**CLOUD BASED E-SALVATION FOR WOMEN SAFETY**

**BY**

AVULA ROHITHA 17BLC1007

GUDETI BHAVYA 17BLC1023

ISHAN GOGNA 17BLC1148

A project report submitted to

**Dr. SOFANA REKA S**

**SCHOOL OF ELECTRONICS ENGINEERING**

in partial fulfilment of the requirements for the course of

**CLOUD AND DISTRIBUTED COMPUTING ECM3002**



**Vellore Institute of Technology (VIT) CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**June 2020**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “**CLOUD BASED E-SALVATION FOR WOMEN SAFETY”** is a bonafide work of **AVULA ROHITHA (17BLC1007), GUDETI BHAVYA (17BLC1023)** and **ISHAN GOGNA (17BLC1148)** who carried out the Project work under my supervision and guidance.

**Dr. SOFANA REKA S**

Assistant Professor (Sr.)

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

**ABSTRACT**

Many unfortunate incidents have been taking place in woman’s case. Problems may come from any direction such as women walking on the road after the work, going to super market or many other reasons for which they go alone. People at home are not sure of their return safely.

Women safety, in India is a big concern and a very important social justice issue since the very beginning. It’s an unfortunate truth that women, especially if they are by themselves, face threats on the road that men generally don't have to worry about. There have been numerous instances of women being followed by vehicular stalkers, and then attacked. We are designing a one -stop solution which will act as a virtual helper for woman. It is a social welfare application which is being created on Android platform especially for women from where their closed ones will be able to know about their current location through live tracking i.e. where they are and if they need any help in any situation, they can activate the application for their safety. This Android and Cloud based application would provide many important features like, women can add their stories and thoughts and can read others’ stories and thoughts too; they can follow other women and many more features. It also provides a feature to get involved with local authorities for improving the condition of unsafe roads and localities. The goal is of course a city in which women can travel in every train compartment they want or walk the streets at any time of the day without having to check their phones for safe travel routes first. Until then, technology will help keep them safe from harm.

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr.Sofana Reka S,** Assistant Professor(Sr.), School of Electronics Engineering , for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian A** Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Programme Chair**, Dr.Thiripurasundari D** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **SERIAL NO.** |  | **NAME** | **PAGE NO.** |
|  |  | ABSTRACT | 3 |
|  |  | ACKNOWLEDGEMENT | 4 |
|  |  |  |  |
| 1 |  | INTRODUCTION | 7-8 |
|  |  |  |  |
|  | 1.1 | OBJECTIVES AND GOALS | 7 |
|  | 1.2 | STATEMENTOF PROBLEM | 7 |
|  | 1.3 | MOTIVATION | 7-8 |
|  |  |  |  |
| 2 |  | LITERATURE SURVEY | 8-9 |
|  | 2.1 | EXISTING SYSTEM | 8-9 |
|  | 2.2 | PROPOSED SYSTEM | 9 |
| 3 |  | BASIC CONCEPTS AND TOOLS | 9-18 |
|  | 3.1 | INTRODUCTION TO JAVA | 9-11 |
|  | 3.2  3.3  3.4 | ANDROID  ECLIPSE  CODELAB ARCHITECTURE | 11-16  16-17  17 |
| 4 |  | DESIGN METHODOLOGY | 17-18 |
|  | 4.1 | STATE DIAGRAM | 17 |
|  | 4.2 | SEQUENCE DIAGRAM | 18 |

5 IMPLEMENTATION 18-25

5.1 STEPS TO DESIGN 18-25

6 TESTING AND VALIDATION 25-70

6.1 TESTING FUNDAMENTALS 25-26

6.2 CODE 26-61

6.3 SCREENSHOTS IN ANDROID STUDIO 61-66

6.4 RESULTS 66-70

7 CONCLUSION AND SCOPE

FOR FUTUREWORK 71-72

7.1 CONCLUSION 71

7.2 FUTURE WORK 71

7.3 REFERENCES 71-72

**CHAPTER 1**

**INTRODUCTION AND STATEMENT OF THE PROBLEM**

* 1. **OBJECTIVE**

This report explains the details of development process for the **Android Safety Application Using GPS and Messaging System** software package.

The main feature of this application is to provide location tracking functionality to Android devices using SMS. This application locates a device by making a call to the device and gets its location in the form of the latitude and longitude of that Android device. The application also has the capability of authentication to allow the sender to share the location information with the receiver of SMS.

The application features a personal safety alarm that sends an emergency message to the user’s chosen contacts with the push of single button. In this application all ‘guardians’ receive a text message with a link to a map showing the user’s location via GPS. It gives the user two additional levels of safety: A risk mode with real time GPS tracking of the user’s positions and a timer mode with automatic alarm activation.

This code lab is an introduction to **Google Cloud Endpoints**, the technology that enables the publication of RESTful APIs which can easily be consumed by Android applications. We will start from an **existing mobile Android application** and connect it to a brand-new **backend powered by Cloud Endpoints.** All development are done with the [Android Studio IDE](https://developer.android.com/sdk/installing/studio.html).

* 1. **STATEMENT OF THE PROBLEM**

People talk much about safety; but are unable to do anything to prevent unsafe incidents. Citizens fight for human rights on the television and other public forums, but are often unable to take any concrete actions to help the general public. Here we have tried to develop an Android application to help the users enhance their security with the help of their mobile device.

* 1. **MOTIVATION**

As a result of technological progress, we are facing an incredible variety of possibilities to communicate regardless of the distance. Smart phones provide a great choice of features that facilitate the life for the users as well as make it more comfortable.

Every day the features and capabilities of handheld mobile phones are increasing at a surprising rate. For this reason we wanted to create an Android application which gives us the opportunity to improve our knowledge of Mobile application development. We find this project a great opportunity to combine many technologies and languages in the same software system, and learn how to work as a team in projects. An important motivation for us was the challenge to solve all the problems that arise while the project is developed. In addition, we wanted to make both the user experience comfortable and the program portable, reliable, secure, stable and intuitive.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 EXISTING SYSTEMS**

**Ringer**

A silent phone can be extremely tricky to find. People in the habit of losing a silent cell phone may wish to invest in a phone sensor, also known as a phone detector. These are tools that, when placed near a cell phone, will actually pick up the call signal and make sounds to indicate that the phone is somewhere within proximity. If the phone is lost, all you need to do is have someone call you as you walk around with the sensor until the device begins to indicate that a call signal is nearby. When you hear the signal, you then have a basic idea of where to start looking for your cell phone.

**Phone Tracking Using IMEI Number**

Every phone comes with a unique International Mobile Equipment Identify Number which can come in useful to track it in case of loss or theft. This number can be accessed by dialing \*#06# and it is advisable to make a note of it as soon as you purchase your handset. In case the phone gets stolen, one is advised to file an FIR with the police and give them its identity number.

A copy of the FIR and IMEI number can be passed on to the service provider who will then be able to track the lost handset. With its IMEI number, a device can be traced even if it is being used with another SIM or even if it is switched off. Once the handset is located, the user can request the service provider to block it from being used the user is able to get your hands on it again.

## 2.2 PROPOSED SYSTEM

**Ringer**

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

**Location Tracking**

In this proposed system the user can locate any phone that has been misplaced or stolen. Once the App is installed on the phone, it can be located by sending an SMS with predefined keyword. The system gets you current updated location.[2]

**CHAPTER 3**

**BASIC CONCEPTS AND TOOLS**

### 

### 3.1 INTRODUCTION TO JAVA

Java is a programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The first publicly available version of Java (Java 1.0) was released in 1995. Sun Microsystems was acquired by the Oracle Corporation in 2010. Over time new enhanced versions of Java have been released. The current version of Java is Java 1.7 which is also known as Java 7. From the Java programming language the Java platform evolved. The Java platform allows software developers to write program code in other languages than the Java programming language and still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries.

**Java Virtual machine**

The Java virtual machine (JVM) is a software implementation of a computer that executes programs like a real machine. The Java virtual machine is written specifically for a specific operating system, e.g. for Linux a special implementation is required as well as for Windows.

**Java Runtime Environment vs. Java Development Kit**

A Java distribution comes typically in two flavors, the Java Runtime Environment (JRE) and the Java Development Kit (JDK).The Java runtime environment (JRE) consists of the JVM and the Java class libraries and contains the necessary functionality to start Java programs. The JDK contains in addition the development tools necessary to create Java programs. The JDK consists therefore of a Java compiler, the Java virtual machine, and the Java class libraries.[3

**Characteristics of Java**

The target of Java is to write a program once and then run this program on multiple operating systems.

Java has the following properties:

**Platform independent:** Java programs use the Java virtual machine as abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program (which is standard complaint and follows certain rules) can run unmodified on all supported platforms, e.g. Windows or Linux.

**Object-orientated programming language:** Except the primitive data types, all elements in Java are objects.

**Strongly-typed programming language:** Java is strongly-typed, e.g. the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g. must be done in most cases by the programmer.

**Interpreted and compiled language:** Java source code is transferred into the byte code format which does not depend on the target platform. These byte code instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so-called Hotspot-Compiler which translates performance critical byte code instructions into native code instructions.

**Automatic memory management:** Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector deletes automatically objects to which no active pointer exists.

**3.2 ANDROID**

Android is a software platform and operating system for mobile devices. Android is available as open source. It allows developers to write managed code in the Java language, controlling the device via Google-developed Java libraries.

Android SDK was released by Open Handset Alliance in the month of November of the year 2007. Android was actually developed using the kernel of Linux 2.6 and the highlighting features of Android include the following:

• No fees for licensing, distribution and release approval

• GSM, 3G EDGE networks for telephony

• IPC message passing

• Background processes and applications

• Shared data stores

• Complete multimedia hardware control

• API’s for location based services such as GPS.[5]

**3.2.1 Architecture of Android OS**

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



**Figure 3.1: Architecture of Android OS**

**Applications Layer**

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

**Application Framework Layer**

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

**Libraries Layer**

The next layer contains the native libraries of Android. These shared libraries are all written in C or C++, compiled for the particular hardware architecture used by the phone and preinstalled by the phone vendor.

**Android Runtime Layer**

Android Runtime layer includes Dalvik Virtual Machine (DVM) and a set of core java libraries. Every Android app gets its own instance of DVM. Dalvik has been written so that a device can run multiple virtual machines efficiently and it executes files with .dex (Dalvik Executable Format) extension optimized for minimum memory.

**3.2.2 Components of Android**

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

**Figure 3.2 Android Components**

**Activity**

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

**Broadcast Receiver**

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

**Service**

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

**Content Provider**

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

**3.2.3 Location based Services in Android**

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

1. GPS (Global Positioning System)

2. Network Service Location

**GPS (Global Positioning System)**

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

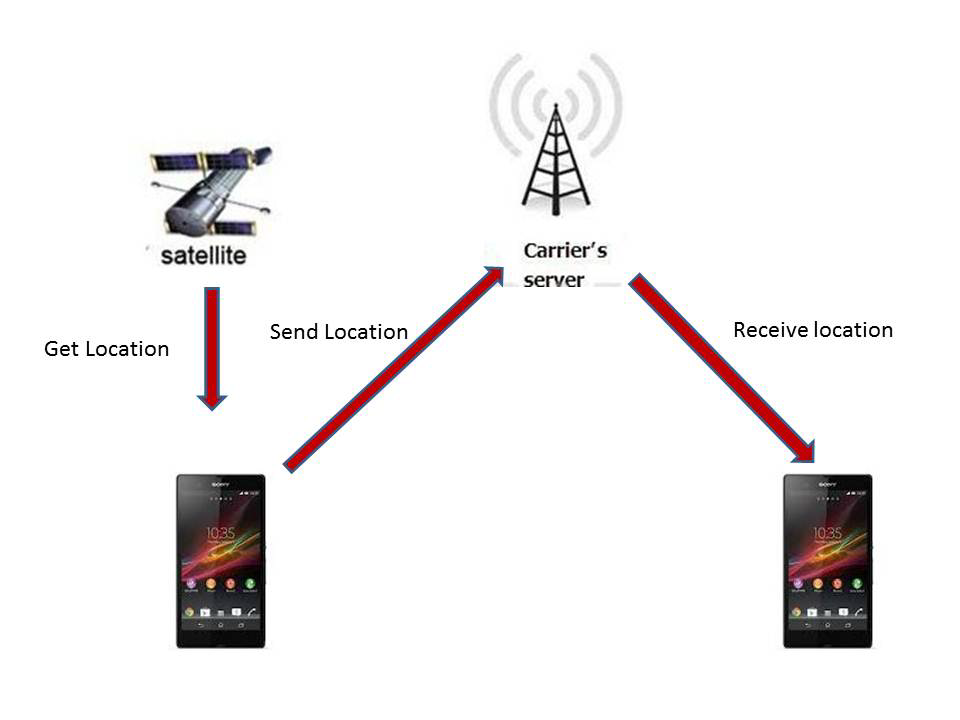


**Figure 3.3 Architecture of a GPS System**

**Network Service Location**

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

To get access to Network Provider android we request using following statement **LocationManager.NETWORK\_PROVIDER.**



**Figure 3.4 Showing network services**

**3.3 Eclipse**

**Eclipse** is a multi-language [software development environment](http://en.wikipedia.org/wiki/Software_development_environment) comprising an [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) and an extensible [plug-in](http://en.wikipedia.org/wiki/Plug-in_%28computing%29) system. It can be used to develop applications in Advanced Java and, by means of various plug-ins with other [programming languages](http://en.wikipedia.org/wiki/Programming_language) like C and C++. Its features are:

* It is Open Source.
* It is strongly recommended by Android developers.
* It is directly linked with the compiler, so we don’t need to compile the program.
* It has a good User Interface.

