

APPLICATION DEVELOPMENT ON LOCATION GUARDIAN

**A MINI PROJECT REPORT
SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE ACADEMIC PROJECT**

**IN
COMPUTER SCIENCE
AND ENGINEERING BY**

VENKAT SAI CHARAN BANDI 2210315763

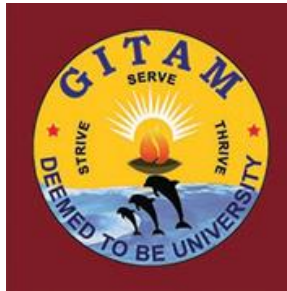
UNDER THE GUIDANCE OF

Mr.S.Durga Prasad

M.Tech(JNTU), (PhD)/ Parallel & Super Computing

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



SCHOOL OF TECHNOLOGY

GITAM (Deemed to be

university) HYDERABAD

DEPARTMENT OF COMPUTER SCIENCE

AND ENGINEERING OCTOBER – 2018




**SCHOOL OF TECHNOLOGY
GITAM (Deemed to be university)
HYDERABAD .**

**Rudraram Village, Patancheru Mandal,
Sangareddy Dist – 502329, TS, INDIA.
Ph: 08455-220556/57; Fax : 08455-20046
Website : www.gitam.edu**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE



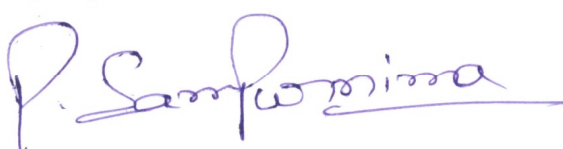
This is to certify that the mini project entitled **APPLICATION DEVELOPMENT ON LOCATION GUARDIAN** was presented satisfactorily to Department of Computer Science and Engineering , SCHOOL OF TECHNOLOGY , GITAM (DEEMED TO BE UNIVERSITY), HYDERABAD by **VENKAT SAI CHARAN BANDI** bearing H.T-NO's **2210315763** in partial fulfillment of requirement for their mini project work carried out under my guidance and supervision.


Mr. S. Durga Prasad
Project guide




Dr. S. Phani Kumar
Head of the Department
Dept of CSE

Name and Signature of the project panel members.

1. 
2. 
3. 

Dr. S. Phani Kumar
Professor & Head
Department of Computer Science & Engineering
School of Technology
GITAM (Deemed to be University)
Rudraram - 502 329
Sangareddy District, Telangana State, INDIA

DECLARATION

I, **VENKAT SAI CHARAN BANDI**, bearing roll number **2210315763** hereby declare that the project report entitled "**APPLICATION DEVELOPMENT ON LOCATION GUARDIAN**", under the guidance of **Mr. S. Durga Prasad, Assistant Professor**, Department of Computer Science and Engineering, School of Technology, GITAM (DEEMED TO BE UNIVERSITY), Hyderabad, has been submitted for Project evaluation.

This is a record of bonafide work carried out by me and the content embodied in this project have not been reproduced /copied from any source. The content embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

Name

Registration No.

Signature

B V Sai Charan

2210315763



ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success.

We are grateful to our project guide **Mr.S.Durga Prasad**, Assistant Professor, GITAM (DEEMED TO BE UNIVERSITY), for the guidance, inspiration and constructive suggestions that helped us in the preparation of this project.

We thank our project coordinator **Mr. B. Rajendra Prasad Babu**, Assistant Professor for his support and encouragement throughout the course of the project work.

At the outset, we thank our Head of the Department **Dr. S. Phani Kumar** and our honourable Pro-Vice Chancellor **Prof. N. Siva Prasad** for the moral support and the excellent facilities provided. I would also like to thank all the teaching and non-teaching staff members of Computer Science department who have extended their full cooperation during course of my industrial training.

We wish to express our warm and grateful thanks to our project coordinator for the guidance and assistance he provided in completing our project successfully.

I thank all my friends who helped me sharing knowledge and by providing material to complete the project in time.

Place: Hyderabad

VENKAT SAI CHARAN BANDI

(2210315763)

Date

CONTENTS

Abstract	i
List of Figures	ii
List of Screens	iii
1. INTRODUCTION	1
1.1 About Project	1
1.2 Objective	1
1.3 Scope	1
2. LITERATURE SURVEY REPORT	2
2.1 Existing System	2
2.2 Proposed System	2
2.3 Feasibility Study	2
2.3.1 Technical Feasibility	3
2.3.2 Operational Feasibility	3
2.3.3 Economic Feasibility	3
3. REQUIRMENT SPECIFICATION	4
3.1 Functional Requirements	4
3.2 Non Functional Requirements	4
3.3 Hardware and Software Requirements	5
3.3.1 Hardware Requirements	5
3.3.2 Software Requirements	5
4. TECHNOLOGIES USED	6
4.1 Android	6
4.2 Android Architecture	6
4.3 Hardware Running Android	7
4.4 Software Development Kit	7
4.5 Core JAVA	7
4.5.1 Java Technology	8
4.5.2 Importance of Java to the Internet	8
4.5.3 Features of Java Security	8
4.5.4 The Byte Code	9

4.5.5 Java Virtual Machine	9
4.6 Benefits of the Java Collections Framework	10
5. SYSTEM ARCHITECTURE AND DESIGN	14
5.1 Architecture of Android Security Application	14
5.2 UML	15
5.2.1 UML Concepts	15
5.2.2 Building Blocks of the UML	15
5.2.3 Things in the UML	16
5.2.4 Relationships in the UML	16
5.3 Use Case Diagrams	17
5.4 Class Diagrams	18
5.5 Sequence Diagram	19
5.6 Activity Diagram	19
5.7 Collaborative Diagram	20
5.8 Object Diagram	20
5.9 State Chart Diagram	21
5.10 Component Diagram	22
5.11 Deployment Diagram	22
6. DETAILED DESCRIPTION OF COMPONENTS	23
7. TESTING	45
8. SCREENSHOTS	47
9. CONCLUSION	51
REFERENCES	

ABSTRACT

This is an android application which is used for security purpose. Here, the app has guardians and users, where user adds other people as guardians. When the user does not pick up the call, the details of the users location is sent to the guardian. Here the address will be sent, which can be located on the map. And this application should be installed on both the guardians and users phone. So, only the guardian will receive the location details of the user if the call is not received, but not when the user calls and when guardian does not lift the phone.

LIST OF FIGURES

Fig 4.1	Android Operating System	6
Fig 4.2	Java Editions	10
Fig 4.3	Core Collections Interface	12
Fig 5.1	Application Architecture Design	14
Fig 5.2	Usecase Diagram For Andriod Security Application	17
Fig 5.3	Class Diagram For Andriod Security Application	18
Fig 5.4	Sequence Diagram For Andriod Security Application	19
Fig 5.5	Activity Diagram For Andriod Security Application	19
Fig 5.6	Collaboration Diagram For Andriod Security Application	20
Fig 5.7	Object Diagram For Andriod Security Application	20
Fig 5.8	State Chart Diagram For Andriod Security Application	21
Fig 5.9	Component Diagram For Andriod Security Application	22
Fig 5.10	Deployment Diagram For Andriod Security Application	22
Fig 7.1	Test Cases	46

LIST OF SCREENS

8.1	Home Screen	47
8.2	Registration	47
8.3	Adding Phone Numbers	48
8.4	Country Code	48
8.5	Guardian List	49
8.6	SMS Format	49
8.7	Map Location	50

1. INTRODUCTION

1.1 ABOUT PROJECT

This is an application which is used for security purpose. This is an android application. Here we will be having guardians and users, where users add other people as his guardians. When the user does not lift the phone, the details of his location are sent to the guardian. Here the address will be sent using which we can locate it in the map. And This application should be installed on both the guardian's and user's phone. So, only the guardian will receive the location details of the user if he does not lift the phone but not when the user calls and when the guardian does not lift the phone.

1.2 OBJECTIVE

- Here the main objective is to provide security for people.
- The main purpose is to send the location of user to the guardian.
- We can get the location of the person when the person does not lift the phone.
- We can even see the map regarding the location.

1.3 SCOPE

- Internet connection is compulsory for this application to send the location of the person.
- Once a guardian calls a person and if the person does not lift the phone then automatically the person's location details will be sent to the guardian.
- The application should be installed on both the guardian's and user's phone. So, only the guardian will receive the location details of the user if he does not lift the phone but not when the user calls and when the guardian does not lift the phone.
- In another method, both the user and guardian will be having application and both of them will add each other as a guardian and the location details will be send to both of them in case where either of them does not attend the call.

2. LITERATURE SURVEY REPORT

- Current problems people face in society.
- Now a day's security is becoming a major issue.
- When the user does not lift the phone the dear ones of that user may get worried.

2.1 EXISTING SYSTEM:

1. At times, people might not attend the call due to some reason and the people who care about them might get tensed.

2. We are not having any application for security reasons like this. Hence, we are proposing the following solution.

2.2 PROPOSED SYSTEM:

1. Here the location is sent in the form of latitude and longitude.

2. The location details will be converted into address internally in the application and the address is sent to the guardian.

3. We even provide a special feature of seeing that location in the map using this application.

2.3 FEASIBILITY STUDY:

The next step in analysis is to verify the feasibility of the proposed system. "All projects are feasible given unlimited resources and infinite time". But in reality both resources and time are scarce. Project should confirm to time bounce and should be optimal in there consumption of resources. This place a constant are approval of any project.

