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
    super.onCreate(savedInstanceState);
        setContentView(R.layout.graphactivity);
        Intent g=getIntent();
        num  = g.getIntExtra("i",999);
                switch (num) {
                        case 0:
                        data = new ArrayList<Integer>(24);
                                for(int x=0;x<24;x++){
                                        data.add(0);
                                }
                                SimpleDateFormat formatter = new
SimpleDateFormat("dd/MM/yyyy "+"--"+" HH:mm");
            CallHelper helper = new CallHelper(this);
```

```
ArrayList<CallObject> calls = (ArrayList<CallObject>) helper.getAllCall();

Iterator<CallObject> it = calls.iterator();

while(it.hasNext()){

        CallObject c= it.next();

        try {

                Date parsedDate = formatter.parse(c.getTime());

                int s = parsedDate.getHours();

                data.set(s, data.get(s)+1);

                } catch (ParseException e) {

                        // TODO Auto-generated catch block

                        e.printStackTrace();

                }

}

drawPlot();

        plot.setRangeLabel("Calls");

                break;

        case 1:

                data = new ArrayList<Integer>(24);

                for(int x=0;x<24;x++){

                        data.add(0);

                }
```

```
                        SimpleDateFormat formatter1 = new
SimpleDateFormat("dd/MM/yyyy "+"--"+" HH:mm");

        SmsHelper helper1 = new SmsHelper(this);

        ArrayList<SmsObject> sms = (ArrayList<SmsObject>) helper1.getAllSMS();

        Iterator<SmsObject> it1 = sms.iterator();

        while(it1.hasNext()){

                SmsObject c= it1.next();

                try {

                        Date parsedDate = formatter1.parse(c.getTime());

                        int s = parsedDate.getHours();

                        data.set(s, data.get(s)+1);

                        } catch (ParseException e) {

                                // TODO Auto-generated catch block

                                e.printStackTrace();

                        }

        }

        drawPlot();


        plot.setRangeLabel("Sms");


                        break;
```

```
                    case 3:


                            data = new ArrayList<Integer>(24);

                            for(int x=0;x<24;x++){

                                    data.add(0);

                            }

                            SimpleDateFormat formatter3 = new
SimpleDateFormat("dd/MM/yyyy "+"--"+" HH:mm");

            GalleryHelper helper3 = new GalleryHelper(this);

            ArrayList<GalleryObject> gal = (ArrayList<GalleryObject>)
helper3.getAllGallery();

            Iterator<GalleryObject> it3 = gal.iterator();

            while(it3.hasNext()){

                    GalleryObject c= it3.next();

                    try {

                            Date parsedDate = formatter3.parse(c.getTime());

                            int s = parsedDate.getHours();

                            data.set(s, data.get(s)+1);

                            } catch (ParseException e) {

                                    // TODO Auto-generated catch block

                                    e.printStackTrace();
                    }
            }
            drawPlot();


            plot.setRangeLabel("Gallery");


                    break;


                case 4:


                    data = new ArrayList<Integer>(24);
                    for(int x=0;x<24;x++){
                        data.add(0);
                    }
                    SimpleDateFormat formatter4 = new
SimpleDateFormat("dd/MM/yyyy "+"--"+" HH:mm");
            MusicHelper helper4 = new MusicHelper(this);
            ArrayList<MusicObject> music = (ArrayList<MusicObject>)
helper4.getAllMusic();
            Iterator<MusicObject> it4 = music.iterator();
            while(it4.hasNext()){
                MusicObject c= it4.next();
                try {
                        Date parsedDate = formatter4.parse(c.getTime());
                        int s = parsedDate.getHours();
                        data.set(s, data.get(s)+1);
                        } catch (ParseException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
        }
        drawPlot();

        plot.setRangeLabel("Music");

                        break;

  }

}

void drawPlot(){
   // initialize our XYPlot reference:

plot = (XYPlot) findViewById(R.id.mySimpleXYPlot);



series2 = new SimpleXYSeries(data,

            SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, "");



BarFormatter barf = new BarFormatter(Color.parseColor("#328fde"),

            Color.parseColor("#328fde"));

BarRenderer<BarFormatter> r = new BarRenderer<BarFormatter>(plot);

r.setBarGap(0f);

r.setBarWidth(10f);

r.setBarRenderStyle(BarRenderStyle.SIDE_BY_SIDE);




// reduce the number of range labels

plot.setTicksPerRangeLabel(1);

plot.setRangeLowerBoundary(0, BoundaryMode.FIXED);

plot.getGraphWidget().setGridPadding(30, 10, 30, 0);

plot.setTicksPerDomainLabel(1);

plot.addSeries(series2, barf);
```

```
   plot.setDomainLabel("Hours");
               plot.setDomainValueFormat(new GraphXLabelFormat());
               xLabels=hours;

}


class GraphXLabelFormat extends Format {

   @Override
   public StringBuffer format(Object arg0, StringBuffer arg1, FieldPosition arg2) {
      // TODO Auto-generated method stub

      int parsedInt = Math.round(Float.parseFloat(arg0.toString()));
      Log.d("test", parsedInt + " " + arg1 + " " + arg2);
      String labelString = xLabels[parsedInt];

      arg1.append(labelString);
      Log.e("domain values", labelString);
      return arg1;
   }
```