**3.4 CODELAB ARCHITECTURE:**

We'll be building an [App Engine](https://developers.google.com/appengine) application which will store the user tasks in Google's [Cloud Datastore](https://developers.google.com/datastore/) and communicate with the Android application via Endpoints. This offers optimal flexibility for defining your own endpoints, implementing business logic and interacting with the rest of the Google Cloud Platform services.

Using this new backend will require making small changes to the current Android application.

To make to calls to our RESTful endpoints, we can rely on generated client libraries which will help us make calls from the Android application as if they we local. No JSON parsing or HTTP-level coding required!

**CHAPTER 4**

**DESIGN METHODOLOGY**

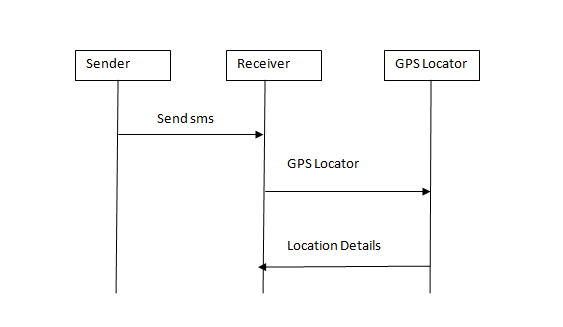
### 4.1 STATE MACHINE DIAGRAM

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

### 4.2 SEQUENCE DIAGRAM

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

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

****

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 STEPS TO DESIGN**

After designing the new system, the whole system is required to be converted into a language understood by the computer. This is accomplished by coding. Coding is an important stage where the defined procedures are transformed into control specifications with the help of a computer language. This is also called the programming phase in which the programmer converts the program specifications into computer instructions, which we refer to as programs. The programs coordinate the data movements and control the entire process in a system.

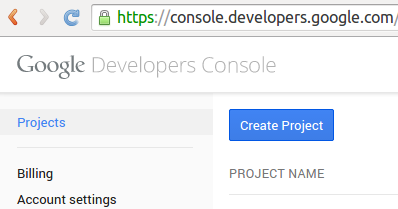
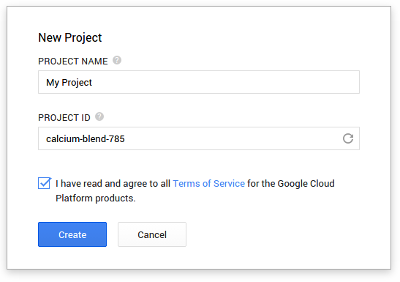
It is generally felt that programs must be modular in nature. This helps in fast development, maintenance and future change, if required.

The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing the user to take over its operation for use and evaluation.

Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operates the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

**Step 1- Create your own project in the Google Developers Console**

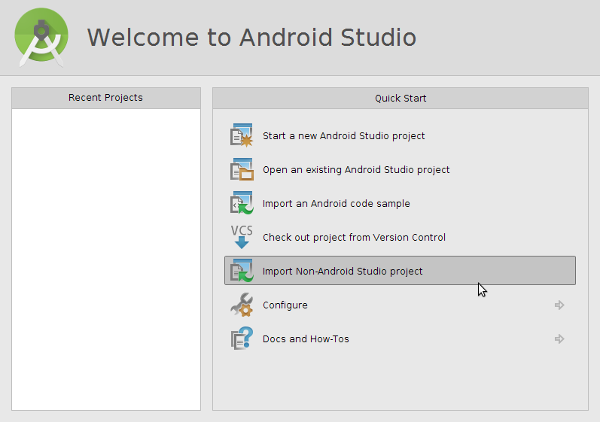
Point your browser to the [Google Developers Console](https://console.developers.google.com/) and create a new project (login first if required):

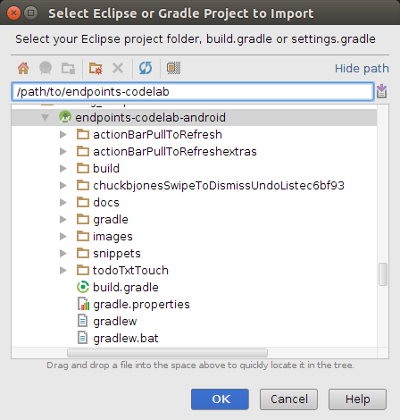
[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/DeveloperConsoleNewProject_1.png) [](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/DeveloperConsoleNewProject_3.png)

Use the suggested **Project ID** (it is unique and available) or create our own. We'll need this to deploy the backend after it's created using Android Studio. Once created, this Google Cloud Platform project will enable to manage all the services the Cloud Platform has to offer. In this lab, we'll use App Engine and Cloud Datastore and we'll create a **new backend project from scratch**.

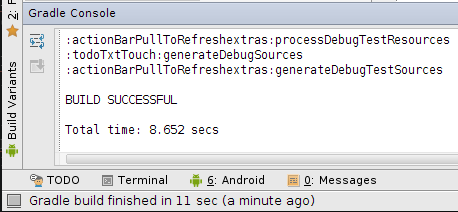
Open Android Studio and Import Non-Android Studio project (as shown in the screenshot below) the code as a new project.

Remember, this needs to use **Java 7**. Check the JDK Location in your **File** > **Project Structure**.

[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/ImportProjectAndroidStudio.png)

[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/ImportProjectAndroidStudio_2.png)

This should trigger a successful (Gradle) build:

[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/InitialBuild.png)

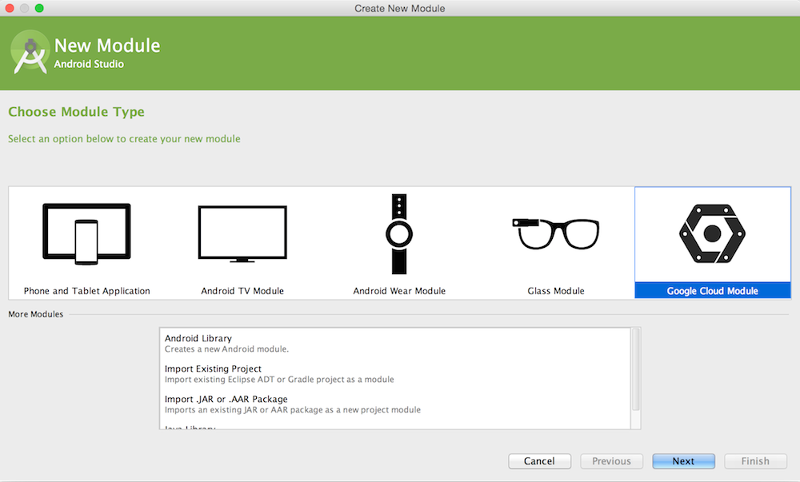
* If the build fails, make sure you have the **Support Repository** installed
* If you are asked to setup an Android SDK ("Project SDK is not defined"), simply click on the "Setup SDK" link and select the SDK without configuring it. This codelab has been tested with API level 19.

From here we'll create, build and test the backend before we make our way back to the Android client to hook it up to this new backend.

**Step 2 - Create the Endpoints backend project**

Android Studio has built-in support for Google Cloud Endpoints (as part of the IDE's support for [App Engine backend templates](https://github.com/GoogleCloudPlatform/gradle-appengine-templates))

To add a backend to the current Android project, simply go to: **File > New Module** or right-click on your "endpoints-codelab-project" in the Project side pane and choose **New > Module** and choose "Google Cloud Module".

[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/NewModule.png)

**IMPORTANT**: Make sure you select the "App Engine Java **Endpoints** Module" template

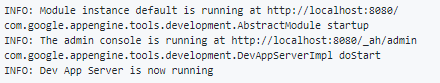
This creates a new backend Gradle module for our Android Studio project (using the [Gradle App Engine Plugin](https://github.com/GoogleCloudPlatform/gradle-appengine-plugin)) and adds it as an additional dependency in the settings.gradle file. It also adds the required dependencies for the generated client libraries to the app's build.gradle file. The generated code is pretty straight-forward with an object model for the data manipulated by the Endpoint, and the actual Endpoint implementation:

The convention here is that sending data to the Endpoint implies that the server state will change and thus only POST requests will be mapped to this method. All Cloud Endpoints annotations and attributes are documented here: <https://developers.google.com/appengine/docs/java/endpoints/annotations>

* Parameter types and return values are documented here: <https://developers.google.com/appengine/docs/java/endpoints/paramreturn_types>

Start the development app server (a full App Engine **local** environment) by pressing the green "Play" button after selecting  the current module:

Once the development app server has started we should see this log message:



See the [Appendix - Using the APIs Explorer](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/#user-content-appendix---using-the-apis-explorer) at the end of this document for further details.

Stop the local development app server (use the red square in the log window or **Run > Stop**).

At this point we have the basic Endpoints infrastructure in place but we still need to store tasks in a persistent Datastore and wire up the Android application to use this new backend.

**Step 3 - Modify the Android application to use the new backend. Run. Test. Repeat.**

While exposing standard RESTful interfaces makes it possible to access them from an Android application using HTTP/JSON calls, the Cloud Endpoints technology is able to **create client libraries** (in our case a Java library) to enable a much easier implementation on the client-side with high-level abstractions such as Java classes rather than the underlying JSON and HTTP concepts.

To generate these client libraries, we can re-build the backend by navigating to **Build > Make Module 'womensafetyTxtBackend'**.

To start calling these libraries, let's first check that when creating the backend Android Studio properly added the appropriate dependency to the Android client module.

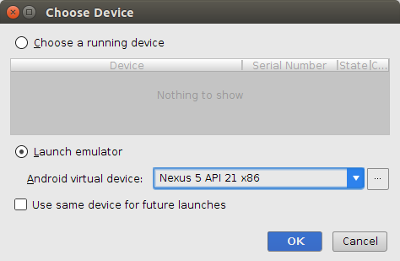
Our Android client application is now ready to start using the Endpoints client library to store tasks in our Google Cloud backend!

The pushToRemote and pullToRemote methods implement the actual communication with the backend which we want to set to use our new Cloud Endpoints-powered backend.

This constructor above sets up a development environment by initializing the TaskApi instance to enable testing with the local development app server and the local Android emulator.

The **very last step** is to switch the Android client to use this new subclass implementation.

* First, make sure the backend is running: and run the server.
* Second, select the  module and press run. If we don't have the emulator running, this should bring up this dialog:

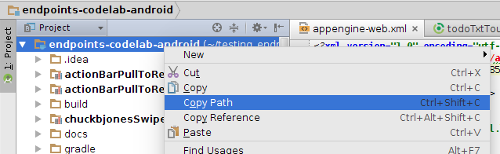
[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/ChooseDevice.png)

Click OK and wait for the emulator to start.

Once started, we can start using the application

Now, navigate to the build.gradle file at the root of the  module and make sure the appengine section contains this authorization attribute which will allow Gradle to propagate the Google Cloud OAuth2 credentials to deploy the backend to Google App Engine:

Now **open a terminal window** and set the directory to the root of the project (select the top node and **Edit > Copy Path**).

[](https://github.com/GoogleCloudPlatform/endpoints-codelab-android/blob/master/images/CopyPath.png)

On the command line type the command : ./gradlew WomensafetyTxtBackend:appengineUpdate.

Once the backend is successfully deployed, we can go back to the [Developers Console](https://console.developers.google.com/) (console.developers.google.com) and check the dashboard, current application version, logs, etc.

Rebuilding the Android application will pick up the changes made to the backend so you don't need to explicitly re-install the Endpoints client libraries.

Run the Android client one more time

**5.2 Modules**

**1. Register three numbers:** Firstly, the user has to register the three numbers as per their choice.

**2. Send predefined text:** Then after clicking on send option a message can send on the four registered numbers.

**3. Get latitude and longitude of device and create a Google map link:** Send device location to the sender of SMS.

**4. Exit Application**

The Broadcast receiver then alerts the application when each new SMS arrives.

**5.2.1** **Implementation of the modules**

1. **Get Location and Acknowledge user.**

Example received predefined text is “I am in trouble. My location is……...”.

In this module we provide the functionality of getting location details of the device.