These feasibility studies are 3 types:

- Technical feasibility
- Operational feasibility
- Economic feasibility

2.3.1 Technical Feasibility:

To determine whether the proposed system is technically feasible, we should take into consideration the technical issues involved behind the system.

2.3.2 Operational Feasibility:

To determine the operational feasibility of the system we should take into consideration the awareness level of the users. This system is operational feasible since the users are familiar with the technologies and hence there is no need to gear up the personnel to use system. Also the system is very friendly and to use.

2.3.3 Economic Feasibility:

To decide whether a project is economically feasible, we have to consider various factors as:

- Cost benefit analysis
- Long-term returns
- Maintenance costs

It requires average computing capabilities and access to internet, which are very basic requirements and can be afforded by any organization hence it doesn't incur additional economic overheads, which renders the system economically feasible.

3.REQUIREMENT SPECIFICATIONS

3.1 FUNCTIONAL REQUIREMENTS:

No.	Module	Requirement/ Functionality	Essential or Desirable	Description of the Requirement
RS1	Add Contacts	Add guardians contacts	Essential	User can add contacts to guardians
RS2	Retrieving calls	Get Missed calls	Essential	Getting the missed call who added in guardian.
RS3	Sending Sms	Send location's latitude and longitude	Essential	Whenever we get the missed call from guardians at that time send the location to particular guardian.
RS4	Show map	Displaying the map	Essential	Whenever we get message regarding location then show the map
RS5	Message log	Storing the location messages	Essential	Storing and showing the messages of the location.

3.2 NON FUNCTIONAL REQUIREMENTS:

1. There is a 24/7 availability of this software.
2. Internet should be present in the mobile to use this Software.
3. Both user and guardian should be having this application but it is not a compulsion that both of them should add one another as their guardian but at least one of them should act as a guardian to another person.
4. This application depends on android sdk we use. It is better if the user uses the phone which has the current version of android. We need to use android supported mobile only. In this application, we assume the user to add guardians who get the location when the person whom he called does not lift the phone.

3.3 HARDWARE AND SOFTWARE REQUIREMENTS:

3.3.1 Hardware Requirements:

- Mobile with android compatibility
- **RAM:** 512 MB RAM and above
- **Hard disk:** 1 GB

3.3.2 Software Requirements:

- **IDE:** Eclipse
- **OS:** Windows XP and above
- **SDK:** Android 2.2 and above

4. TECHNOLOGIES USED

4.1 ANDROID:

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

4.2 ANDROID ARCHITECTURE:

The following diagram shows the major components of the Android operating system.

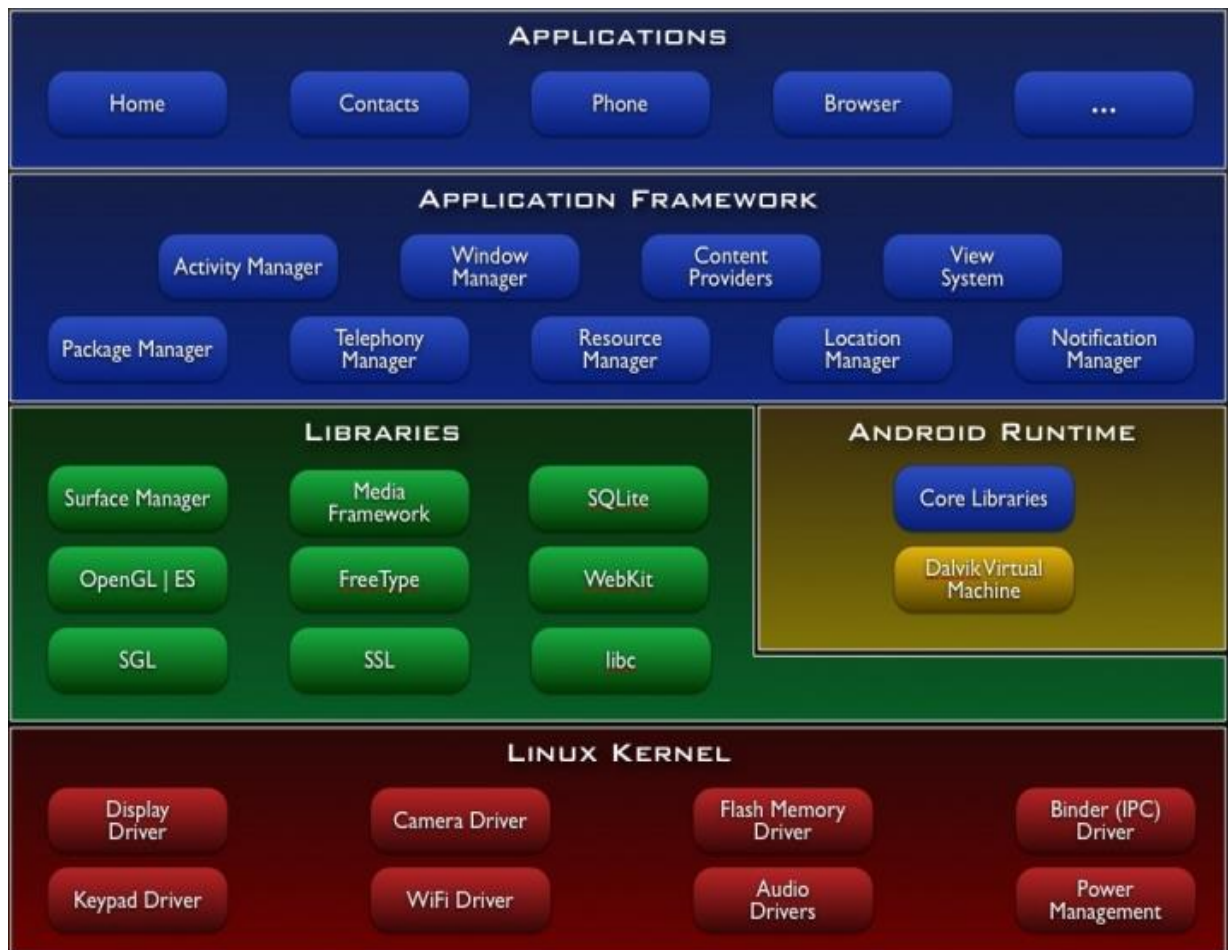


Fig 4.1 Android Operating System

4.3 HARDWARE RUNNING ANDROID:

The Android OS can be used as an operating system for cellphones, netbooks and tablet PCs, including the Dell Streak, Samsung Galaxy Tab and other devices.

The world's first TV running Android, called Scandinavia, has also been launched by the company People of Lava.

The first commercially available phone to run the Android operating system was the HTC Dream, released on 22 October 2008.

4.4 SOFTWARE DEVELOPMENT KIT:

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator (based on QEMU), documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution),

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible to root user only for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executable), resource files, etc.

4.5 CORE JAVA:

Java is a machine independent language. It has a gui interface. It debugs the program with the help of both compiler and interpreter. It has many features and functionalities. Using java, we can develop applications which work on a gui interface. Using java we can develop windows applications and web applications. Even the internal functioning of android applications is written in java language.

4.5.1 Java Technology:

Initially the language was called as “oak” but it was renamed as “Java” in 1995. The primary motivation of this language was the need for a platform-independent (i.e., architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language.
- Java is cohesive and consistent.
- Except for those constraints imposed by the Internet environment, Java gives the programmer, full control.

4.5.2 Importance of Java to the Internet:

Java has had a profound effect on the Internet. This is because; Java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the Server and the Personal computer. They are: Passive information and Dynamic active programs. The Dynamic, Self-executing programs cause serious problems in the areas of Security and probability. But, Java addresses those concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

4.5.3 Features of Java Security:

Every time you that you download a “normal” program, you are risking a viral infection. Prior to Java, most users did not download executable programs frequently, and those who did scan them for viruses prior to execution. Most users still worried about the possibility of infecting their systems with a virus. In addition, another type of malicious program exists that must be guarded against. This type of program can gather private information, such as credit card numbers, bank account balances, and passwords. Java answers both these concerns by providing a “firewall” between a network application and your computer.

4.5.4 The Byte Code:

The key that allows the Java to solve the security and portability problems is that the output of Java compiler is Byte code. Byte code is a highly optimized set of instructions designed to be executed by the Java run-time system, which is called the Java Virtual Machine (JVM). That is, in its standard form, the JVM is an interpreter for byte code.

Translating a Java program into byte code helps makes it much easier to run a program in a wide variety of environments. The reason is, once the run-time package exists for a given system, any Java program can run on it.

Although Java was designed for interpretation, there is technically nothing about Java that prevents on-the-fly compilation of byte code into native code. Sun has just completed its Just in Time (JIT) compiler for byte code. When the JIT compiler is a part of JVM, it compiles byte code into executable code in real time, on a piece-by-piece, demand basis. It is not possible to compile an entire Java program into executable code all at once, because Java performs various run-time checks that can be done only at run time. The JIT compiles code, as it is needed, during execution.

4.5.5 Java Virtual Machine (JVM):

Beyond the language, there is the Java virtual machine. The Java virtual machine is an important element of the Java technology. The virtual machine can be embedded within a web browser or an operating system. Once a piece of Java code is loaded onto a machine, it is verified. As part of the loading process, a class loader is invoked and does byte code verification makes sure that the code that's has been generated by the compiler will not corrupt the machine that it's loaded on. Byte code verification takes place at the end of the compilation process to make sure that is all accurate and correct. So byte code verification is integral to the compiling and executing of Java code.