```
        @Override

        public Object parseObject(String arg0, ParsePosition arg1) {

            // TODO Auto-generated method stub

            return java.util.Arrays.asList(xLabels).indexOf(arg0);

        }

    }





}
```

Graph Comparisons

5.5.1 Screenshots

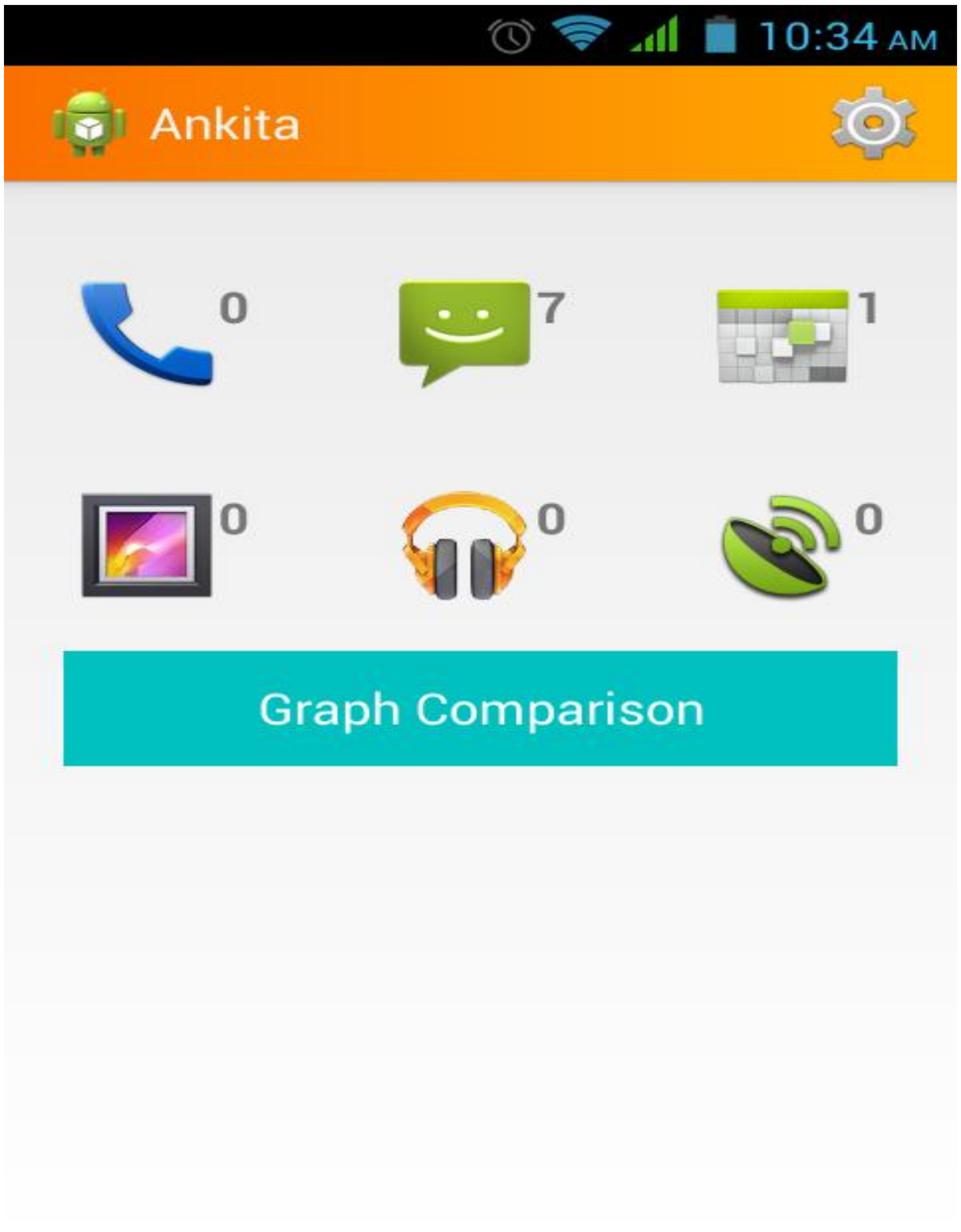
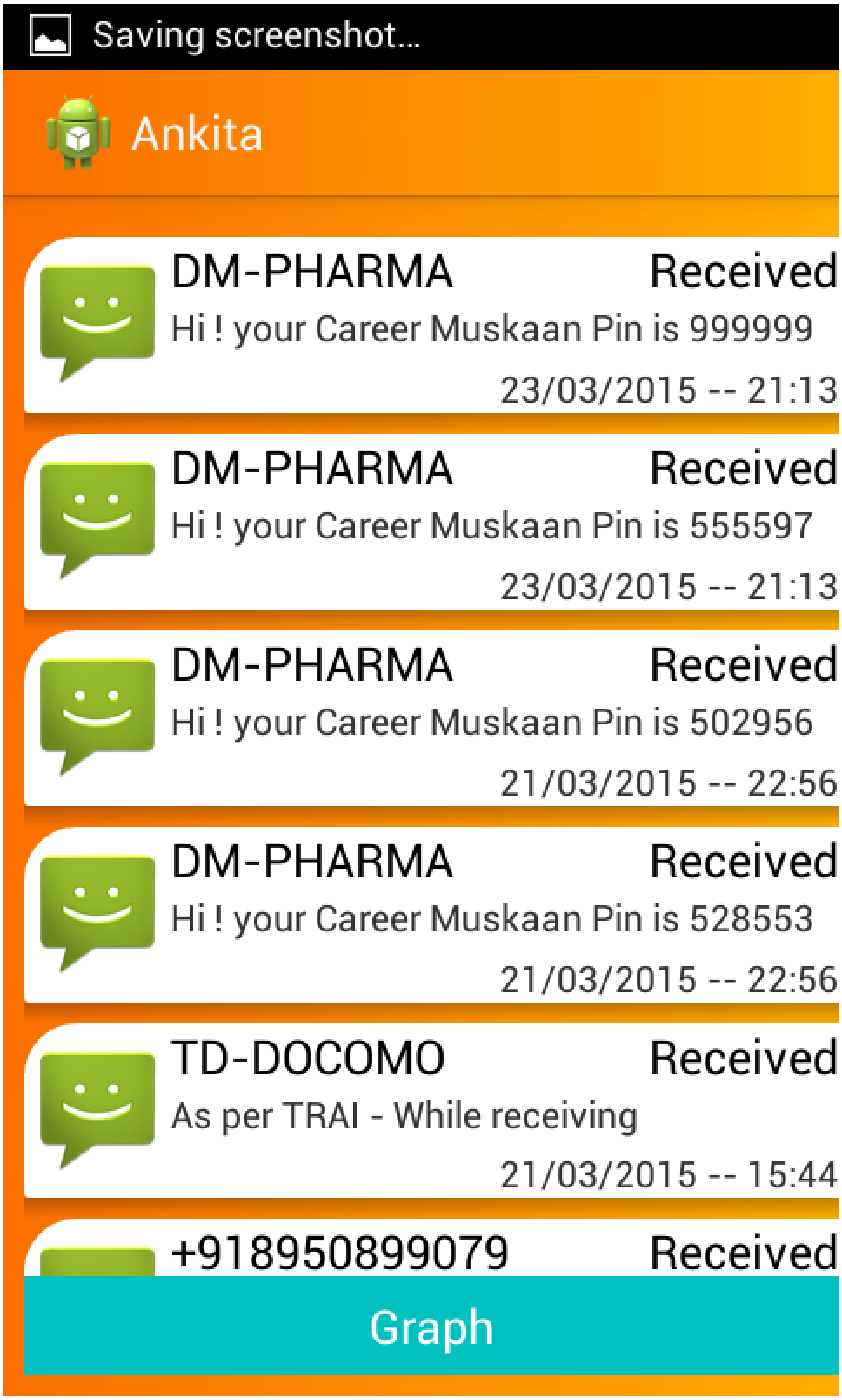


Fig 16 Figure for homepage


Saving screenshot...

Ankita

DM-PHARMA Received
Hi ! your Career Muskaan Pin is 999999