**Step 1:** Start

**Step 2:** Check that the internet is available.

**Step 3:** If the internet is available then get location details from Network Provider.

**Step 4:** If internet is not available, then check if GPS is turned on.

**Step 5:** If GPS is available then get location details.

**Step 6:** Send location information to the receiver.

**Step 7:** End

**CHAPTER 6**

**TESTING AND VALIDATION**

**6.1 TESTING FUNDAMENTALS**

The Android testing framework, an integral part of the development environment, provides architecture and powerful tools that help us test every aspect of our application at every level from unit to framework.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

The SDK tools for building the code and testing it are available in Eclipse with ADT, and also in command-line form for use with other IDEs. These tools get information from the project of the application under test and use this information to automatically create the build files, manifest file and the directory structure for the test package.

**Procedure of running the application on a system using two virtual machines:**

1. First, we create and install two android virtual machines.
2. Then we interface them with each other using their respective numbers.
3. The application which we have running will be automatically saved into the virtual machine.
4. Then when we open the application, it will first ask for the GPS connection and the mobile data connection with an appropriate message. It will then direct the option to the settings for checking the connection.
5. Now we have to fill the numbers of the people to whom we want to send message by selecting

“Enter/change the numbers”. We can also enter the number of virtual machine of the other device.

1. After filling the numbers, we are ready to send the message to the selected numbers.

**How to run the application on an android phone**

1. After running the application on the system once, an .apk file is generated automatically in the workplace.
2. We now copy the .apk file from /bin on the computer to the phone and install it.
3. After installing we run the application in the same way as we run it on a virtual machine.

**6.2 CODE:**

1. **GPSTracker.java**

package com.example.rohitha;

import android.app.AlertDialog;

import android.app.Service;

import android.content.Context;

import android.content.DialogInterface;

import android.content.Intent;

import android.location.Location;

import android.location.LocationListener;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.IBinder;

import android.provider.Settings;

import android.util.Log;

public class GPSTracker extends Service implements LocationListener {

private final Context mContext;

// flag for GPS status

boolean isGPSEnabled = false;

// flag for network status

boolean isNetworkEnabled = false;

// flag for GPS status

boolean canGetLocation = false;

Location location; // location

double latitude; // latitude

double longitude; // longitude

// The minimum distance to change Updates in meters

private static final long MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES = 10; // 10 meters

// The minimum time between updates in milliseconds

private static final long MIN\_TIME\_BW\_UPDATES = 1000 \* 60 \* 1; // 1 minute

// Declaring a Location Manager

protected LocationManager locationManager;

public GPSTracker(Context context) {

this.mContext = context;

getLocation();

}

public Location getLocation() {

try {

locationManager = (LocationManager) mContext

.getSystemService(LOCATION\_SERVICE);

// getting GPS status

isGPSEnabled = locationManager

.isProviderEnabled(LocationManager.GPS\_PROVIDER);

// getting network status

isNetworkEnabled = locationManager

.isProviderEnabled(LocationManager.NETWORK\_PROVIDER);

if (!isGPSEnabled && !isNetworkEnabled) {

// no network provider is enabled

} else {

this.canGetLocation = true;

if (isNetworkEnabled) {

locationManager.requestLocationUpdates(

LocationManager.NETWORK\_PROVIDER,

MIN\_TIME\_BW\_UPDATES,

MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);

Log.d("Network", "Network");

if (locationManager != null) {

location = locationManager

.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);

if (location != null) {

latitude = location.getLatitude();

longitude = location.getLongitude();

}

}

}

// if GPS Enabled get lat/long using GPS Services

if (isGPSEnabled) {

if (location == null) {

locationManager.requestLocationUpdates( LocationManager.GPS\_PROVIDER,

MIN\_TIME\_BW\_UPDATES,

MIN\_DISTANCE\_CHANGE\_FOR\_UPDATES, this);

Log.d("GPS Enabled", "GPS Enabled");

if (locationManager != null) {

location = locationManager

.getLastKnownLocation(LocationManager.GPS\_PROVIDER);

if (location != null) {

latitude = location.getLatitude();

longitude = location.getLongitude();

}

}

}

}

}

} catch (Exception e) {

e.printStackTrace();

}

return location;

}

/\*\*

\* Stop using GPS listener

\* Calling this function will stop using GPS in your app

\* \*/

public void stopUsingGPS(){

if(locationManager != null){

locationManager.removeUpdates(GPSTracker.this);

}

}

/\*\*

\* Function to get latitude

\* \*/

public double getLatitude(){

if(location != null){

latitude = location.getLatitude();

}

// return latitude

return latitude;

}

/\*\*

\* Function to get longitude

\* \*/

public double getLongitude(){

if(location != null){

longitude = location.getLongitude();

}

// return longitude

return longitude;

}

/\*\*

\* Function to check GPS/wifi enabled

\* @return boolean

\* \*/

public boolean canGetLocation() {

return this.canGetLocation;

}

/\*\*

\* Function to show settings alert dialog

\* On pressing Settings button will lauch Settings Options

\* \*/

public void showSettingsAlert(){

AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);

// Setting Dialog Title

alertDialog.setTitle("GPS is settings");

// Setting Dialog Message

alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?");

// On pressing Settings button

alertDialog.setPositiveButton("Settings", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog,int which) {

Intent intent = new Intent(Settings.ACTION\_LOCATION\_SOURCE\_SETTINGS);

mContext.startActivity(intent);

}

});

// on pressing cancel button

alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

dialog.cancel();

}

});

// Showing Alert Message

alertDialog.show();

}

@Override

public void onLocationChanged(Location location) {

}

@Override

public void onProviderDisabled(String provider) {

}

@Override

public void onProviderEnabled(String provider) {

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras) {

}

@Override

public IBinder onBind(Intent arg0) {

return null;

}

}

**2)Instructions.java**

package com.example.rohitha;

import android.os.Bundle;

import android.view.View;

import android.app.Activity;

import android.content.Intent;

public class Instructions extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_instructions);

}

public void back(View v) {

Intent i\_back=new Intent(Instructions.this,MainActivity.class);

startActivity(i\_back);

}

}

**3)MainActivity.java**

package com.example.rohitha;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

public class MainActivity extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

public void register(View v) {

Intent i\_register=new Intent(MainActivity.this,Register.class);

startActivity(i\_register);

}

public void display\_no(View v) {

Intent i\_view=new Intent(MainActivity.this,Display.class);

startActivity(i\_view);

}

public void instruct(View v) {

Intent i\_help=new Intent(MainActivity.this,Instructions.class);

startActivity(i\_help);

}

public void verify(View v) {

Intent i\_verify=new Intent(MainActivity.this,Verify.class);

startActivity(i\_verify);

}

}

**4)Register.java**

package com.example.rohitha;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

public class Register extends Activity {

EditText name,number;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_register);

//Toast.makeText(getApplicationContext(), "Activity created",Toast.LENGTH\_LONG).show();

}

public void display(View v) {

Intent i\_view=new Intent(Register.this,Display.class);

startActivity(i\_view);

}

public void instructions(View v) {

Intent i\_help=new Intent(Register.this,Instructions.class);

startActivity(i\_help);

}

public void storeInDB(View v) {

Toast.makeText(getApplicationContext(), "save started",Toast.LENGTH\_LONG).show();

name = (EditText) this.findViewById(R.id.editText1);

number = (EditText) this.findViewById(R.id.editText2);

String str\_name=name.getText().toString();

String str\_number=number.getText().toString();

SQLiteDatabase db;

db=openOrCreateDatabase("NumDB", Context.MODE\_PRIVATE, null);

//Toast.makeText(getApplicationContext(), "db created",Toast.LENGTH\_LONG).show();

db.execSQL("CREATE TABLE IF NOT EXISTS details(name VARCHAR,number VARCHAR);");

//Toast.makeText(getApplicationContext(), "table created",Toast.LENGTH\_LONG).show();

Cursor c=db.rawQuery("SELECT \* FROM details", null);

if(c.getCount()<2)

{

db.execSQL("INSERT INTO details VALUES('"+str\_name+"','"+str\_number+"');");

Toast.makeText(getApplicationContext(), "Successfully Saved",Toast.LENGTH\_SHORT).show();

}

else {

db.execSQL("INSERT INTO details VALUES('"+str\_name+"','"+str\_number+"');");

Toast.makeText(getApplicationContext(), "Maximun Numbers limited reached. Previous numbers are replaced.",Toast.LENGTH\_SHORT).show();

}

db.close();

}

}

**5)RGeocoder.java**

package com.example.rohitha;

import java.io.IOException;

import java.util.List;

import java.util.Locale;

import android.content.Context;

import android.location.Address;

import android.location.Geocoder;

import android.os.Bundle;

import android.os.Handler;

import android.os.Message;

import android.util.Log;

public class RGeocoder {

private static final String TAG = "LocationAddress";

public void getAddressFromLocation(final double latitude, final double longitude,

final Context context, final Handler handler) {

Thread thread = new Thread() {

@Override

public void run() {

Geocoder geocoder = new Geocoder(context, Locale.getDefault());

String result = null;

try {

List<Address> addressList = geocoder.getFromLocation(latitude, longitude, 1);

if (addressList != null && addressList.size() > 0) {

Address address = addressList.get(0);

StringBuilder sb = new StringBuilder();

for (int i = 0; i < address.getMaxAddressLineIndex(); i++) {

sb.append(address.getAddressLine(i)).append("\n");

}

sb.append(address.getLocality()).append("\n");

sb.append(address.getPostalCode()).append("\n");

sb.append(address.getCountryName());

result = sb.toString();

}

}

catch (IOException e) {

Log.e(TAG, "Unable connect to Geocoder", e);

}

finally {

Message message = Message.obtain();

message.setTarget(handler);

if (result != null) {

message.what = 1;

Bundle bundle = new Bundle();

result = "Latitude: " + latitude + " Longitude: " + longitude +

"\n\nAddress:\n" + result;

bundle.putString("address", result);

message.setData(bundle);

} else {

message.what = 1;

Bundle bundle = new Bundle();

result = "Latitude: " + latitude + " Longitude: " + longitude +

"\n Unable to get address for this lat-long.";

bundle.putString("address", result);

message.setData(bundle);

}

message.sendToTarget();

}

}

};

thread.start();

}

}

**6)Verify.java:**

package com.example.rohitha;

import android.os.Bundle;

import android.app.Activity;

import android.view.Menu;

import android.view.MenuItem;

import android.view.View;

import android.widget.EditText;

import android.widget.Toast;

import android.support.v4.app.NavUtils;

import android.annotation.TargetApi;

import android.content.Context;

import android.content.Intent;

import android.database.sqlite.SQLiteDatabase;

import android.os.Build;

public class Verify extends Activity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_verify);

// Show the Up button in the action bar.

setupActionBar();

}

public void verify\_no(View v) {

EditText source\_no = (EditText) this.findViewById(R.id.editText1);

String str\_source\_no=source\_no.getText().toString();

SQLiteDatabase db;

db=openOrCreateDatabase("NumDB", Context.MODE\_PRIVATE, null);

// if(source\_no.getText()!=null){

db.execSQL("CREATE TABLE IF NOT EXISTS source(number VARCHAR);");

db.execSQL("INSERT INTO source VALUES('"+str\_source\_no+"');");

Toast.makeText(getApplicationContext(), str\_source\_no+" Successfully Saved",Toast.LENGTH\_SHORT).show();

db.close();

back(v);

// }

// else{

// Toast.makeText(getApplicationContext(), "Enter Your Number.",Toast.LENGTH\_SHORT).show();

// }

}

/\*\*

\* Set up the {@link android.app.ActionBar}, if the API is available.