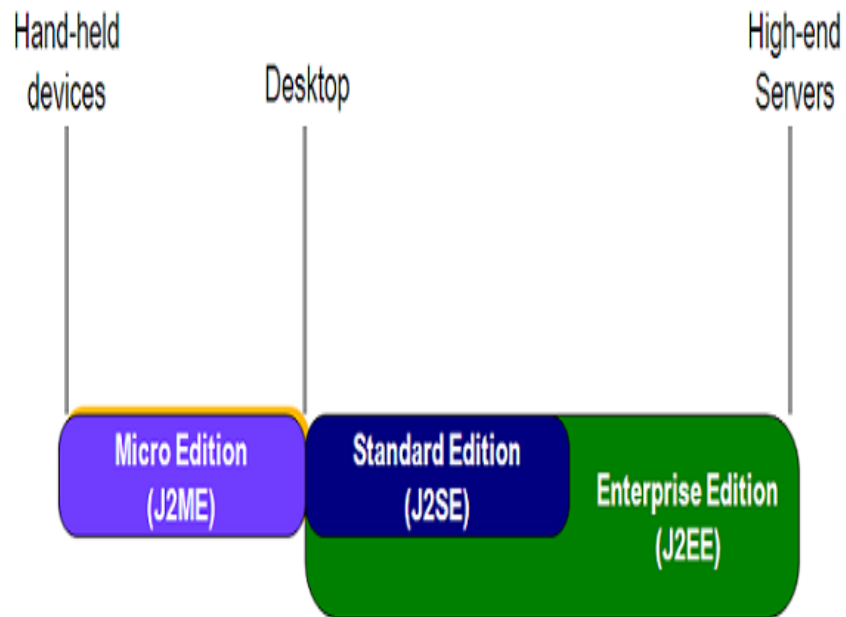


Fig 4.2 Java Editions

4.6 BENEFITS OF THE JAVA COLLECTIONS FRAMEWORK:

The Java Collections Framework Provides The Following Benefits:

Reduces programming effort:

By providing useful data structures and algorithms, the Collections Framework frees you to concentrate on the important parts of your program rather than on the low-level "plumbing" required to make it work. By facilitating interoperability among unrelated APIs, the Java Collections Framework frees you from writing adapter objects or conversion code to connect APIs.

Increases Program Speed and Quality:

This Collections Framework provides high performance, high-quality implementations of useful data structures and algorithms. The various implementations of each interface are interchangeable, so programs can be easily tuned by switching collection implementations. Because you're freed from the drudgery of writing your own data structures, you'll have more time to devote to improving programs' quality and performance.

Allows Interoperability Among Unrelated APIs:

The collection interfaces are the vernacular by which APIs pass collections back and forth. If my network administration API furnishes a collection of node names and if your GUI toolkit expects a collection of column headings, our APIs will interoperate seamlessly, even though they were written independently.

Reduces Effort To Learn And To Use New APIs:

Many APIs naturally take collections on input and furnish them as output. In the past, each such API had a small sub-API devoted to manipulating its collections. There was little consistency among these ad hoc collections sub-APIs, so you had to learn each one from scratch, and it was easy to make mistakes when using them.

Reduces effort to design new APIs: This is the flip side of the previous advantage. Designers and implementers don't have to reinvent the wheel each time they create an API that relies on collections; instead, they can use standard collection interfaces.

Fosters software reuse:

New data structures that conform to the standard collection interfaces are by nature reusable. The same goes for new algorithms that operate on objects that implement these interfaces.

The Core Collection Interfaces:

A Set is a special kind of Collection, a SortedSet is a special kind of Set, and so forth. Note also that the hierarchy consists of two distinct trees — a Map is not a true Collection.

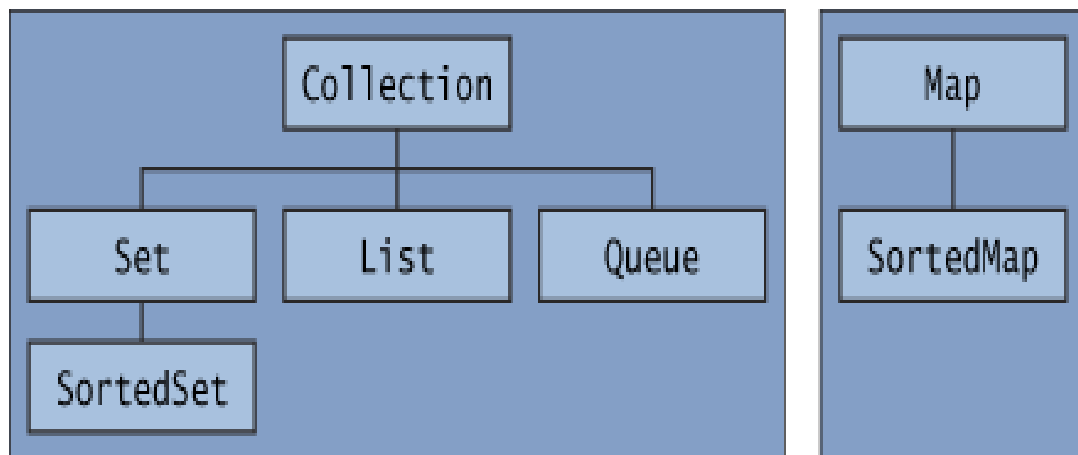


Fig 4.3 Core Collection Interface

The Following List Describes The Core Collection Interfaces:

Collection — the root of the collection hierarchy. A collection represents a group of objects known as its elements. The Collection interface is the least common denominator that all collections implement and is used to pass collections around and to manipulate them when maximum generality is desired. Some types of collections allow duplicate elements, and others do not. Some are ordered and others are unordered. The Java platform doesn't provide any direct implementations of this interface but provides implementations of more specific sub interfaces, such as Set and List. Also see The Collection Interface section.

Set — a collection that cannot contain duplicate elements. This interface models the mathematical set abstraction and is used to represent sets, such as the cards comprising a poker hand, the courses making up a student's schedule, or the processes running on a machine. See also The Set Interface section.

List — an ordered collection (sometimes called a sequence). Lists can contain duplicate elements. The user of a List generally has precise control over where in the list each element is inserted and can access elements by their integer index (position). If you've used Vector, you're familiar with the general flavor of List. Also see The List Interface section.

Queue — a collection used to hold multiple elements prior to processing. Besides basic Collection operations, a Queue provides additional insertion, extraction,

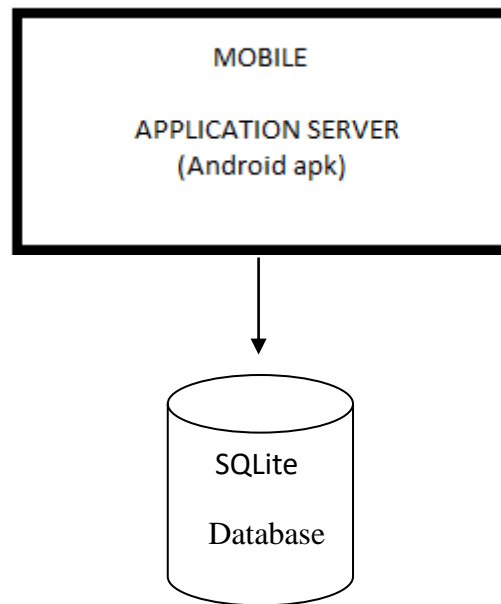
and inspection operations. Queues typically, but do not necessarily, order elements in a FIFO (first-in, first-out) manner. Among the exceptions are priority queues, which order elements according to a supplied comparator or the elements' natural, ordering? Whatever the ordering used, the head of the queue is the element that would be removed by a call to `remove` or `poll`. In a FIFO queue, all new elements are inserted at the tail of the queue. Other kinds of queues may use different placement rules. Every Queue implementation must specify its ordering properties. Also see The Queue Interface section.

Map:

`SortedSet` — a Set that maintains its elements in ascending orders. Several additional operations are provided to take advantage of the ordering. Sorted sets are used for naturally ordered sets, such as word lists and membership rolls. Also see The `SortedSet` Interface section.

`SortedMap` — a Map that maintains its mappings in ascending key order. This is the Map analog of `SortedSet`. Sorted maps are used for naturally ordered collections of key/value pairs, such as dictionaries and telephone directories. Also see The `SortedMap` Interface section.

5. SYSTEM ARCHITECTURE AND DESIGN



APPLICATION ARCHITECTURE DESIGN

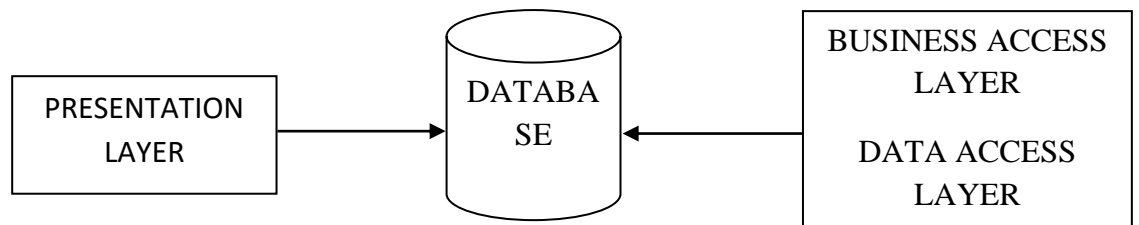


Fig 5.1 Application Architecture Design

5.1 ARCHITECTURE OF ANDROID SECURITY APPLICATION:

To execute or run any android application on mobile u need an apk file. Here the mobile itself acts as an application server in which the application is deployed. We add guardians to our list which will be stored in the database. Not only the details. This application comprises of a two-tier architecture. A layer consists of the presentation layer which is the interface designed and another is the data access and

business access layer which consists of the logic of functioning of the application and the data will be accessed in this layer. Everything is extracted from the database and shown to the user.