23/03/2015 -- 21:13

DM-PHARMA Received
Hi ! your Career Muskaan Pin is 555597
23/03/2015 -- 21:13

DM-PHARMA Received
Hi ! your Career Muskaan Pin is 502956
21/03/2015 -- 22:56

DM-PHARMA Received
Hi ! your Career Muskaan Pin is 528553
21/03/2015 -- 22:56

TD-DOCOMO Received
As per TRAI - While receiving
21/03/2015 -- 15:44

+918950899079 Received

Fig17 text messages data

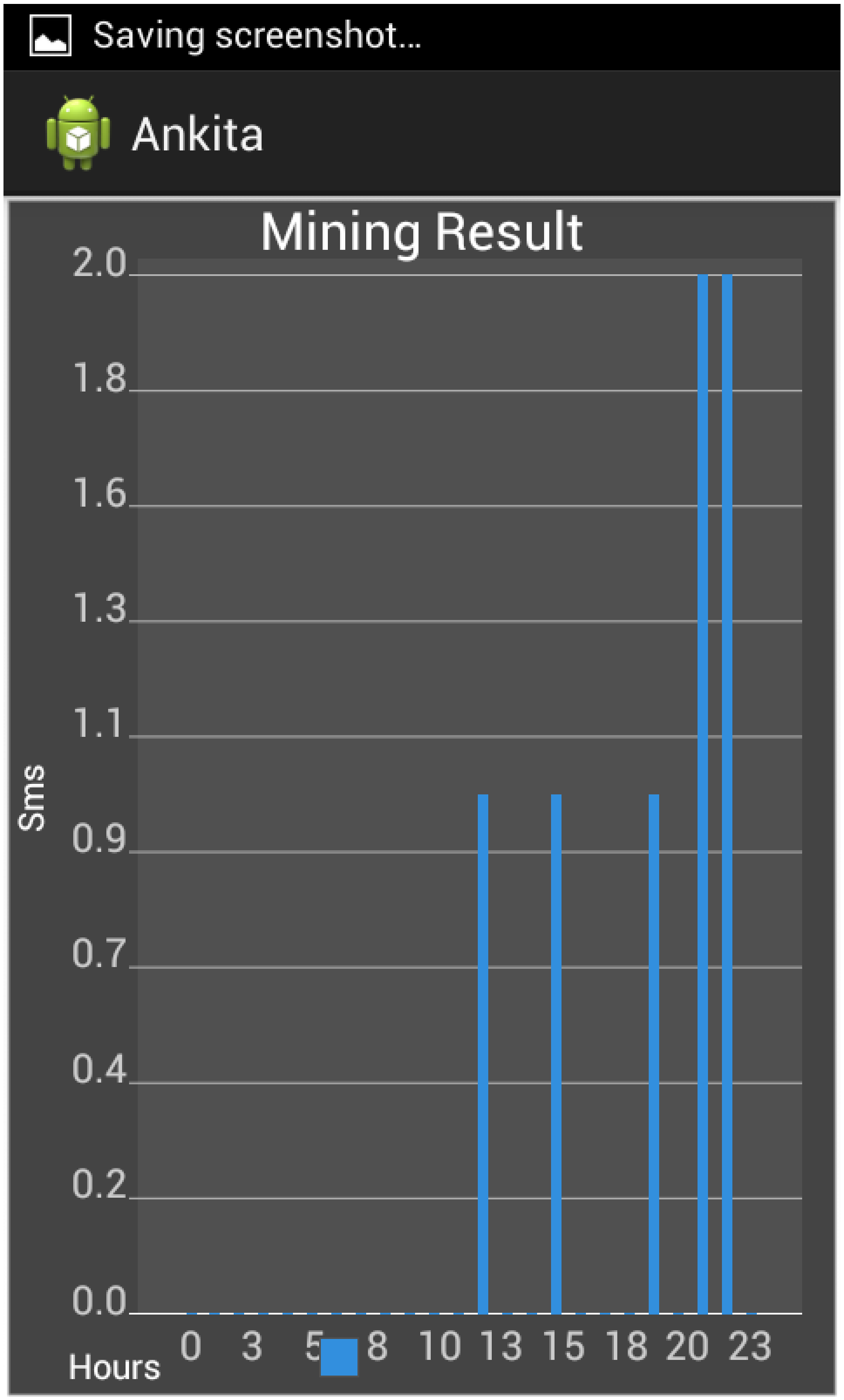


Fig18 graph comparisons for sms

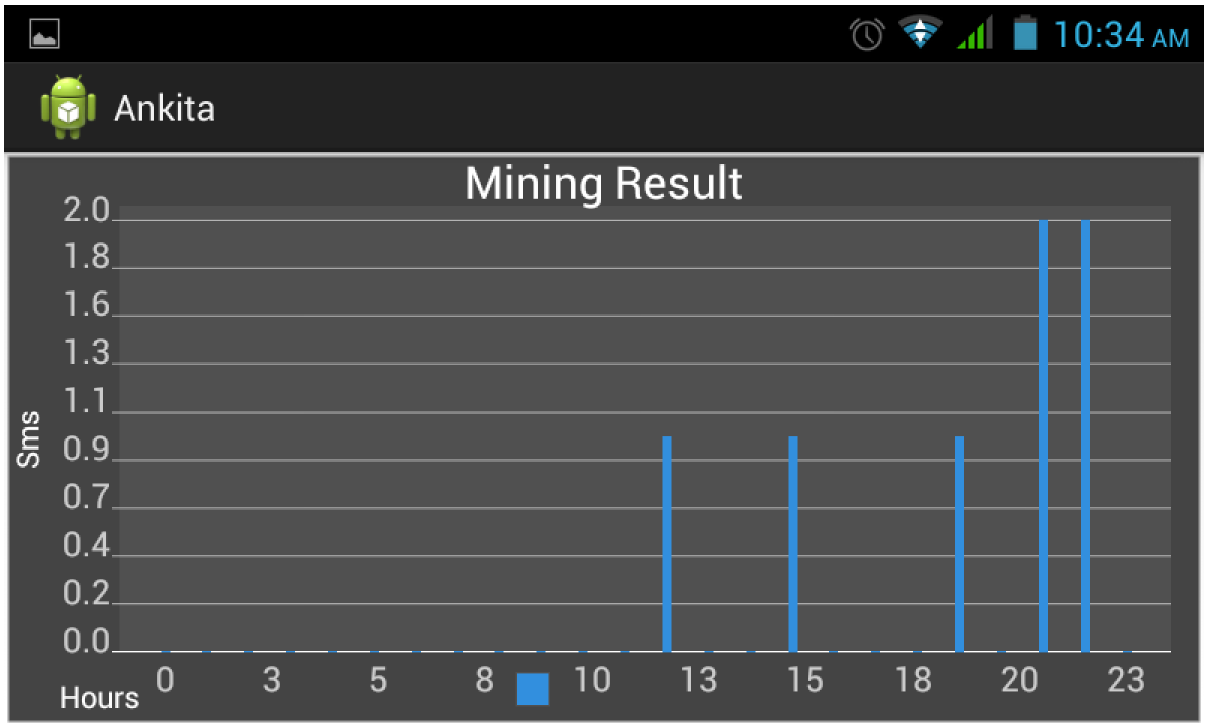