\*/

@TargetApi(Build.VERSION\_CODES.HONEYCOMB)

private void setupActionBar() {

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB) {

getActionBar().setDisplayHomeAsUpEnabled(true);

}

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

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

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

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case android.R.id.home:

// This ID represents the Home or Up button. In the case of this

// activity, the Up button is shown. Use NavUtils to allow users

// to navigate up one level in the application structure. For

// more details, see the Navigation pattern on Android Design:

//

// http://developer.android.com/design/patterns/navigation.html#up-vs-back

//

NavUtils.navigateUpFromSameTask(this);

return true;

}

return super.onOptionsItemSelected(item);

}

public void back(View v) {

Intent i\_back=new Intent(Verify.this,MainActivity.class);

startActivity(i\_back);

}

}

**7)AccelrometerListener.java:**

package com.example.rohitha;

public interface AccelerometerListener {

public void onAccelerationChanged(float x, float y, float z);

public void onShake(float force);

}

**8)AccelerometerManager.java:**

package com.example.rohitha;

import java.util.List;

import android.content.Context;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.widget.Toast;

public class AccelerometerManager {

private static Context aContext=null;

/\*\* Accuracy configuration \*/

private static float threshold = 15.0f;

private static int interval = 200;

private static Sensor sensor;

private static SensorManager sensorManager;

// you could use an OrientationListener array instead

// if you plans to use more than one listener

private static AccelerometerListener listener;

/\*\* indicates whether or not Accelerometer Sensor is supported \*/

private static Boolean supported;

/\*\* indicates whether or not Accelerometer Sensor is running \*/

private static boolean running = false;

/\*\*

\* Returns true if the manager is listening to orientation changes

\*/

public static boolean isListening() {

return running;

}

/\*\*

\* Unregisters listeners

\*/

public static void stopListening() {

running = false;

try {

if (sensorManager != null && sensorEventListener != null) {

sensorManager.unregisterListener(sensorEventListener);

}

} catch (Exception e) {}

}

/\*\*

\* Returns true if at least one Accelerometer sensor is available

\*/

public static boolean isSupported(Context context) {

aContext = context;

if (supported == null) {

if (aContext != null) {

sensorManager = (SensorManager) aContext.

getSystemService(Context.SENSOR\_SERVICE);

// Get all sensors in device

List<Sensor> sensors = sensorManager.getSensorList(

Sensor.TYPE\_ACCELEROMETER);

supported = new Boolean(sensors.size() > 0);

} else {

supported = Boolean.FALSE;

}

}

return supported;

}

/\*\*

\* Configure the listener for shaking

\* @param threshold

\* minimum acceleration variation for considering shaking

\* @param interval

\* minimum interval between to shake events

\*/

public static void configure(int threshold, int interval) {

AccelerometerManager.threshold = threshold;

AccelerometerManager.interval = interval;

}

/\*\*

\* Registers a listener and start listening

\* @param accelerometerListener

\* callback for accelerometer events

\*/

public static void startListening( AccelerometerListener accelerometerListener )

{

sensorManager = (SensorManager) aContext.

getSystemService(Context.SENSOR\_SERVICE);

// Take all sensors in device

List<Sensor> sensors = sensorManager.getSensorList(

Sensor.TYPE\_ACCELEROMETER);

if (sensors.size() > 0) {

sensor = sensors.get(0);

// Register Accelerometer Listener

running = sensorManager.registerListener(

sensorEventListener, sensor,

SensorManager.SENSOR\_DELAY\_GAME);

listener = accelerometerListener;

}

}

/\*\*

\* Configures threshold and interval

\* And registers a listener and start listening

\* @param accelerometerListener

\* callback for accelerometer events

\* @param threshold

\* minimum acceleration variation for considering shaking

\* @param interval

\* minimum interval between to shake events

\*/

public static void startListening(

AccelerometerListener accelerometerListener,

int threshold, int interval) {

configure(threshold, interval);

startListening(accelerometerListener);

}

/\*\*

\* The listener that listen to events from the accelerometer listener

\*/

private static SensorEventListener sensorEventListener =

new SensorEventListener() {

private long now = 0;

private long timeDiff = 0;

private long lastUpdate = 0;

private long lastShake = 0;

private float x = 0;

private float y = 0;

private float z = 0;

private float lastX = 0;

private float lastY = 0;

private float lastZ = 0;

private float force = 0;

public void onAccuracyChanged(Sensor sensor, int accuracy) {}

public void onSensorChanged(SensorEvent event) {

// use the event timestamp as reference

// so the manager precision won't depends

// on the AccelerometerListener implementation

// processing time

now = event.timestamp;

x = event.values[0];

y = event.values[1];

z = event.values[2];

// if not interesting in shake events

// just remove the whole if then else block

if (lastUpdate == 0) {

lastUpdate = now;

lastShake = now;

lastX = x;

lastY = y;

lastZ = z;

Toast.makeText(aContext,"No Motion detected",

Toast.LENGTH\_SHORT).show();

} else {

timeDiff = now - lastUpdate;

if (timeDiff > 0) {

/\*force = Math.abs(x + y + z - lastX - lastY - lastZ)

/ timeDiff;\*/

force = Math.abs(x + y + z - lastX - lastY - lastZ);

if (Float.compare(force, threshold) >0 ) {

//Toast.makeText(Accelerometer.getContext(),

//(now-lastShake)+" >= "+interval, 1000).show();

if (now - lastShake >= interval) {

// trigger shake event

listener.onShake(force);

}

else

{

Toast.makeText(aContext,"No Motion detected.",

Toast.LENGTH\_SHORT).show();

}

lastShake = now;

}

lastX = x;

lastY = y;

lastZ = z;

lastUpdate = now;

}

else

{

Toast.makeText(aContext,"No Motion detected", Toast.LENGTH\_SHORT).show();

}

}

// trigger change event

listener.onAccelerationChanged(x, y, z);

}

};

}

**9)BgService.java:**

package com.example.rohitha;

import android.annotation.SuppressLint;

import android.app.Service;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.os.Handler;

import android.os.HandlerThread;

import android.os.IBinder;

import android.os.Looper;

import android.os.Message;

import android.telephony.SmsManager;

import android.util.Log;

import android.widget.Toast;

@SuppressLint("HandlerLeak")

public class BgService extends Service implements AccelerometerListener{

String str\_address;

private Looper mServiceLooper;

private ServiceHandler mServiceHandler;

// Handler that receives messages from the thread.

private final class ServiceHandler extends Handler {

public ServiceHandler(Looper looper) {

super(looper);

}

@Override

public void handleMessage(Message msg) {

// REPLACE THIS CODE WITH YOUR APP CODE

// Wait before Toasting Service Message

// to give the Service Started message time to display.

// Toast Service Message.

/\* Context context = getApplicationContext();

CharSequence text = "Service Message";

int duration = Toast.LENGTH\_LONG;

Toast toast = Toast.makeText(context, text, duration);

toast.show();

\*/

// Service can stop itself using the stopSelf() method.

// Not using in this app. Example statement shown below.

//stopSelf(msg.arg1);

}

}

@Override

public IBinder onBind(Intent arg0) {

return null;

}

@Override

public void onCreate() {

super.onCreate();

if (AccelerometerManager.isSupported(this)) {

AccelerometerManager.startListening(this);

}

HandlerThread thread = new HandlerThread("ServiceStartArguments",android.os.Process.THREAD\_PRIORITY\_BACKGROUND);

thread.start();

mServiceLooper = thread.getLooper();

mServiceHandler = new ServiceHandler(mServiceLooper); }

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

// Get message from message pool using handler.

Message msg = mServiceHandler.obtainMessage();

// Set start ID (unique to the specific start) in message.

msg.arg1 = startId;

// Send message to start job.

mServiceHandler.sendMessage(msg);

// Toast Service Started message.

// Context context = getApplicationContext();

/\* CharSequence text = "Service Started";

int duration = Toast.LENGTH\_SHORT;

Toast toast = Toast.makeText(context, text, duration);

toast.show();

\*/

// Start a sticky.

return START\_STICKY;

}

public class GeocoderHandler extends Handler {

@Override

public void handleMessage(Message message) {

Toast.makeText(getApplicationContext(), "geocoderhandler started", Toast.LENGTH\_SHORT).show();

switch (message.what) {

case 1:

Bundle bundle = message.getData();

str\_address = bundle.getString("address");

// TelephonyManager tmgr=(TelephonyManager)BgService.this.getSystemService(Context.TELEPHONY\_SERVICE);

// String ph\_number=tmgr.getLine1Number();

SQLiteDatabase db;

db=openOrCreateDatabase("NumDB", Context.MODE\_PRIVATE, null);

Cursor c=db.rawQuery("SELECT \* FROM details", null);

Cursor c1=db.rawQuery("SELECT \* FROM SOURCE", null);

String source\_ph\_number=c1.getString(0);

while(c.moveToNext())

{

String target\_ph\_number=c.getString(1);

// SmsManager smsManager=SmsManager.getDefault();

// smsManager.sendTextMessage("+918121668944", "+918121668944", "Please help me. I need help immediately. This is where i am now:"+str\_address, null, null);

Toast.makeText(getApplicationContext(), "Source:"+source\_ph\_number+"Target:"+target\_ph\_number, Toast.LENGTH\_SHORT).show();

}

db.close();

break;

default:

str\_address = null;

}

Toast.makeText(getApplicationContext(), str\_address, Toast.LENGTH\_SHORT).show();

}

}

@Override

public void onAccelerationChanged(float x, float y, float z) {

// TODO Auto-generated method stub

}

@Override

public void onShake(float force) {

GPSTracker gps;

gps = new GPSTracker(BgService.this);

if(gps.canGetLocation()){

double latitude = gps.getLatitude();

double longitude = gps.getLongitude();

RGeocoder RGeocoder = new RGeocoder();

RGeocoder.getAddressFromLocation(latitude, longitude,getApplicationContext(), new GeocoderHandler());

Toast.makeText(getApplicationContext(), "onShake", Toast.LENGTH\_SHORT).show();

}

else{

gps.showSettingsAlert();

}

}

// onDestroy method. Display toast that service has stopped.

@Override

public void onDestroy() {

super.onDestroy();

// Toast Service Stopped.

Context context = getApplicationContext();

Log.i("Sensor", "Service distroy");

if (AccelerometerManager.isListening()) {

AccelerometerManager.stopListening();

}

CharSequence text = "Women Safety App Service Stopped";

int duration = Toast.LENGTH\_SHORT;

Toast toast = Toast.makeText(context, text, duration);

toast.show();

}

}

**10)DisplayProblem.java:**

package com.example.rohitha;

import android.app.Activity;

import android.app.AlertDialog.Builder;

import android.content.Context;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.os.Bundle;

import android.view.View;

public class Display extends Activity{

Cursor c;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_display);

SQLiteDatabase db;

db=openOrCreateDatabase("NumDB", Context.MODE\_PRIVATE, null);

c=db.rawQuery("SELECT \* FROM details", null);

if(c.getCount()==0)

{

showMessage("Error", "No records found.");

return;

}

StringBuffer buffer=new StringBuffer();

while(c.moveToNext())

{

buffer.append("Name: "+c.getString(0)+"\n");

buffer.append("Number: "+c.getString(1)+"\n");

}

showMessage("Details", buffer.toString());

Intent i\_startservice=new Intent(Display.this,BgService.class);

startService(i\_startservice);

}

public void showMessage(String title,String message)

{

Builder builder=new Builder(this);

builder.setCancelable(true);

builder.setTitle(title);

builder.setMessage(message);

builder.show();

}

public void back(View v) {

Intent i\_back=new Intent(Display.this,MainActivity.class);

startActivity(i\_back);

}

}

**11)activity\_register.xml:**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".Register" >

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentLeft="true"

android:layout\_alignParentTop="true"

android:layout\_marginTop="64dp"

android:text="Name:" />

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/textView1"

android:layout\_below="@+id/textView1"

android:layout\_marginTop="71dp"

android:text="Number:" />

<Button

android:id="@+id/button1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/textView2"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="68dp"

android:onClick="storeInDB"

android:text="Save" />

<EditText

android:id="@+id/editText2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignBaseline="@+id/textView2"

android:layout\_alignBottom="@+id/textView2"

android:layout\_alignRight="@+id/button1"

android:ems="10"

android:hint="Mobile Number"

android:inputType="phone" />

<EditText

android:id="@+id/editText1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignBaseline="@+id/textView1"

android:layout\_alignBottom="@+id/textView1"

android:layout\_alignLeft="@+id/editText2"

android:ems="10"

android:hint="Person Name"

android:inputType="textPersonName" >

<requestFocus />

</EditText>

<Button

android:id="@+id/button2"

style="?android:attr/buttonStyleSmall"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/button1"

android:layout\_alignParentBottom="true"

android:layout\_marginBottom="21dp"

android:onClick="instructions"

android:text="Instructions" />

<Button

android:id="@+id/button3"

style="?android:attr/buttonStyleSmall"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignRight="@+id/button1"

android:layout\_alignTop="@+id/button2"

android:onClick="display"

android:text="View Registered" />

</RelativeLayout>

**12)activity\_verify.xml:**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".Verify" >

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="176dp"

android:onClick="verify\_no"

android:text="Submit" />

<EditText

android:id="@+id/editText1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_above="@+id/button1"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="76dp"

android:ems="10"

android:hint="Your Phone Number"

android:inputType="phone" >

<requestFocus />

</EditText>

<Button

android:id="@+id/button2"

style="?android:attr/buttonStyleSmall"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:layout\_marginBottom="94dp"

android:onClick="back"

android:text="Back" />

</RelativeLayout>

**13)dimens.xml:**

**a)**

<resources>

<!-- Default screen margins, per the Android Design guidelines. -->

<dimen name="activity\_horizontal\_margin">16dp</dimen>

<dimen name="activity\_vertical\_margin">16dp</dimen>

</resources>

**b)**

<resources>

<!--

Customize dimensions originally defined in res/values/dimens.xml (such as

screen margins) for sw600dp devices (e.g. 7" tablets) here.

-->

</resources>

**c)**

<resources>

<!--

Customize dimensions originally defined in res/values/dimens.xml (such as

screen margins) for sw720dp devices (e.g. 10" tablets) in landscape here.