5.2 UML:

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

5.2.1 UML Concepts:

The Unified Modeling Language (UML) is a standard language for writing software blue prints. The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modeling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modeling yields an understanding of a system.

5.2.2 Building Blocks of the UML:

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

5.2.3 Things in the UML:

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

5.2.4 Relationships in the UML:

There are four kinds of relationships in the UML:

A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).



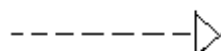
An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.



A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).



A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.



5.3 USE CASE DIAGRAM:

Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality.

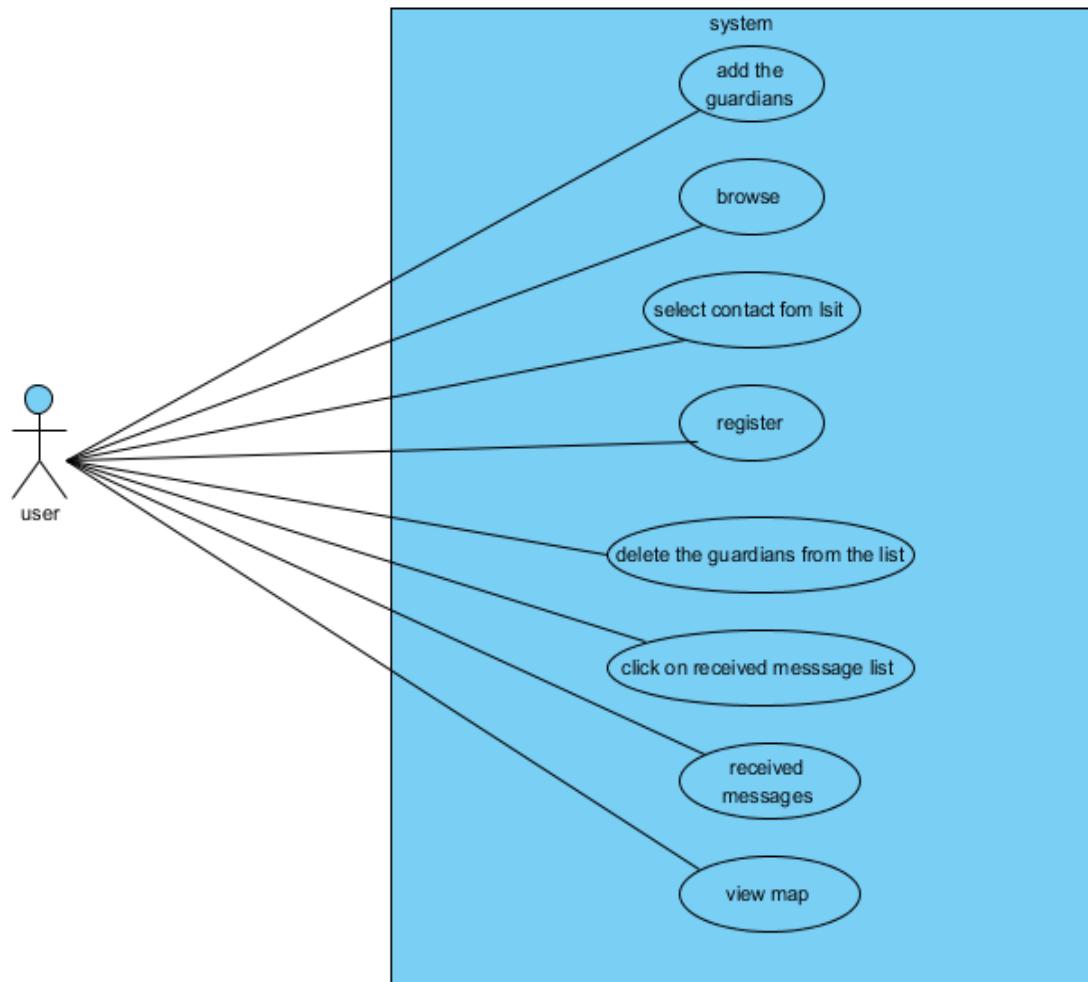


Fig 5.2 Usecase Diagram For Andriod Security Application

5.4 CLASS DIAGRAM:

Class diagrams identify the class structure of a system, including the properties and methods of each class. Also depicted are the various relationships that can exist between classes, such as an inheritance relationship. The Class diagram is one of the most widely used diagrams from the UML specification.

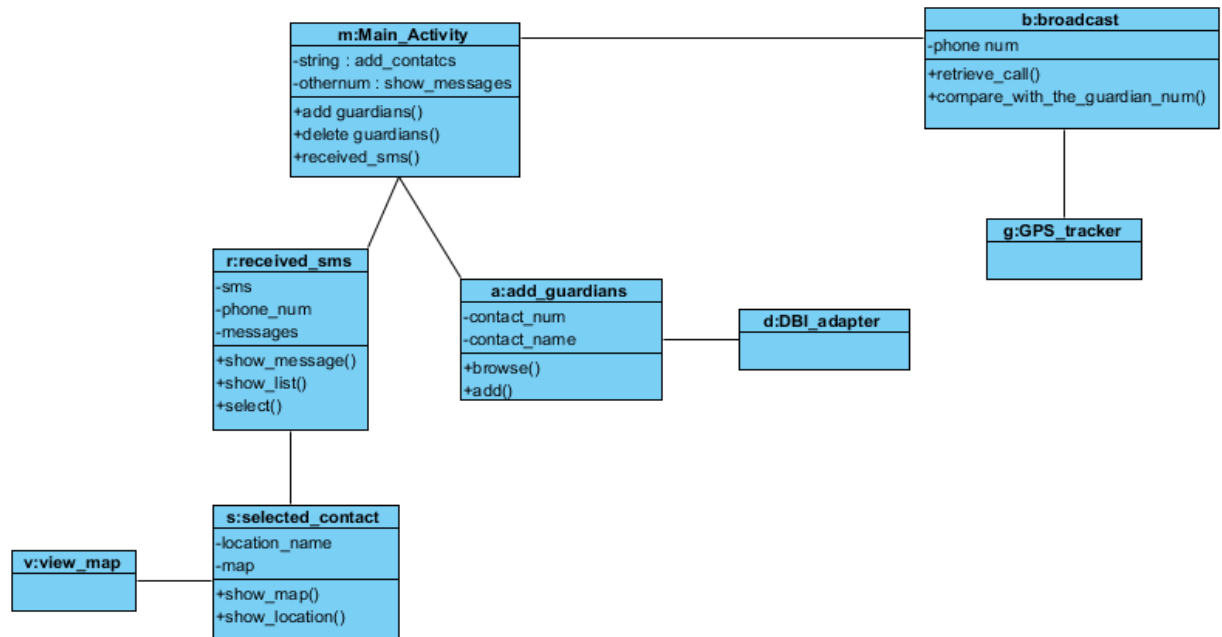


Fig 5.3 Class Diagram For Andriod Security Application

5.5 SEQUENCE DIAGRAM:

Sequence diagrams document the interactions between classes to achieve a result, such as a use case. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.

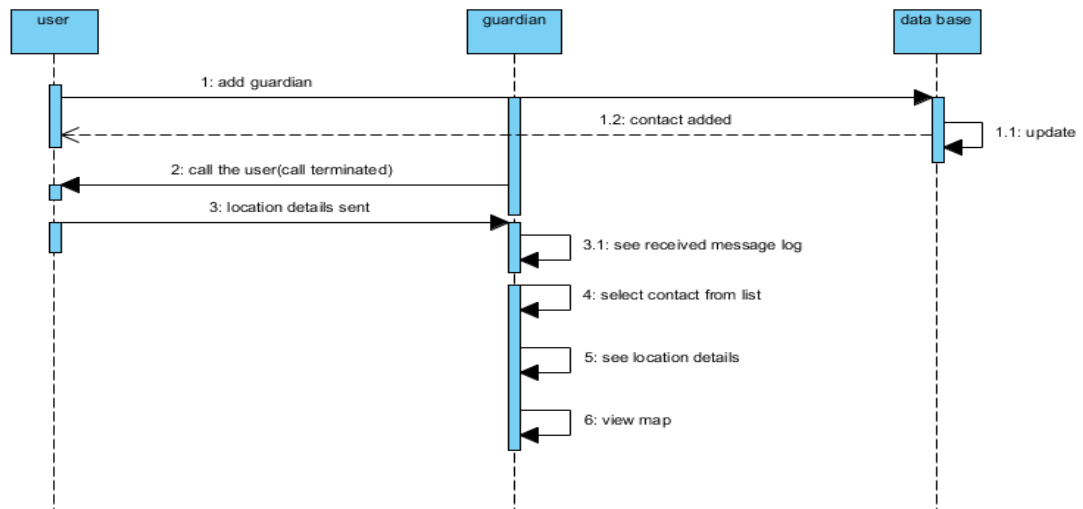


Fig 5.4 Sequence Diagram For Andriod Security Application

5.6 ACTIVITY DIAGRAM:

Activity diagrams are used to document workflows in a system, from the business level down to the operational level. The general purpose of Activity diagrams is to focus on flows driven by internal processing vs. external events.