Fig19 graph comparisons for sms1

Fig20 data mining

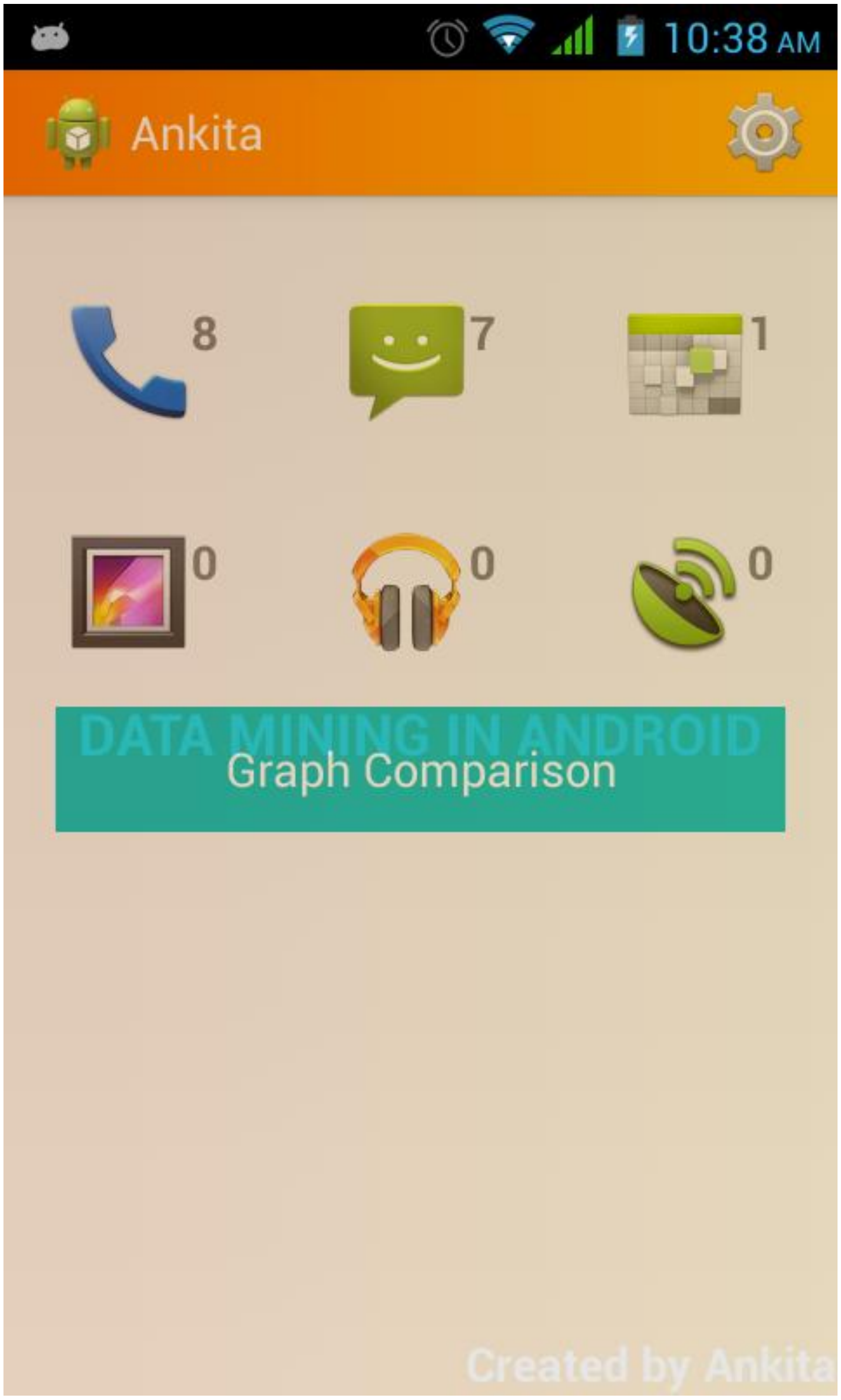


Fig20 data mining

Fig21 data mining1

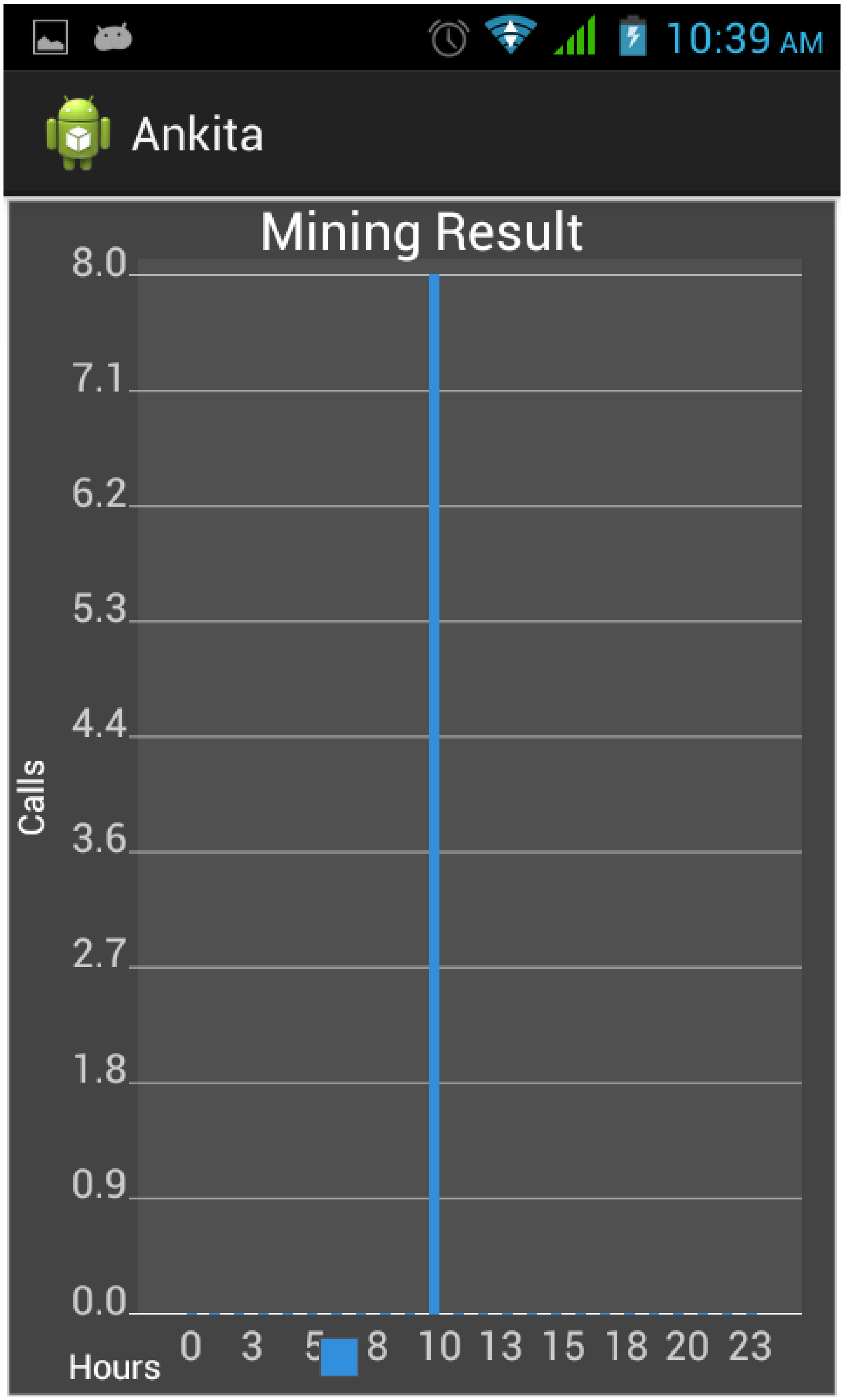


Fig22 data mining results for call logs

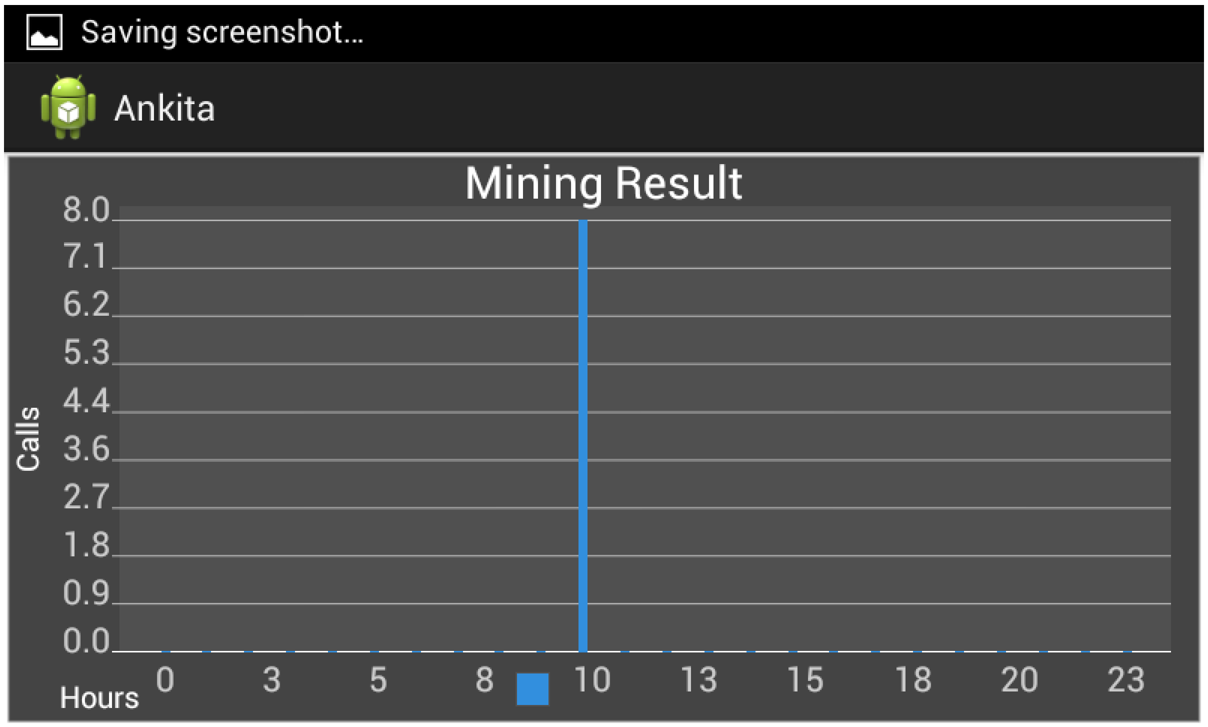


Fig23 data mining results for call logs1

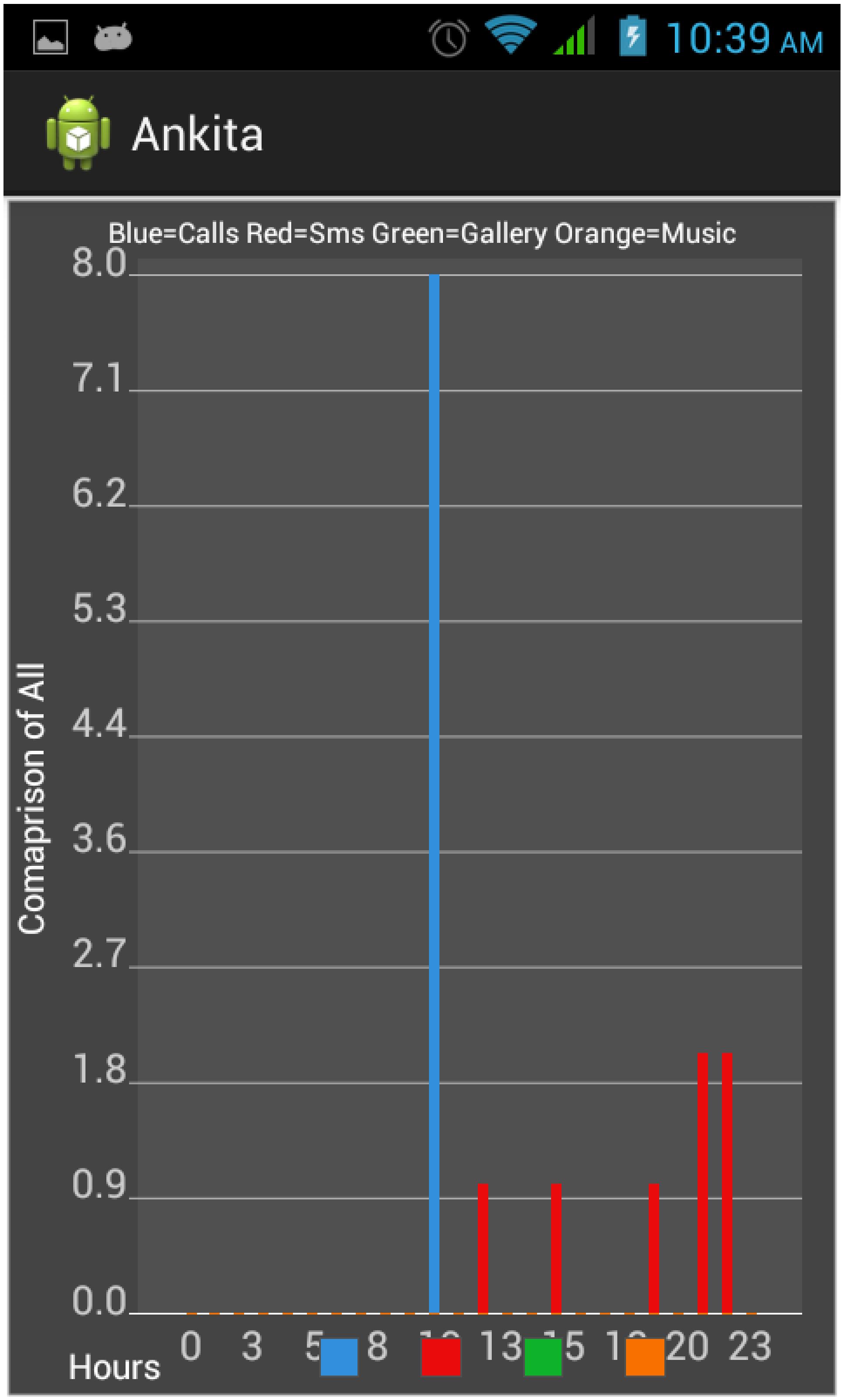


Fig24 graph comparisons for all data

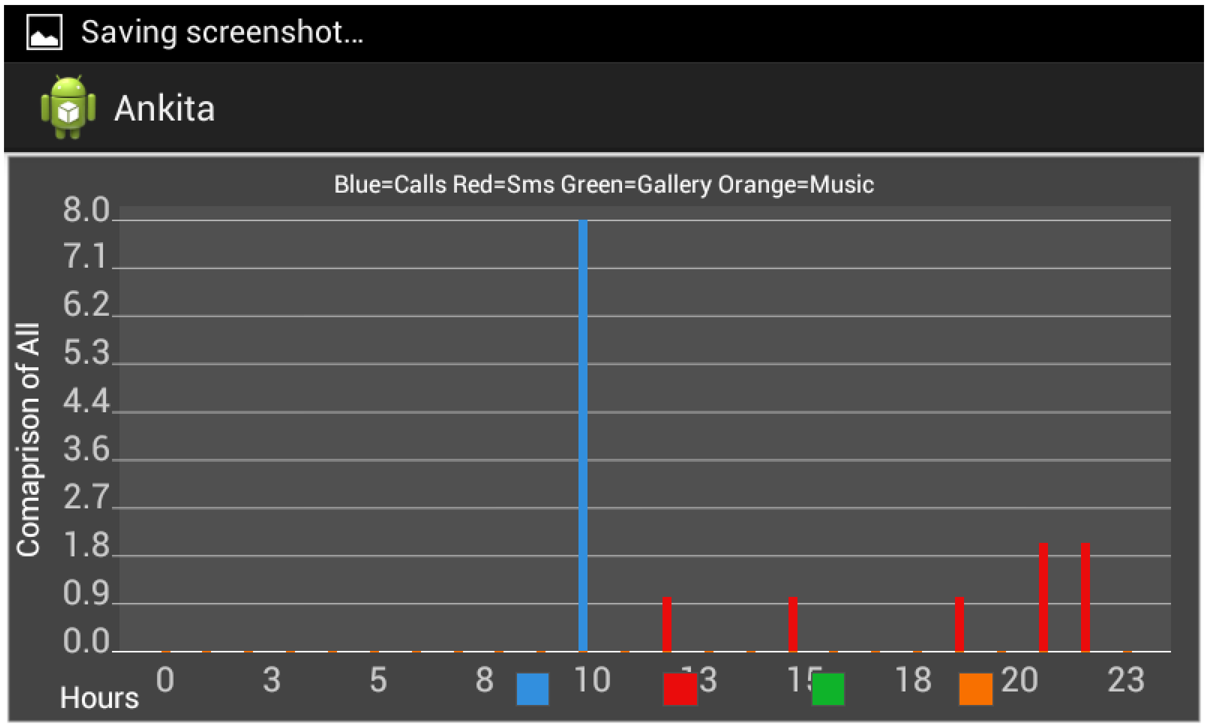


Fig25 graph comparisons for all data1