-->

<dimen name="activity\_horizontal\_margin">128dp</dimen>

</resources>

**14)display.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

**15)instructions.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

**16)main.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

**17)register.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

**18)strings.xml:**

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

<resources>

<string name="app\_name">Women Safety App</string>

<string name="action\_settings">Settings</string>

<string name="hello\_world">Hello world!</string>

<string name="title\_activity\_register">Register</string>

<string name="title\_activity\_display">Display</string>

<string name="title\_activity\_instructions">Instructions</string>

<string name="title\_activity\_verify">Verify</string>

<string name="help">Please follow the below steps to setup your account:\n\n\n\t1:Register Your Phone Number.\n\t2:Register Your Guardian Numbers.\n\t3:Always maintain working internet connection.\n\t4:SHAKE your phone in case of emergency to alert your guardians.\n\t\n\n\n\n\tLETS START NOW!</string>

</resources>

**19)styles.xml:**

**a)**

<resources>

<!--

Base application theme for API 11+. This theme completely replaces

AppBaseTheme from res/values/styles.xml on API 11+ devices.

-->

<style name="AppBaseTheme" parent="android:Theme.Holo.Light">

<!-- API 11 theme customizations can go here. -->

</style>

</resources>

**b)**

<resources>

<!--

Base application theme for API 14+. This theme completely replaces

AppBaseTheme from BOTH res/values/styles.xml and

res/values-v11/styles.xml on API 14+ devices.

-->

<style name="AppBaseTheme" parent="android:Theme.Holo.Light.DarkActionBar">

<!-- API 14 theme customizations can go here. -->

</style>

</resources>

**c)**

<resources>

<!--

Base application theme, dependent on API level. This theme is replaced

by AppBaseTheme from res/values-vXX/styles.xml on newer devices.

-->

<style name="AppBaseTheme" parent="android:Theme.Light">

<!--

Theme customizations available in newer API levels can go in

res/values-vXX/styles.xml, while customizations related to

backward-compatibility can go here.

-->

</style>

<!-- Application theme. -->

<style name="AppTheme" parent="AppBaseTheme">

<!-- All customizations that are NOT specific to a particular API-level can go here. -->

</style>

</resources>

**20)verify.xml:**

<menu xmlns:android="http://schemas.android.com/apk/res/android" >

<item

android:id="@+id/action\_settings"

android:orderInCategory="100"

android:showAsAction="never"

android:title="@string/action\_settings"/>

</menu>

**21)activity\_display.xml:**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".Display" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:text="Shake your phone to TEST." />

<Button

android:id="@+id/button1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:onClick="back"

android:text="Main Menu" />

</RelativeLayout>

**22)activity\_instructions.xml:**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".Instructions" >

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="16dp"

android:text="INSTRUCTIONS"

android:textAppearance="?android:attr/textAppearanceLarge" />

<Button

android:id="@+id/button1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/help"

android:layout\_centerHorizontal="true"

android:layout\_marginTop="34dp"

android:onClick="back"

android:text="Main Menu" />

<TextView

android:id="@+id/help"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/button1"

android:layout\_below="@+id/textView1"

android:layout\_marginTop="15dp"

android:text="@string/help" />

</RelativeLayout>

**23)activity\_main.xml:**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:paddingBottom="@dimen/activity\_vertical\_margin"

android:paddingLeft="@dimen/activity\_horizontal\_margin"

android:paddingRight="@dimen/activity\_horizontal\_margin"

android:paddingTop="@dimen/activity\_vertical\_margin"

tools:context=".MainActivity" >

<Button

android:id="@+id/button1"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_above="@+id/button2"

android:layout\_alignLeft="@+id/button2"

android:layout\_marginBottom="71dp"

android:background="#00AA2C2C"

android:onClick="register"

android:text="Register" />

<Button

android:id="@+id/button3"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignLeft="@+id/button2"

android:layout\_below="@+id/button2"

android:onClick="display\_no"

android:layout\_marginTop="74dp"

android:text="View Registered" />

<Button

android:id="@+id/button2"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:layout\_centerHorizontal="true"

android:onClick="instruct"

android:layout\_centerVertical="true"

android:text="Instructions" />

<Button

android:id="@+id/button4"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentBottom="true"

android:layout\_centerHorizontal="true"

android:onClick="verify"

android:text="Register Your Mobile Number" />

</RelativeLayout>

**24)AndroidMnifest.xml:**

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.rohitha">

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

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

<uses-permission android:name="android.permission.SEND\_SMS" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" >

</uses-permission>

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="com.example.rohitha.MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity

android:name="com.example.rohitha.Register"

android:label="@string/title\_activity\_register"

android:parentActivityName="com.example.rohitha.MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.example.rohitha.MainActivity" />

</activity>

<activity

android:name="com.example.rohitha.Display"

android:label="@string/title\_activity\_display"

android:parentActivityName="com.example.rohitha.MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.example.rohitha.MainActivity" />

</activity>

<service android:name="com.example.rohitha.BgService" />

<activity

android:name="com.example.rohitha.Instructions"

android:label="@string/title\_activity\_instructions"

android:parentActivityName="com.example.rohitha.MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.example.rohitha.MainActivity" />

</activity>

<activity

android:name="com.example.rohitha.Verify"

android:label="@string/title\_activity\_verify"

android:parentActivityName="com.example.rohitha.MainActivity" >

<meta-data

android:name="android.support.PARENT\_ACTIVITY"

android:value="com.example.rohitha.MainActivity" />

</activity>

</application>

</manifest>

**25)build.gradle:**

apply plugin: 'com.android.application'

android {

compileSdkVersion 19

buildToolsVersion "28.0.3"

defaultConfig {

applicationId "com.example.rohitha"

minSdkVersion 8

targetSdkVersion 18

versionCode 1

versionName "1.0"

testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

}

buildTypes {

release {

minifyEnabled false

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

}

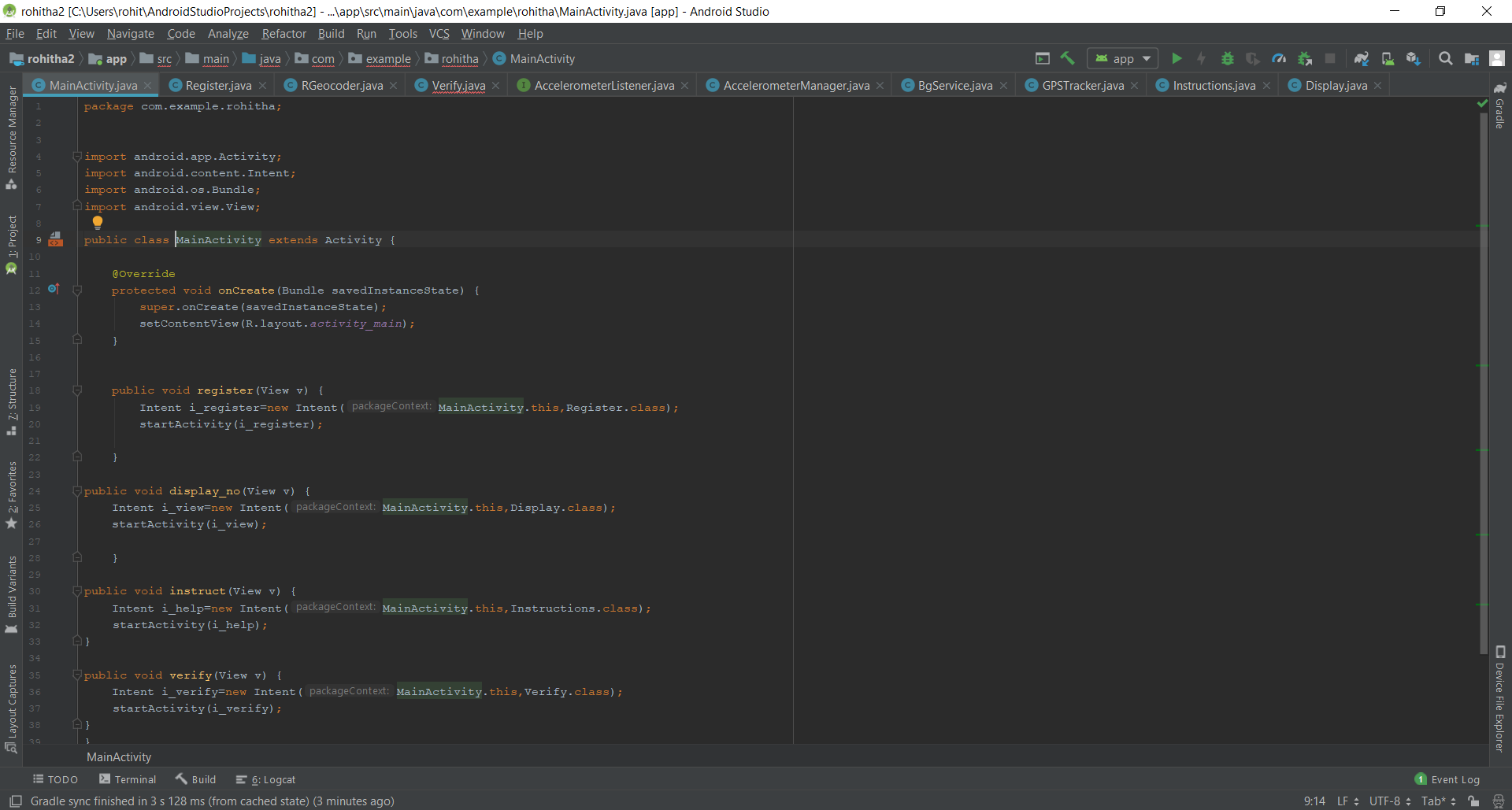
dependencies {

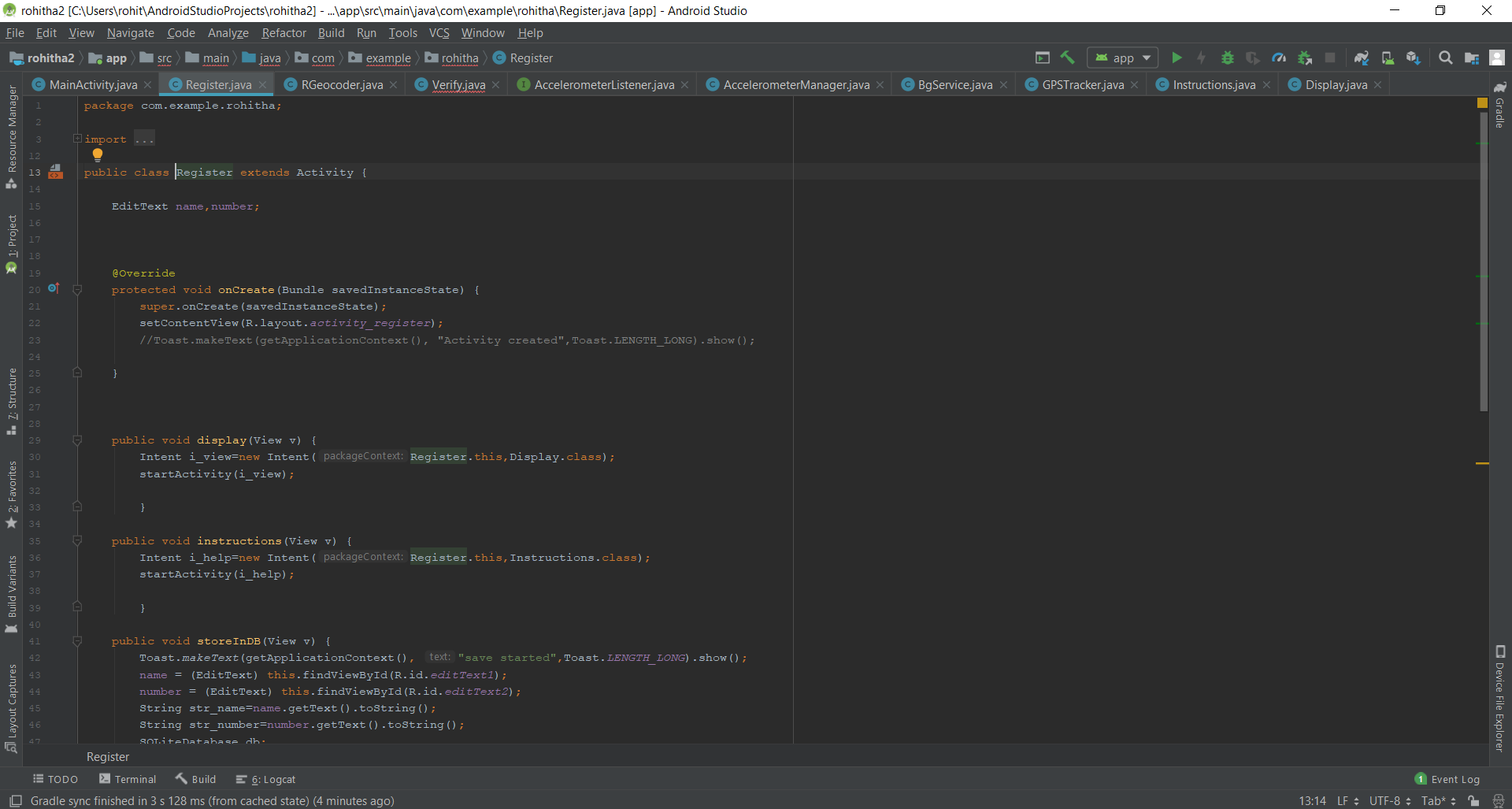
compile 'com.android.support:support-v4:19.1.0'