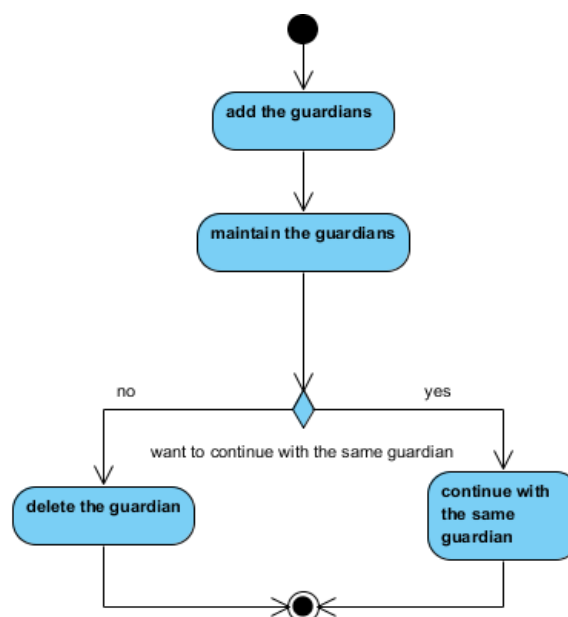


Fig 5.5 Activity Diagram For Andriod Security Application

5.7 COLLABORATION DIAGRAM:

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).

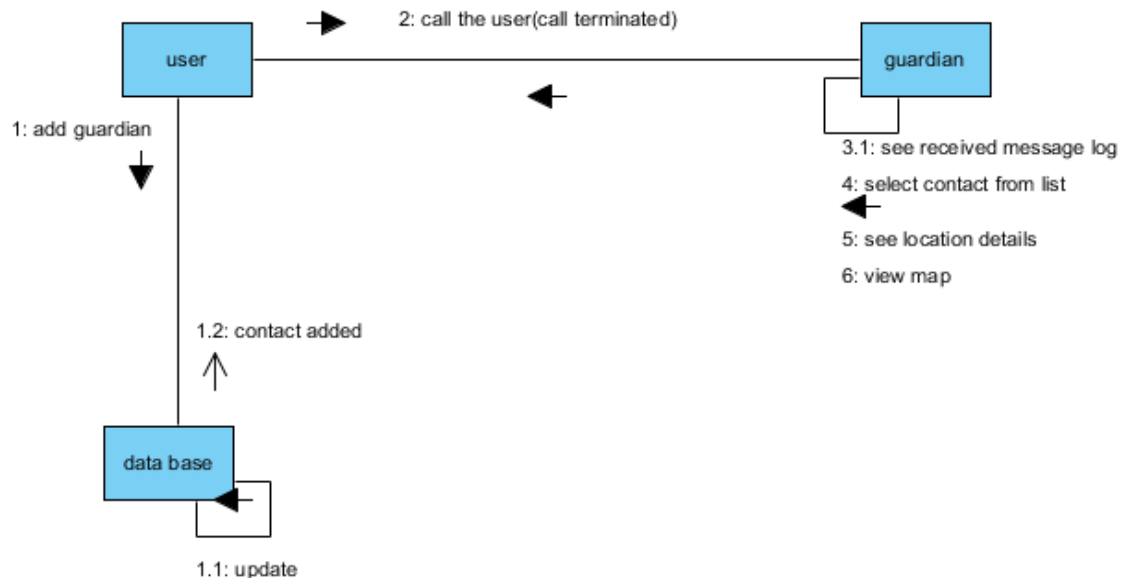


Fig 5.6 Collaboration Diagram For Security Application

5.8 OBJECT DIAGRAM :

An object diagram in the Unified Modeling Language, is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.

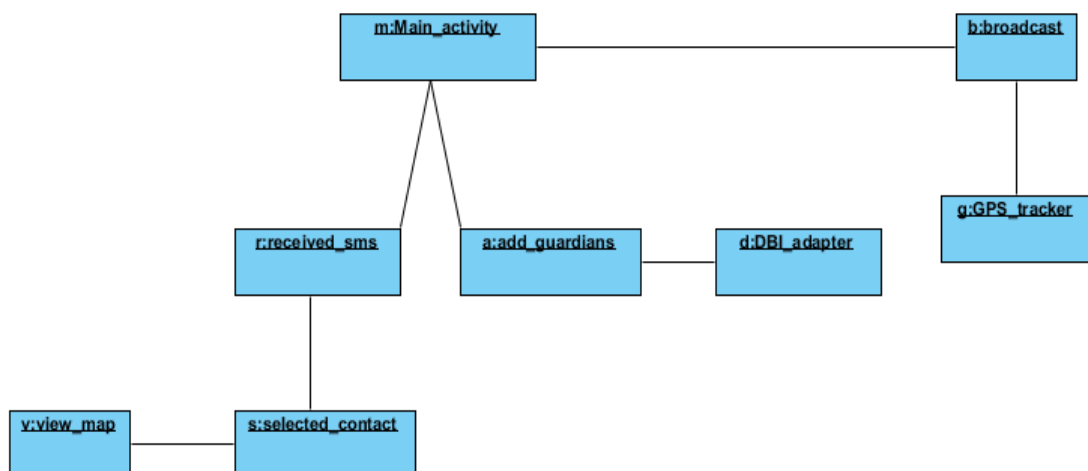


Fig 5.7 Object Diagram For Android Security Application

5.9 STATE CHART DIAGRAM:

State chart diagram is one of the five UML diagrams used to model dynamic nature of a system. They define different states of an object during its lifetime. And these states are changed by events. So State chart diagrams are useful to model reactive systems.

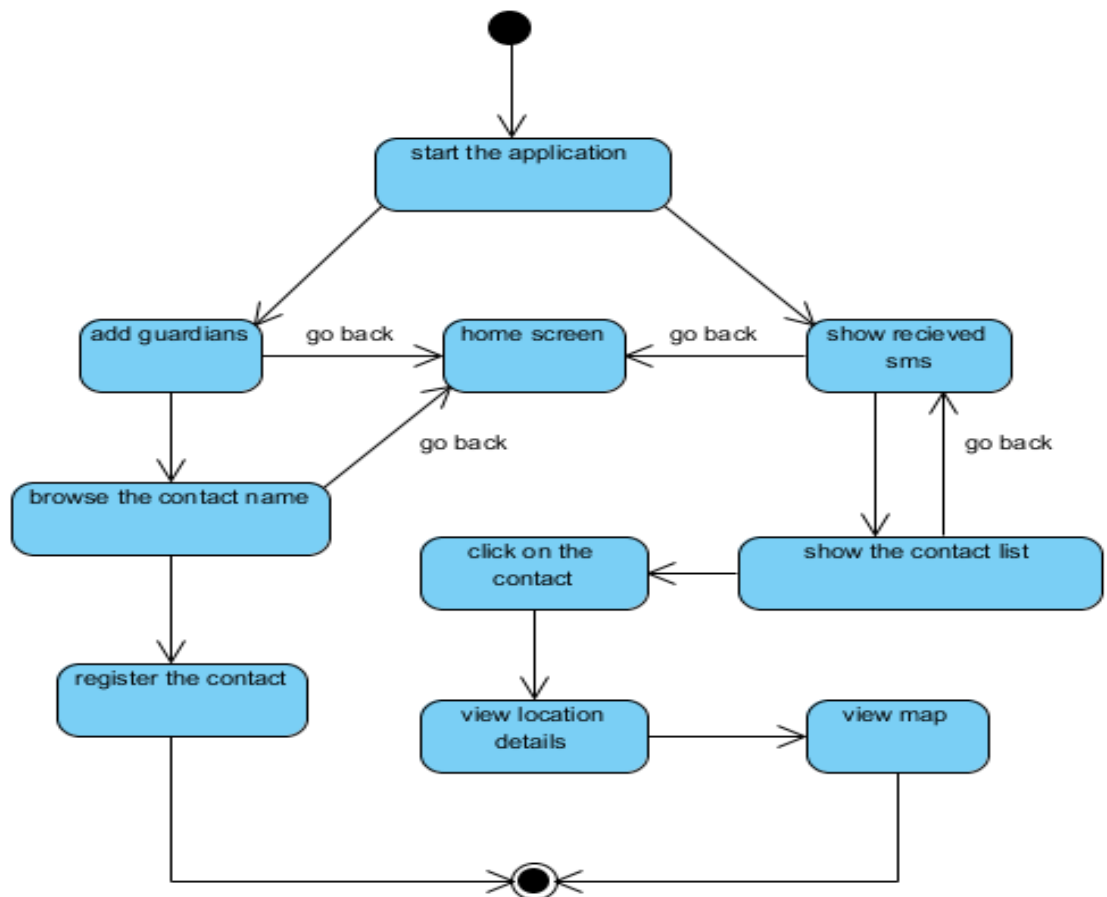


Fig 5.8 State Chart Diagram For Android Security Application

5.10 COMPONENT DIAGRAM:

A component diagram depicts how components are wired together to form larger components and or software systems.