FUTURE SCOPE

The application is based on android .So, it can be used in any mobile containing android OS .In future all the data of different application users can be analyzed to mine useful information which can be helpful in conducting surveys by different organizations . Also new features can be implemented like whatsapp messages can be stored and writing pattern/style can be mined.

USER MANUAL

1. Install the JDK
2. Download and install the Eclipse for developing android application
3. Download and Install the android SDK
4. Install the ADT plugin for eclipse
5. Configure the ADT plugin
6. Create the AVD
7. Import the project
8. Copy the path from sdk i.e C:\eclipse\sdk\extras\google\google_play_services to include in the library options
9. Right click on the project go to properties go to android and add google play service library to it
10. Run the emulator

REFERENCES

- http://www.grokkingandroid.com/androids-calendarcontract-provider/
- http://developer.xamarin.com/guides/android/user_interface/calendar/
- http://stackoverflow.com/questions/848728/how-can-i-read-sms-messages-from-the-inbox-programmatically-in-android
- http://androidexample.com/Incomming_SMS_Broadcast_Receiver_-_Android_Example/index.php?view=article_discription&aid=62&aaid=87
- http://www.vogella.com/tutorials/AndroidLocationAPI/article.html
- http://www.androidhive.info/2012/07/android-gps-location-manager-tutorial/
- http://www.javatpoint.com/
- http://www.androidhive.info/2012/03/android-building-audio-player-tutorial/
- http://code.tutsplus.com/tutorials/create-a-music-player-on-android-user-controls--mobile-22787
- http://androidlabs.org/short-experiments/broadcast-receivers/do-something-when-the-phone-rings/
- https://looksok.wordpress.com/2013/04/13/android-broadcastreceiver-tutorial-detect-outgoing-phone-call-event/
- http://stackoverflow.com/questions/4571461/broadcast-receiver-wont-receive-camera-event