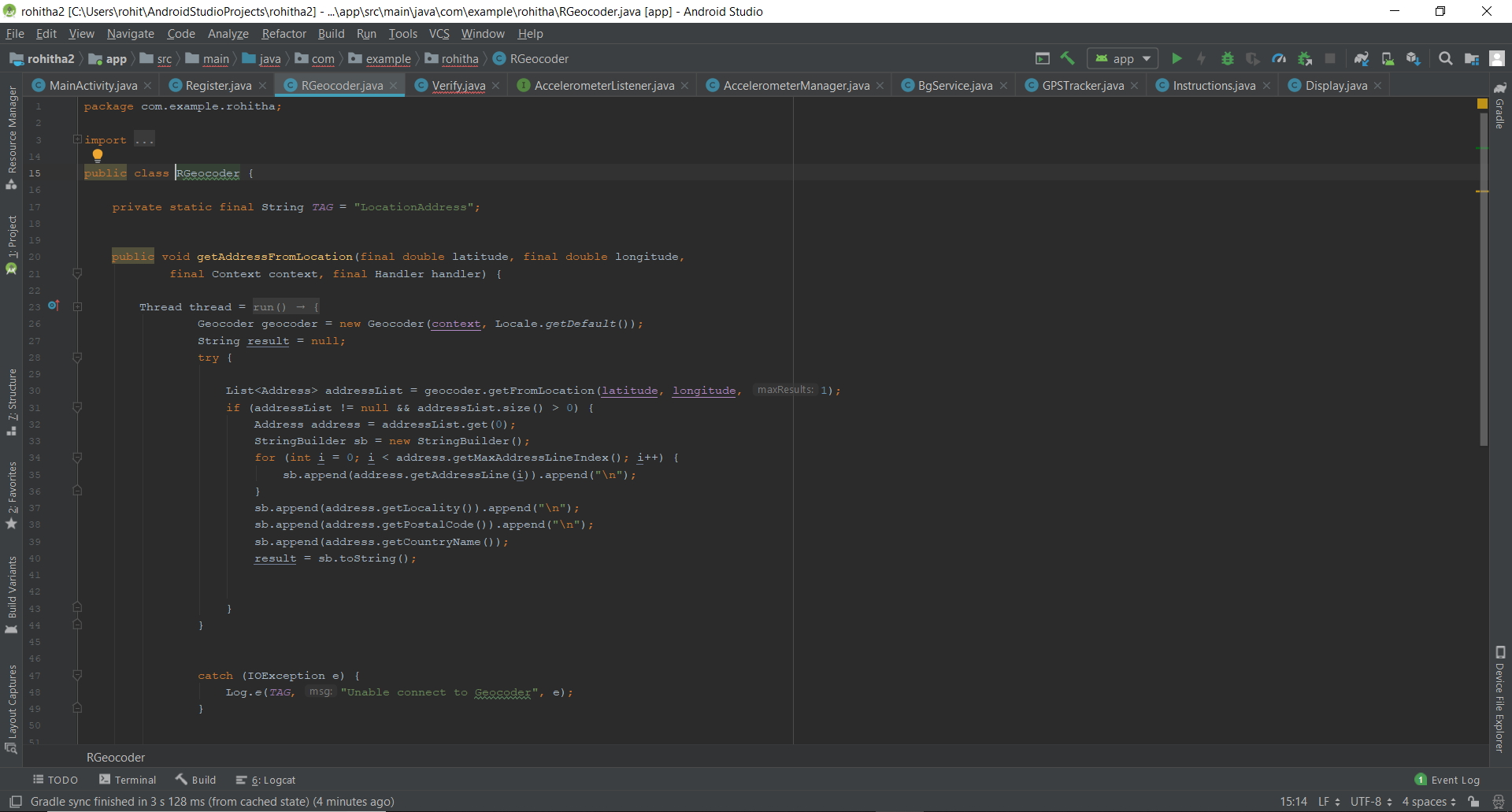
implementation 'com.google.android.gms:play-services-maps:17.0.0'