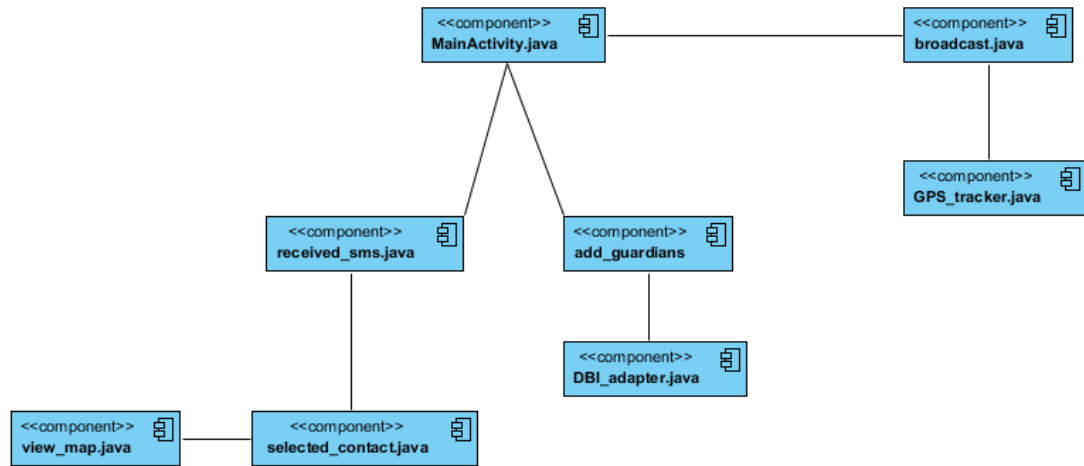


Fig 5.9 Component Diagram Of Android Security Application

5.11 DEPLOYMENT DIAGRAM:

Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed.

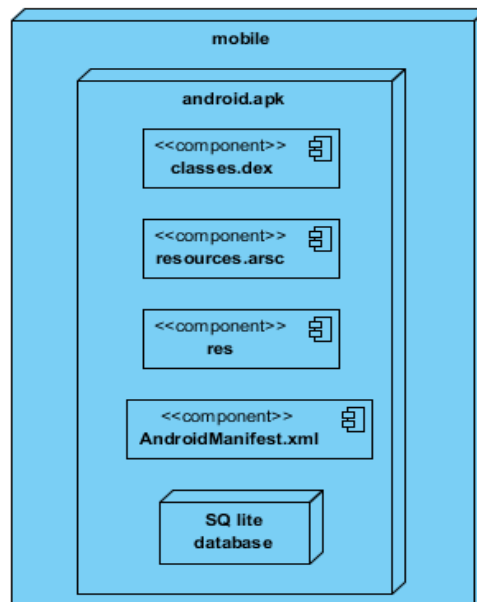


Fig 5.10 Deployment Diagram for Android Security Application

6. DETAILED DESCRIPTIONS OF COMPONENTS

Service:

To send the messages to the guardians we use the network provider and to calculate the latitude and longitude, we use GPS providers. This component checks whether the mobile is in the network area of service provider to send or receive the messages. To send the messages the user should have his GPs system enabled. If GPS is enabled then, the values can be sent otherwise they cannot be sent.

Content Provider:

Content providers manage access to a structured set of data. They encapsulate the data, and provide mechanisms for defining data security. Content providers are the standard interface that connects data in one process with code running in another process. When you want to access data in a content provider, you use the ContentResolver object in your application's Context to communicate with the provider as a client. The ContentResolver object communicates with the provider object, an instance of a class that implements ContentProvider. The provider object receives data requests from clients, performs the requested action, and returns the results.

Google is a web mapping service application and technology provided by Google, that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. Once we get the location details of the user we can see that location in the map using Google maps.

CODE:

Gps Tracker Main Activity:

```
package com.coign.security;

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.os.Looper;

import android.util.Log;

import android.widget.Toast;

public class GPSTracker extends Service implements LocationListener {

    private final Context mContext;

    boolean isGPSEnabled = false;

    boolean isNetworkEnabled = false;

    boolean canGetLocation = false

    Location location; // location

    public double latitude; // latitude

    public double longitude; // longitude
```

```
private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 0; // 10
meters
```

```
private static final long MIN_TIME_BW_UPDATES = 0; // 1 minute
```

```
protected LocationManager locationManager;
```

```
public GPSTracker(Context context) {
```

```
    this.mContext = context;
```

```
    getLocation(); }
```

```
public Location getLocation() {
```

```
    try { locationManager = (LocationManager) mContext
```

```
        .getSystemService(LOCATION_SERVICE);
```

```
    isGPSEnabled = locationManager
```

```
        .isProviderEnabled(LocationManager.GPS_PROVIDER);
```

```
    isNetworkEnabled = locationManager
```

```
        .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
```

```
    if (!isGPSEnabled && !isNetworkEnabled) {
```

```
    } 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);

        onLocationChanged(location); }}

    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);
                onLocationChanged(location);

            } } } } catch (Exception e) { e.printStackTrace(); }

return location; }

public void stopUsingGPS(){

    if(locationManager != null){

        locationManager.removeUpdates(GPSTracker.this);

    }

}

public double getLatitude(){

    if(location != null){

        latitude = location.getLatitude(); }

    return latitude;

}

```

```

public double getLongitude(){
    if(location != null){
        longitude = location.getLongitude();
    }
    return longitude;}

public boolean canGetLocation() {
    return this.canGetLocation;
}

public void showSettingsAlert(){
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(mContext);
    alertDialog.setTitle("GPS is settings");
    alertDialog.setMessage("GPS is not enabled. Do you want to go to settings menu?")
    alertDialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener()
    {
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    alertDialog.show(); }

@Override
public void onLocationChanged(Location location) {
    if (location != null) {
        stopUsingGPS();
        latitude = location.getLatitude();
        longitude = location.getLongitude();
    }
}

```

```
}}
```

```
@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;}  
  

```

Lat & long Main Activity:

```
package com.coign.security;
```

```
public class LatLong1Activity extends ListActivity {  
  
    ArrayList<String> arraylist = new ArrayList<String>();  
  
    SQLiteDatabase db;  
  
    MyDBHelper helper;  
  
    ListView listView;  
  
    ArrayAdapter<String> cadapter;  
  
    String lvmessageitem;  
  
    String[] aray, aray1, aray2, aray3, aray4, ar;  
  
    String lati, longi;  
  

```

```

        Double latitude, longitude;

        final Context context = this;

        Cursor c;

        String number,name;

        String adres;

        @SuppressWarnings("deprecation")

        @Override

        protected void onCreate(Bundle savedInstanceState) {

            super.onCreate(savedInstanceState);

            requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);

            setContentView(R.layout.latlong);

getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.title);


            listView = (ListView) findViewById(android.R.id.list);

            helper = new MyDBHelper(this);

            db = helper.getWritableDatabase();

            Bundle b = getIntent().getExtras();

            number = b.getString("sms_number");

            name=b.getString("sms_name");

            final TextView tv=(TextView)findViewById(R.id.Title);

            if ( tv != null ) {

tv.setText(name);    }

            System.out.println("long lati Activity number" + number);

            try {

                c = db.rawQuery(

```

```
"select _id,snumber,sname,smsmessage,dat from smsdetails where snumber='"+  
number + "' order by dat desc", null);
```

```
if(c!=null)
```

```
{ if(c.moveToFirst())
```

```
{ do
```

```
{
```

```
String s = c.getString(c.getColumnIndex("smsmessage"));
```

```
String date = c.getString(c.getColumnIndex("dat"));
```

```
array = s.split(",");
```

```
System.out.println(array);
```

```
array1 = array[0].split(":");
```

```
array2 = array[1].split(":");
```

```
lati = array1[array1.length - 1];
```

```
longi = array2[array2.length - 1];
```

```
latitude=Double.parseDouble(lati);
```

```
longitude=Double.parseDouble(longi);
```

```
adres=getCompleteAddressString(latitude,longitude);
```

```
arraylist.add(adres+date);
```

```
} while (c.moveToNext());}
```

```
cadapter=new
```

```
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1,arraylist);
```

```
listView.setAdapter(cadapter); }
```

```
if(adres.contentEquals("")) {
```

```
Toast.makeText(getApplicationContext(), "No Network Connection", 90).show();
```

```
}
```

```
else
```

```
{
```

```
listView.setOnItemClickListener(new OnItemClickListener() {
```



```

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    Intent i = new Intent(LatLng1Activity.this,
        SmsMapdisplayActivity.class);

    i.putExtra("listview_latitude", lati);

    i.putExtra("listview_longitude", longi);

    startActivity(i);

    });}}

    catch (Exception e) {

        e.printStackTrace();

    }}

    private String getCompleteAddressString(double LATITUDE, double LONGITUDE)
    {

        String strAdd = "";

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

        try {

            List<Address> addresses = geocoder.getFromLocation(LATITUDE,
                LONGITUDE, 1);

            if (addresses != null) {

                Address returnedAddress = addresses.get(0);

                StringBuilder strReturnedAddress = new StringBuilder("");

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

                    strReturnedAddress.append(returnedAddress.getAddressLine(i)).append("\n");

                }

                strAdd = strReturnedAddress.toString();

                Log.w("My Current loction address", "" + strReturnedAddress.toString());

            } else {Log.w("My Current loction address", "No Address returned!");} }

        catch (Exception e) {

```

```

        e.printStackTrace();

        Log.w("My Current loction address", "Canont get Address!"); }

return strAdd; }

public void backbutton(View v) {

        finish();    }

public void refresh()

{

        c.requery();

        cadapter.notifyDataSetChanged(); }

public boolean onKeyDown(int keyCode, KeyEvent event) {

        if (keyCode == KeyEvent.KEYCODE_BACK

                || keyCode == KeyEvent.KEYCODE_HOME) {

                exitByBackKey();

                return true;    }

        return super.onKeyDown(keyCode, event); }

protected void exitByBackKey() {

        AlertDialog alertbox = new AlertDialog.Builder(this)

                .setMessage("Do you want to exit application?")

                .setPositiveButton("Yes",new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {

Intent intent = new Intent(Intent.ACTION_MAIN);

intent.addCategory(Intent.CATEGORY_HOME);

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

```

```

startActivity(intent);

finish();}})

.setNegativeButton("No", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {}}).show();}

protected void onDestroy() {

    super.onDestroy();

    db.close();    }    }

```

Location Main Activity:

```

package com.coign.security;

public class LocationService implements LocationListener {

private final Context mContext;

boolean isGPSEnabled = false;

boolean isNetworkEnabled = false;

boolean canGetLocation = false;

final static long MIN_TIME_INTERVAL = 60 * 1000L;

Location location;

private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 0; // 10

private static final long MIN_TIME_BW_UPDATES = 1; // 1 minute

protected LocationManager locationManager;

private CountdownTimer timer = new CountdownTimer(5 * 1000, 1000) {

    public void onTick(long millisUntilFinished) {}

    public void onFinish() {

        stopUsingGPS();

    }
}

```

```

};

public LocationService(Context context) {

    this.mContext = context;

    start();

}

public void start() {

    try {

        timer.start();

        locationManager = (LocationManager)mContext

            .getSystemService(Context.LOCATION_SERVICE);

        isGPSEnabled = locationManager

            .isProviderEnabled(LocationManager.GPS_PROVIDER);

        isNetworkEnabled = locationManager

            .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

        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 tempLocation = locationManager

                    .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

                if (tempLocation != null

                    && isBetterLocation(tempLocation,

```

```

        location))

        location = tempLocation;    } }

if (isGPSEnabled) {

    locationManager.requestSingleUpdate(

        locationManager.GPS_PROVIDER, this, 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 tempLocation = locationManager

            .getLastKnownLocation(LocationManager.GPS_PROVIDER);

        if (tempLocation != null

            && isBetterLocation(tempLocation,

                location))

            location = tempLocation;    }    }    } catch (Exception e) {

        e.printStackTrace();

    } }

public void stopUsingGPS() {

    if (locationManager != null) {

        locationManager.removeUpdates(LocationService.this);    } }

public boolean canGetLocation() {

    locationManager = (LocationManager)mContext

        .getSystemService(Context.LOCATION_SERVICE);

    isGPSEnabled = locationManager

```

```

        .isProviderEnabled(LocationManager.GPS_PROVIDER);

isNetworkEnabled = locationManager

        .isProviderEnabled(LocationManager.NETWORK_PROVIDER);

return isGPSEnabled || isNetworkEnabled;

}

@Override

public void onLocationChanged(Location location) {

    if (location != null

        && isBetterLocation(location, this.location)) {

        this.location = location;    }    }

@Override

public void onProviderDisabled(String provider) {    }

@Override

public void onProviderEnabled(String provider) {    }

@Override

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

public static boolean isBetterLocation(Location location,

    Location currentBestLocation) {

    if (currentBestLocation == null) {

        return true;    }

    long timeDelta = location.getTime() - currentBestLocation.getTime();

    boolean isSignificantlyNewer = timeDelta > MIN_TIME_INTERVAL;

    boolean isSignificantlyOlder = timeDelta < -MIN_TIME_INTERVAL;

    boolean isNewer = timeDelta > 0;

    if (isSignificantlyNewer) {

```

```

        return true;

    } else if (isSignificantlyOlder) {

        return false;

    }

    int accuracyDelta = (int) (location.getAccuracy() - currentBestLocation

        .getAccuracy());

    boolean isLessAccurate = accuracyDelta > 0;

    boolean isMoreAccurate = accuracyDelta < 0;

    boolean isSignificantlyLessAccurate = accuracyDelta > 200;

    boolean isFromSameProvider = isSameProvider(location.getProvider(),

        currentBestLocation.getProvider());

    if (isMoreAccurate) {

        return true;

    } else if (isNewer && !isLessAccurate) {

        return true;

    } else if (isNewer && !isSignificantlyLessAccurate

        && isFromSameProvider) {

        return true;    }

    return false;    }

    private static boolean isSameProvider(String provider1, String provider2) {

        if (provider1 == null) {

            return provider2 == null;    }

        return provider1.equals(provider2);    }}

```

Received SMS Details:

```
package com.coign.security;

public class RecievedSmsDetailsActivity extends ListActivity {

    SQLiteDatabase db;

    MyDBHelper helper;

    SimpleCursorAdapter cadapter;

    ListView listView;

    String number,name;

    Cursor cursor;

    final Context context = this;

    @SuppressWarnings("deprecation")

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);

        setContentView(R.layout.recievedsmsdetails);

        getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE,
R.layout.recievedsmstitlebar);

        listView = (ListView) findViewById(android.R.id.list);

        helper = new MyDBHelper(this);

        db = helper.getWritableDatabase();

        Cursor cursor = db.query("smsdetails", new String[] {
            "_id","snumber", "sname","smsmessage", "dat" }, null, null, "sname", null, null);

        startManagingCursor(cursor);

        if(cursor.getCount()==0){

            Toast.makeText(getApplicationContext(), "No messages",
```



```

Toast.LENGTH_LONG).show();    }

else{

    cadapter = new SimpleCursorAdapter(this, R.layout.mylist, cursor,
    new String[] { "sname", "snumber" }, new int[] { R.id.text1,
    R.id.text2 });

    this.setAdapter(cadapter);    }

    listView.setOnItemClickListener(new OnItemClickListener() {

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    Cursor cursor = (Cursor) parent.getItemAtPosition(position);

    number=cursor.getString(cursor.getColumnIndex("snumber"));

    name=cursor.getString(cursor.getColumnIndex("sname"));

    Intent i = new Intent(RecievedSmsDetailsActivity.this,LatLong1Activity.class);

    i.putExtra("sms_number", number);

    i.putExtra("sms_name",name);

    startActivity(i);    } });

    listView.setOnItemLongClickListener(new OnItemLongClickListener() {

public boolean onItemLongClick(AdapterView<?> parent,View view, int position,
long id) {

    Cursor cursor = (Cursor) parent.getItemAtPosition(position);

    final int item_id = cursor.getInt(cursor.getColumnIndex("_id"));

    final String num=cursor.getString(cursor.getColumnIndex("snumber"));

    AlertDialog.Builder myDialog = new AlertDialog.Builder(context);

    myDialog.setTitle("Delete?");

    myDialog.setPositiveButton("OK",

    new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0,int arg1) {

```

```

.delete("smsdetails", "_id=" +item_id, null);

Toast.makeText(getApplicationContext(),"contact deleted",
Toast.LENGTH_LONG).show();

refresh();});

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

public void onClick(DialogInterface dialog, int which) {    }});

myDialog.show();

return true;    });

}

public void refresh()

{

Cursor cursor = dbquery("smsdetails", new String[] { "_id", "snumber", "sname",
"smsmessage", "dat" }, null, null, "sname", null, null);

startManagingCursor(cursor);

cadapter = new SimpleCursorAdapter(this, R.layout.mylist, cursor,
new String[] { "sname", "snumber" }, new int[] { R.id.text1, R.id.text2 });

this.setAdapter(cadapter);

cadapter.notifyDataSetChanged();

public void backbutton(View v) {finish();}

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE_BACK || keyCode
==KeyEvent.KEYCODE_HOME) {

exitByBackKey();

return true; }

return super.onKeyDown(keyCode, event);}

protected void exitByBackKey() {

```

```

AlertDialog alertbox = new AlertDialog.Builder(this)

.setMessage("Do you want to exit application?")

.setPositiveButton("Yes",new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {

Intent intent = new Intent(Intent.ACTION_MAIN);

intent.addCategory(Intent.CATEGORY_HOME);

intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

startActivity(intent);

finish();

}}).setNegativeButton("No", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {}}).show();

protected void onDestroy() {super.onDestroy();

db.close();}}

```

Show Map Activity:

```

package com.coign.security;

public class ShowMapActivity extends Activity {

double latitude, longitude;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.smsmapdetails);

TextView tvdate = (TextView) findViewById(R.id.date);

TextView tvnumber = (TextView) findViewById(R.id.number);

```

```

TextView tvlocation = (TextView) findViewById(R.id.location);

String date = getIntent().getStringExtra("SMS_DATE");

String location = getIntent().getStringExtra("SMS_BODY");

String number = getIntent().getStringExtra("SMS_NUMBER");

String lati = getIntent().getStringExtra("sms_latitude");

String longi = getIntent().getStringExtra("sms_longitude");

latitude = Double.parseDouble(lati);

longitude = Double.parseDouble(longi);

tvnumber.setText("Phone no: " + number);

tvlocation.setText("Location: " + getCompleteAddressString(latitude,longitude));

tvdate.setText("Called on: "+date); }

public void mapbuttonclick(View v) {

String name = "my location";

Intent mapintent = new Intent(android.content.Intent.ACTION_VIEW,

Uri.parse("geo:0,0?q=" + latitude + "," + longitude + " (" + name + ")"));

mapintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

this.startActivity(mapintent);

}

private String getCompleteAddressString(double LATITUDE, double LONGITUDE)
{

String strAdd = "";

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

try {

List<Address> addresses = geocoder.getFromLocation(LATITUDE,
LONGITUDE, 1);

if (addresses != null) {