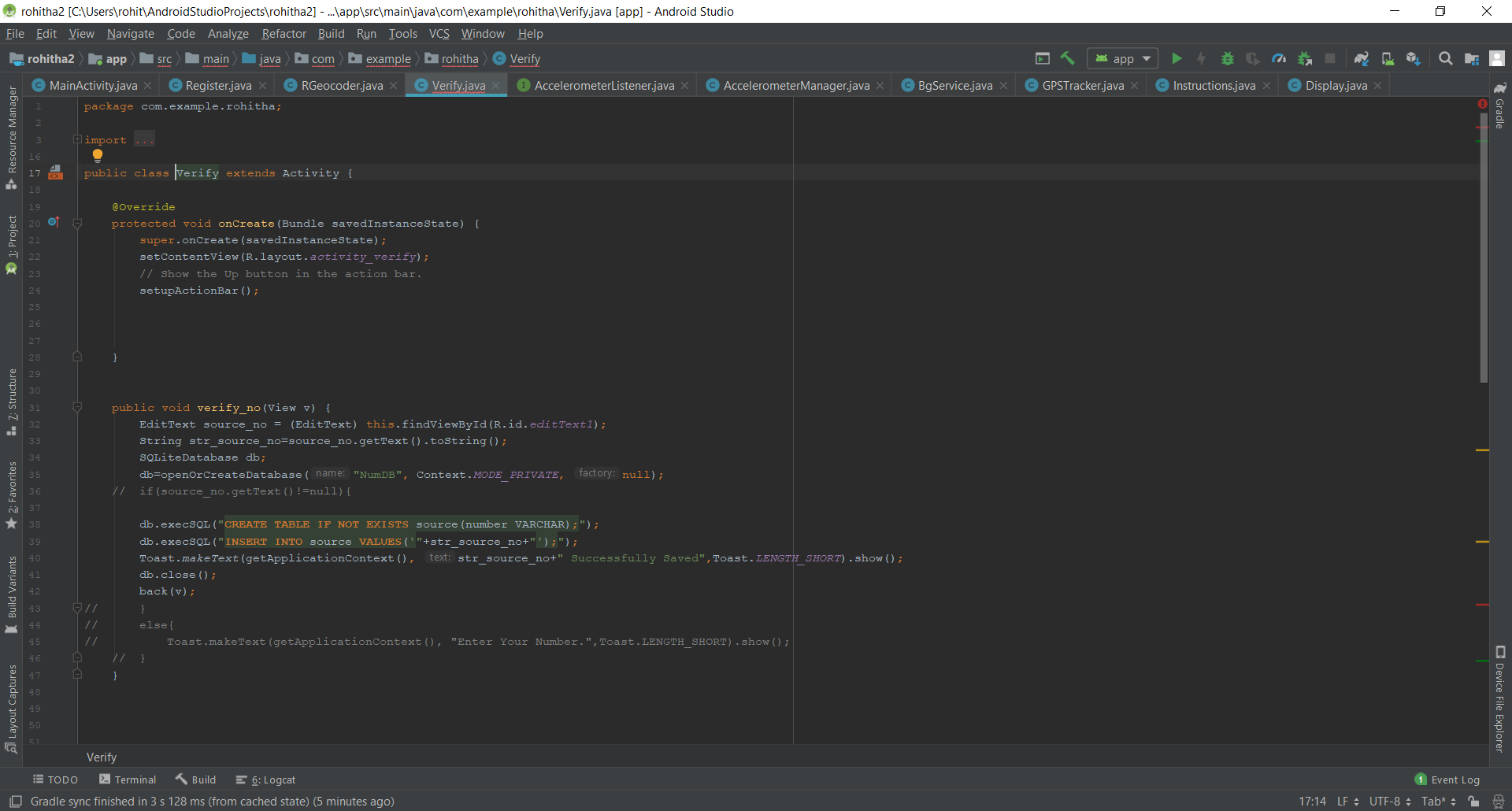
}

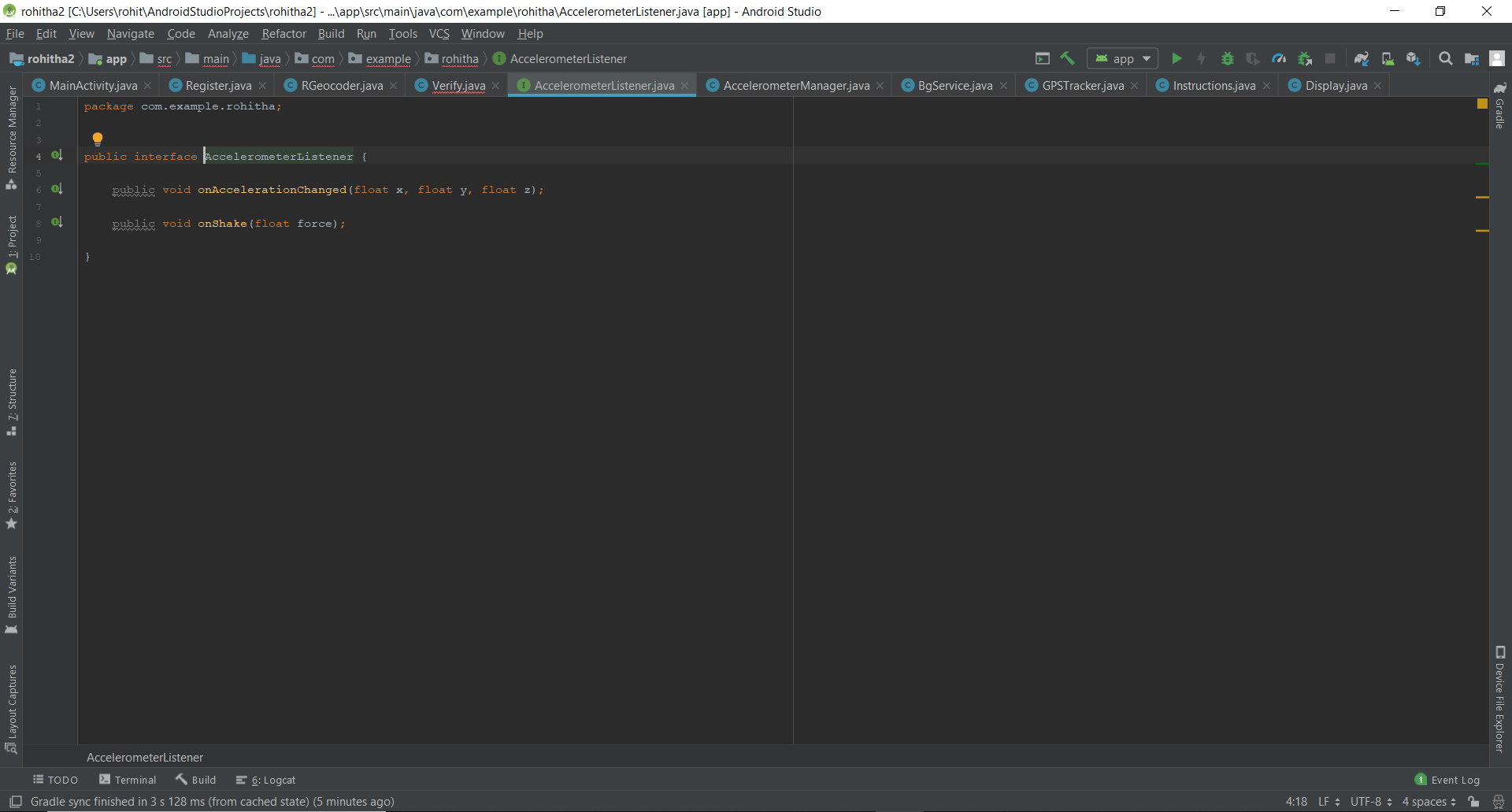
**6.3 SCREENSHOTS IN ANDROID STUDIO:**

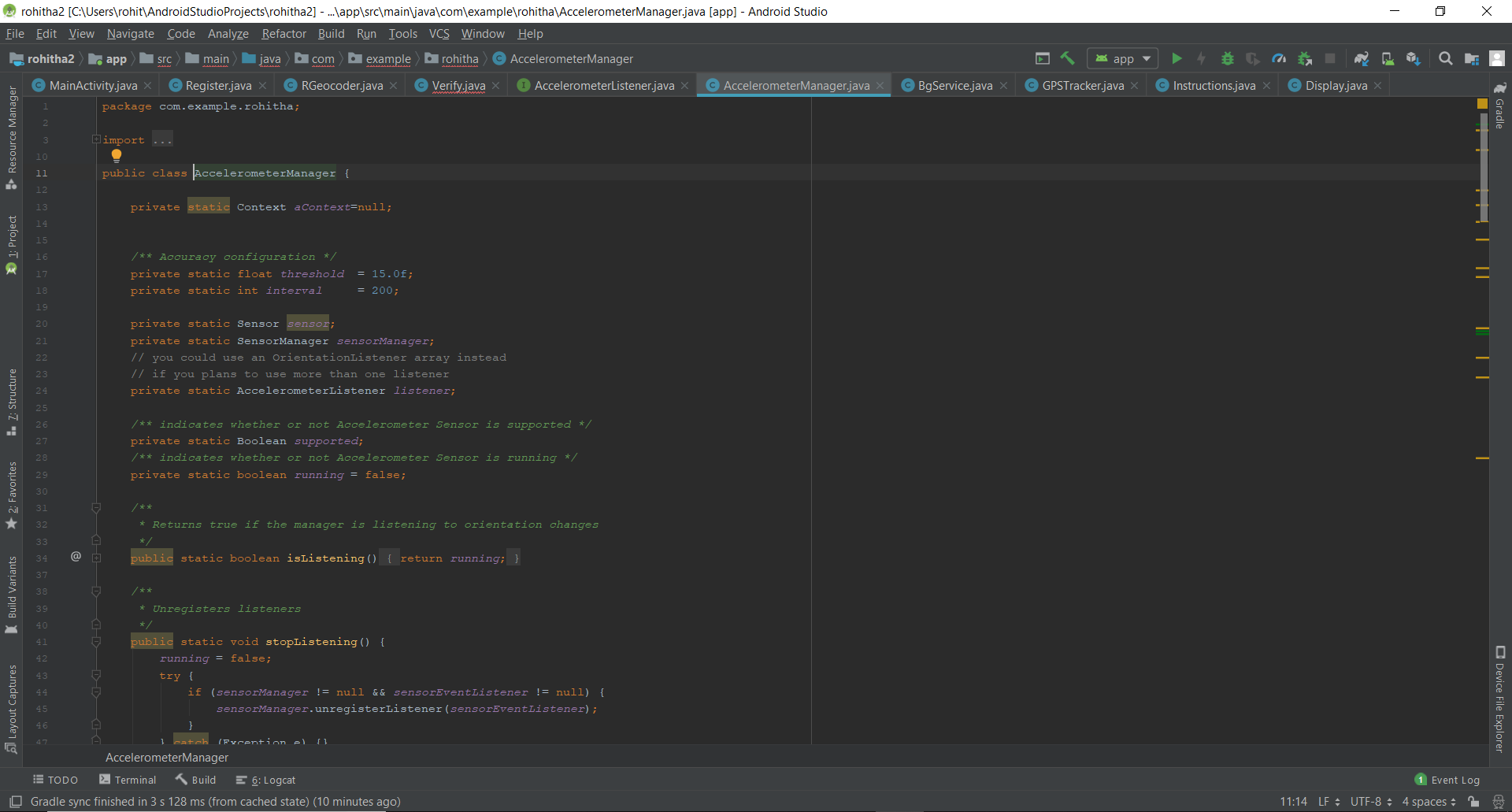


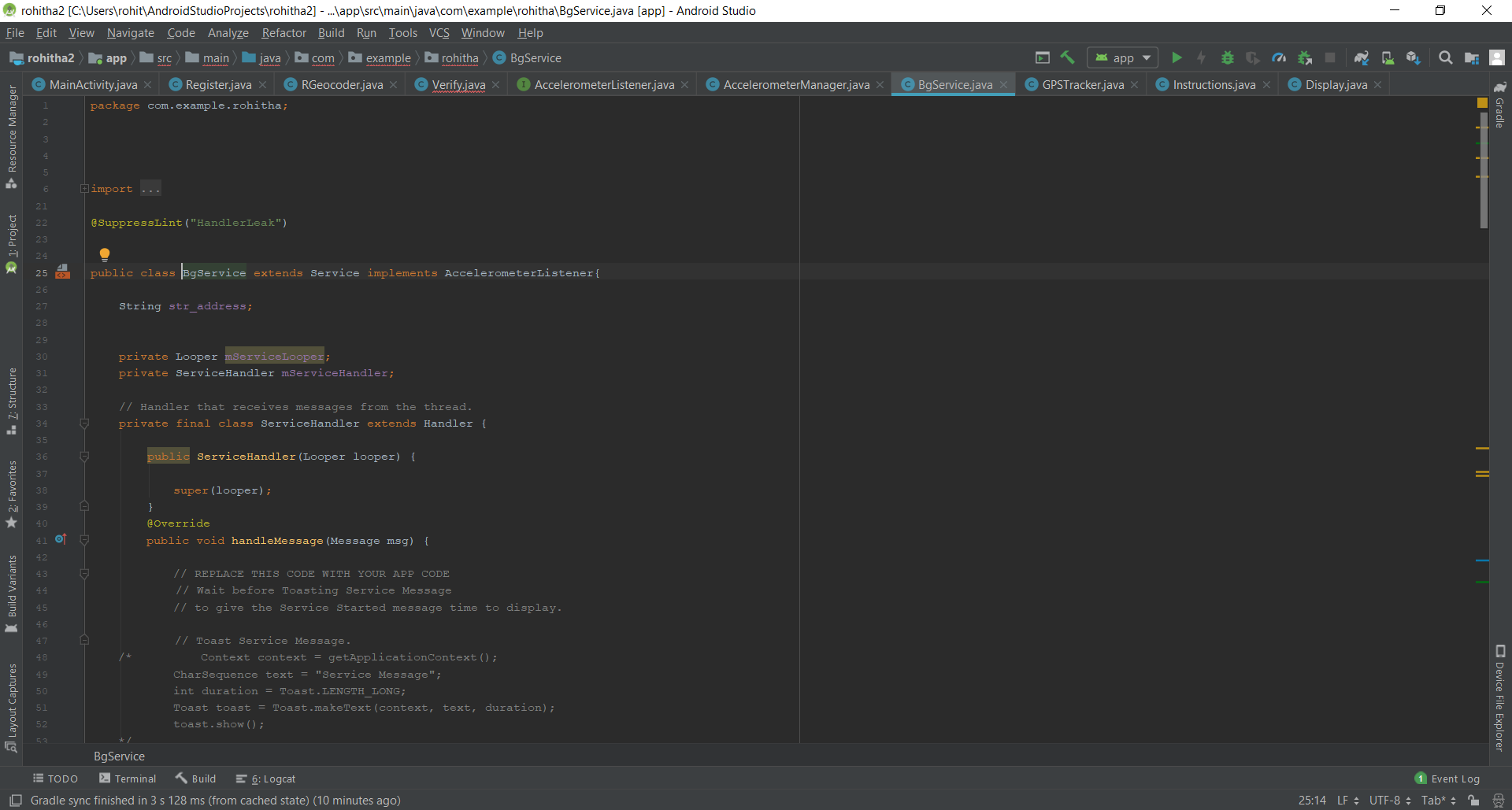


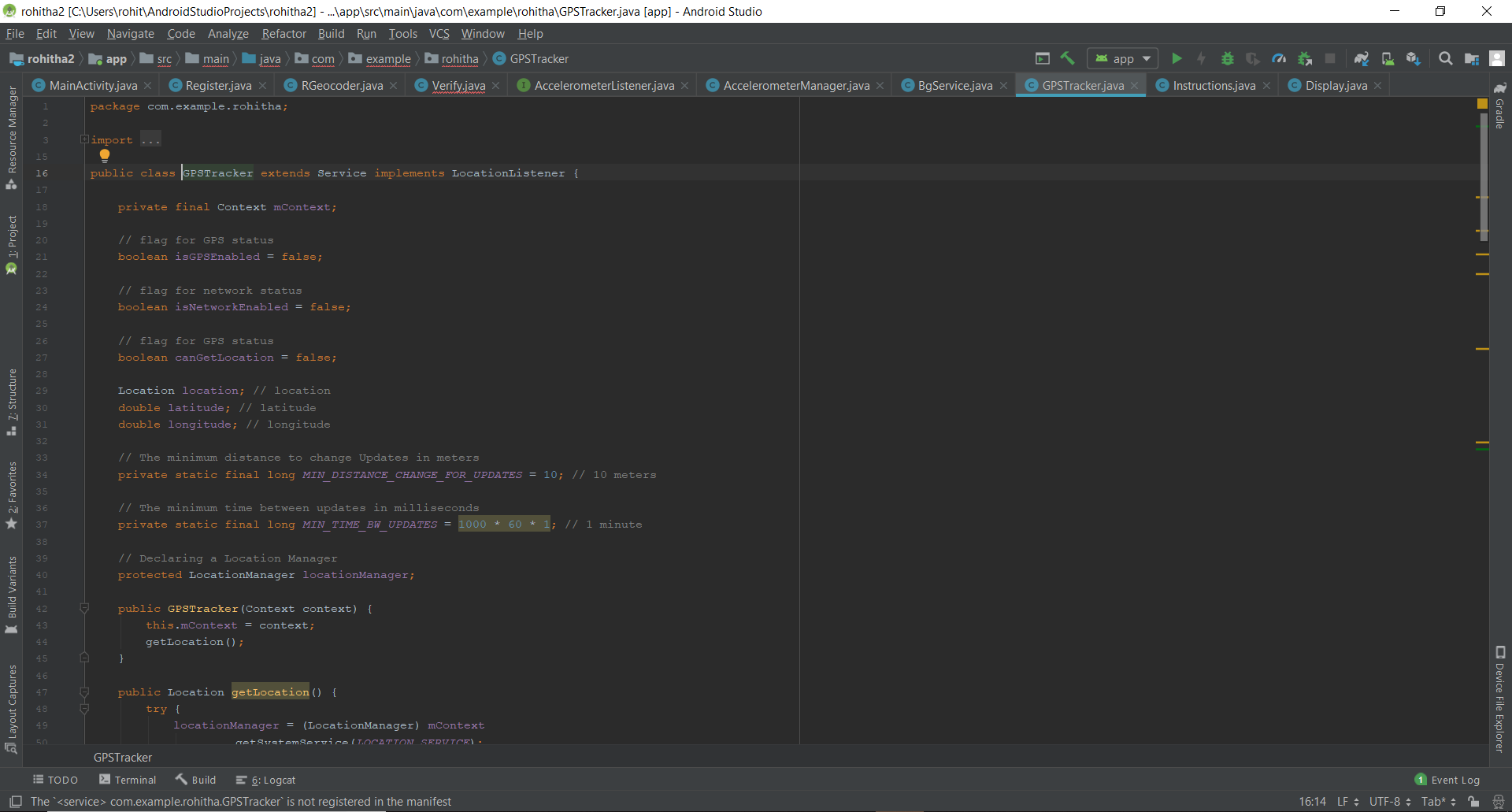


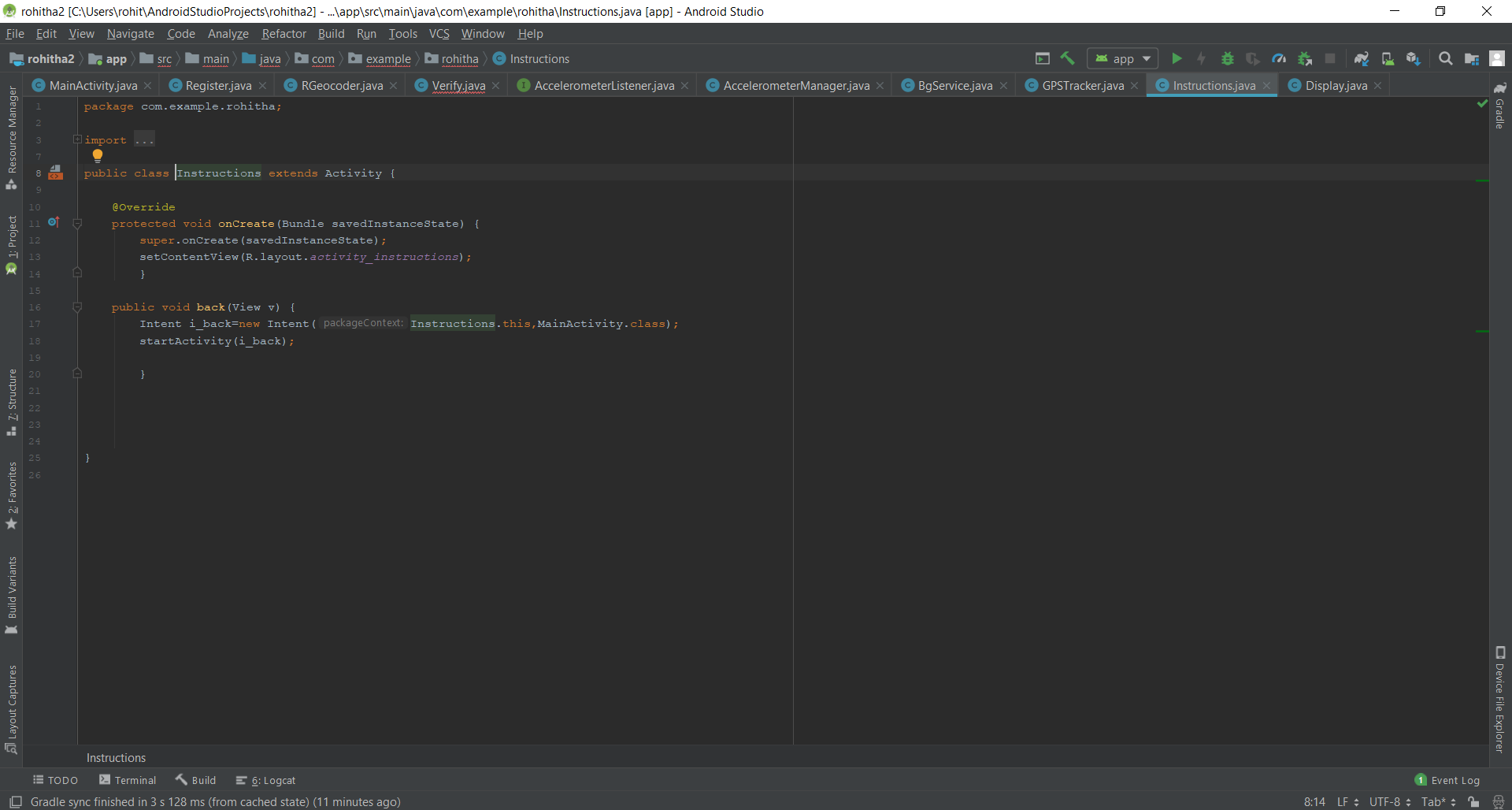


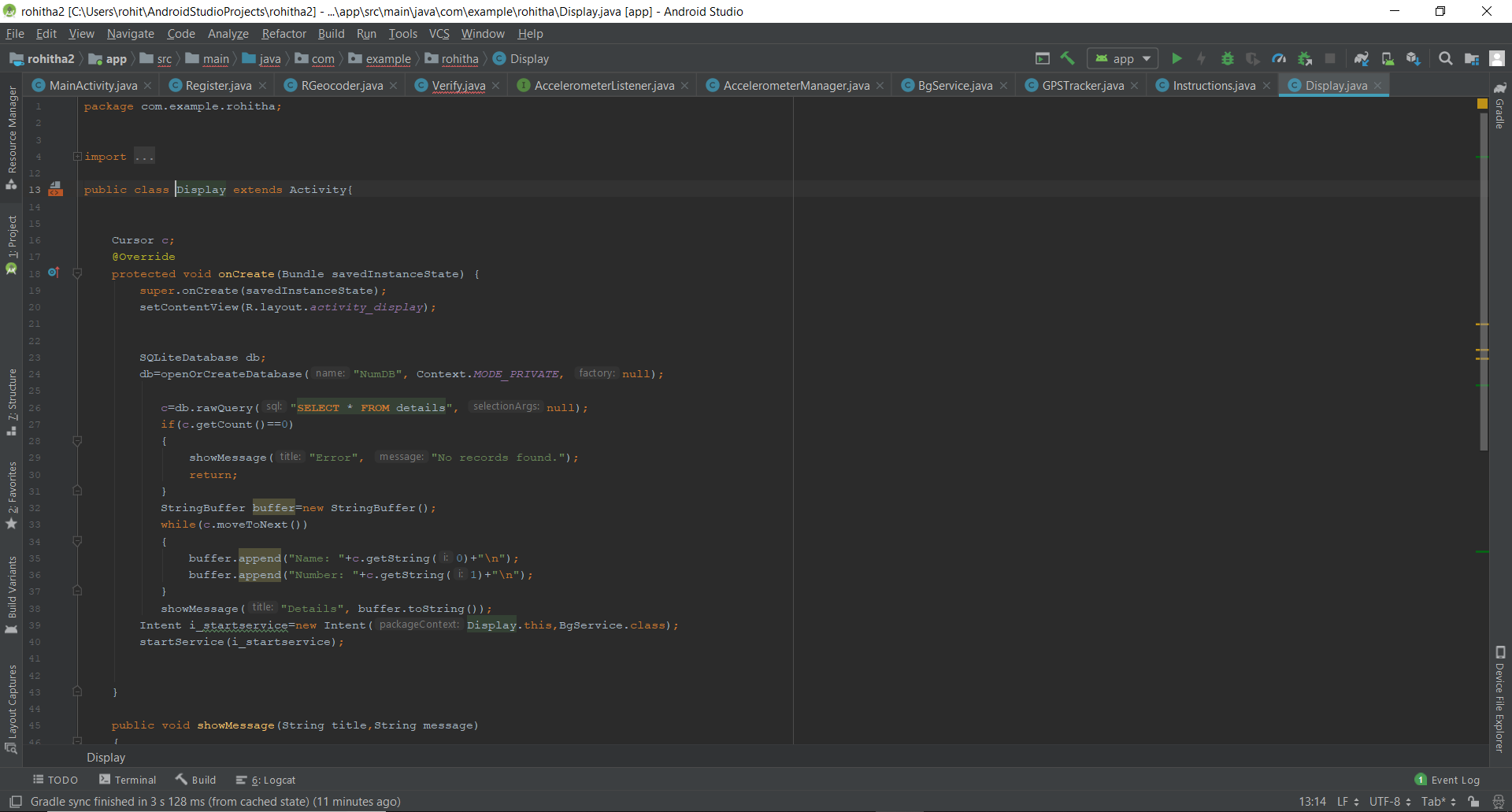








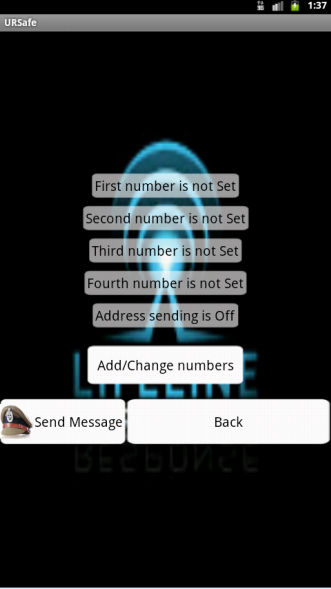




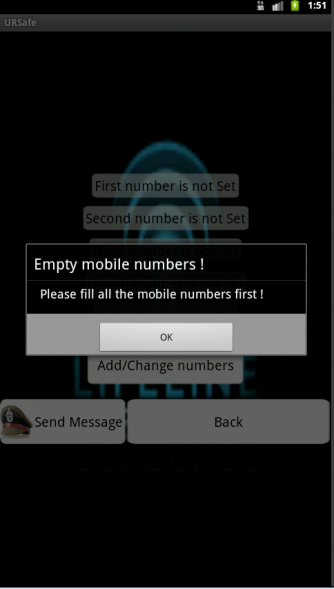
**6.4 RESULTS:**

After testing the application, we get the following results which are available in the form of screen shots as below.

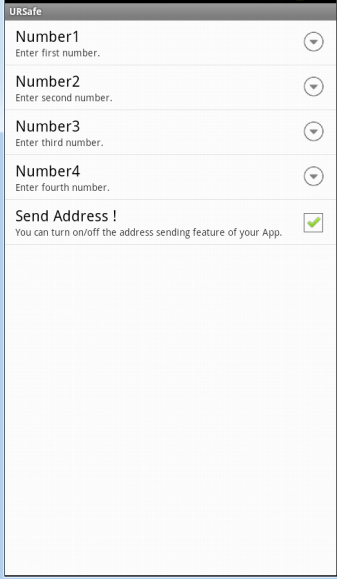
**Screenshot 1:** It is the home page of the application from where we send the message.

****

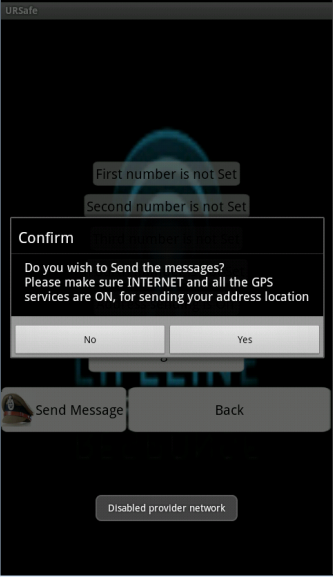
**Screenshot 2**: Shows the screen where the user has to register the four numbers as per their choice.

****

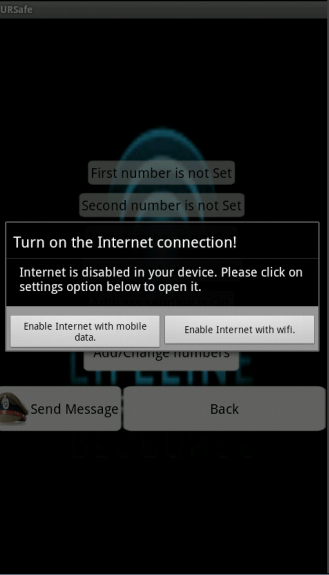
**Screenshot 3:** In this the chosen numbers are filled out.

****

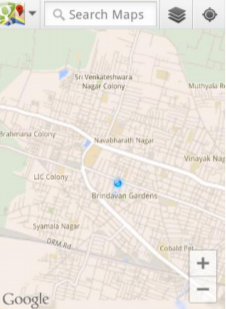
**Screenshot 4:** In this, the system looks for Internet and GPS services to be available before sending the message.

****

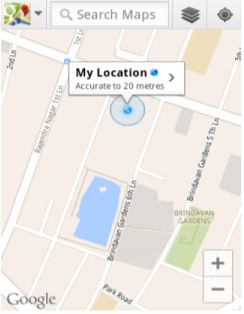
**Screenshot 5:** In this, the system recognizes the internet connection as disabled and asks for it to be enabled.

****

**Screenshot 6:** Google Maps



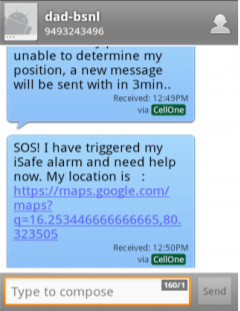
**Screenshot 7:** Map Zooming (Current location)



**Screenshot 8:** Call to guardian



**Screenshot 9:** Message to guardian



**CHAPTER 7**

**CONCLUSION AND SCOPE FOR FUTURE WORK**

**7.1 CONCLUSION**

**"A safe (community) means that all people, regardless of gender, race, ethnicity, language, disability, age or sexual orientation, have an equal right to freedom from fear and violence. We as a community have a responsibility to address the issue of violence because it belongs to everyone."**

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

We have created features, which will enhance the existing tracking system. Application stands different from the existing system as it’s not only the GPS value it makes use of but it works on text messaging services which makes application a simple & unique one.

**7.2 FUTURE WORK**

There is scope for improvement in the existing application. In particular the following facilities can be added:

* It should be able to work without internet facility.
* It should have calling facility.
* It should be able to work with zero balance.

These facilities will improve the usability of this application in the event of a genuine emergency.

**7.3 REFERENCES:**

1. Akshata V.S.1, Rumana Pathan2, Poor nima Patil3 and Farjana Nadaf4 , B’Safe & B’Secure The Door to Safety Swings, Department of Computer Science Engineering, KLS’s VDRIT, Haliyal, India, (IJCEM), ISSN: 2348 9510 Volume 1, Issue 7, October 2014.
2. S.Sangeetha1,P.Radhika PG scholar, Application For Women Safety,Department Of MCA, Panimalar Engineering College,IOSR,ISSN:2278-0661,pISSN:2278-8727, Volume 17,ISSUE 3, Ver.IV(May-Jun.2015),pp01-04.
3. Suraksha. A device to help women in distress: An initiative by a student of ITM University Gurgaon. efytimes.com. 2013. Available from: <http://efytimes.com/e1/118387/SURAKSHA-A-Device-To-Help-Women-In-Distress-An-Initiative-By-A-Student-Of-ITM-University-Gurgaon.pdf>
4. <http://timesofindia.indiatimes.com/topic/mobile-apps-for-women's-safety>
5. Bramarambika Thota , Udaya Kanchana Kumar .P, Sauver: An Android Application For Women Safety, MTech , Dept.Of ECE ,Vignan University , Guntur , India , M.sc , Computer Science , TJPS Colege,Guntur,India ,IJTEEE ,ISSN:2347-4289.VOL 3,ISSUE 05.
6. Android Based Safety Triggering Application P.Kalyanchakravarthy1, T.Lakshmi2 ,R.Rupavathi2, S.Krishnadilip2, P.Lakshmankumar2,Assitant Professor1, BTech Student CSE Department, Lendi Institute Of Engineering & Technology,Affilitated by, JNTUK,Jonada, Vizayanagaram, Andhra Pradesh, India, IJCSIT , ISSN: 0975-9646,Vol. 5(1),2014,646- 647