```

```

        Address returnedAddress = addresses.get(0);

        StringBuilder strReturnedAddress = new StringBuilder("");

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

            strReturnedAddress.append(returnedAddress.getAddressLine(i)).append("\n");

        }

        strAdd = strReturnedAddress.toString();

        Log.w("My Current loction address", "" + strReturnedAddress.toString());

    }

else {

    Log.w("My Current loction address", "No Address returned!");

}

} catch (Exception e) {

    e.printStackTrace();

    Log.w("My Current loction address", "Canont get Address!"); }

return strAdd;    }    }

```

SMS Map Display:

```

package com.coign.security;

public class SmsMapdisplayActivity extends Activity {

    Double latitude,longitude;

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.recievedsmsmap);

        Bundle b=getIntent().getExtras();

        String latitudeval=b.getString("listview_latitude");

        String longitudeval=b.getString("listview_longitude");

```

```

latitude=Double.parseDouble(latitudeval);

longitude=Double.parseDouble(longitudeval);

String name="my location";

Intent mapintent = new Intent(android.content.Intent.ACTION_VIEW,
Uri.parse("geo:0,0?q="+latitude+", "+longitude+" (" + name + ")"));

mapintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

startActivity(mapintent);    }

public boolean onKeyDown(int keyCode, KeyEvent event) {

    if (keyCode == KeyEvent.KEYCODE_BACK

        || keyCode == KeyEvent.KEYCODE_HOME) {

        exitByBackKey();

return true;    }

return super.onKeyDown(keyCode, event); }

protected void exitByBackKey() {

AlertDialog alertbox = new AlertDialog.Builder(this).setMessage("Do you want to
exit application?").setPositiveButton("Yes",new DialogInterface.OnClickListener() {

public void onClick(DialogInterface arg0, int arg1) {

finish();        })).setNegativeButton("No", new DialogInterface.OnClickListener()

{ public void onClick(DialogInterface arg0, int arg1) {} }).show();}

}

```

7. TESTING

TEST CASE DESIGN:

White box testing:

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independent paths in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

Black Box Testing:

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

The following are the different tests at various levels:

Unit Testing:

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is to test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

Integration Testing:

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

Validation Testing:

This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

System Testing:

This testing is a series of different tests whose primary is to fully exercise the computer-based system. This involves:

- Implementing the system in a simulated production environment and testing it.
- Introducing errors and testing for error handling.

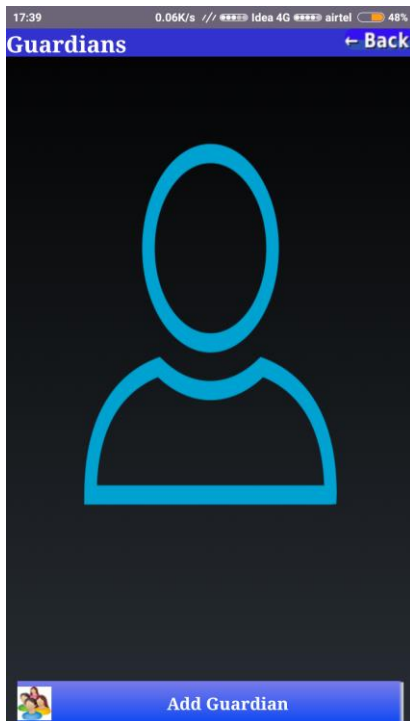
Test cases:

Test Case	Check Item	Test case Objective	Test Data / Input	Expected Result	Actual Result	Results
TC-001	Add Guardian	Checking if guardian is added		some contact will be added	Guardian is added successfully	PASS
TC-002	Country code	Checking if the country code is included		Contact is added which has country code	No contact is added since it does not have country code.	FAIL
TC-003	Contact number	Enter contact number	enter a number	It should allow the user to add the number to the guardian list	Toast message "Contact name saved successfully"	PASS
TC-004	SMS	checking if sms has been received.	switch off wifi or mobile data.	sms to be received	No sms is received as there is no network connection	FAIL
TC-005	GPS	Checking wether the GPS is enabled or not.	switch off GPS.	Latitude and Longitude details are received	No details received as GPS is not enabled.	FAIL
TC-006	SMS	Checking if sms has been sent	latitude, longitude	sms to be sent	Toast message "SMS Sent!"	PASS
	Missedcalls	checking missed calls		It should show the number	Toast message "You have a missed call"	PASS

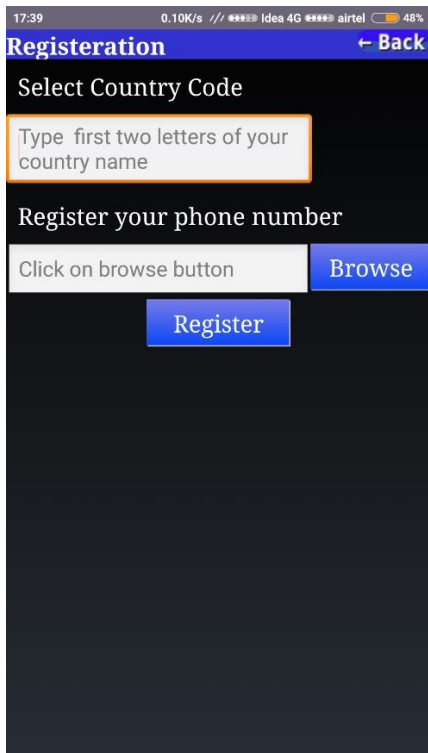
Fig 7.1 Test Cases

8. SCREENSHOTS

8.1 HOME SCREEN



8.2 REGISTRATION



8.3 ADDING PHONE NUMBERS

17:38 2.55K/s // 4G airtel 48%

Registration ← Back

Select Country Code

India (+91)

Register your phone number

+919701462345 Browse

Register

8.4 COUNTRY CODE

17:36 0.83K/s // 4G airtel 49%

Registration ← Back

Select Country Code

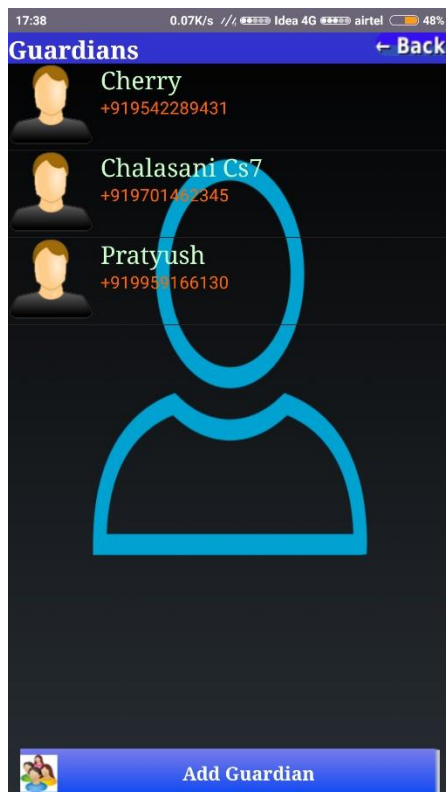
India (+91)

Register your phone number

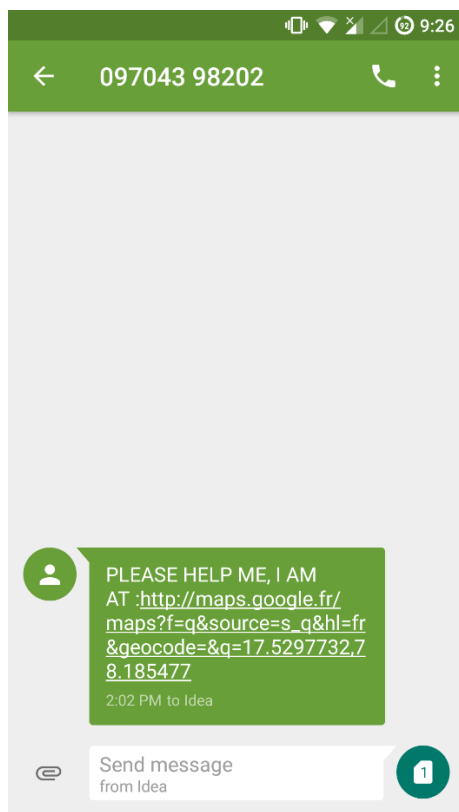
Click on browse button Browse

Register

8.5 GUARDIANS LIST



8.6 SMS FORMAT



8.7 MAP LOCATION



9. CONCLUSION

9.1 CONCLUSION:

This application is mainly used for security purpose. It gives us the details of the location of a person when he does not lift the phone using the GPS system. This application has to be installed in both the phones i.e. the user as well as the guardian. At first the user has to add at least one guardian to his application so that the details will be sent to him. Now, if the guardian calls and the user does not lift the phone the latitude and longitude of the user are calculated based on the GPS system enabled in his mobile and this SMS is sent to the guardians. Then internally these latitude and longitude values are taken and the address is calculated by the application and the address will be displayed in the received SMS list. Then by clicking on this address we can see the map using Google maps.

9.2 FUTURE SCOPE:

Moving to the future scope, we can provide additional features like sending the location messages free of cost and other features like in which there will be a flexibility of using the application only on one mobile instead of installing on every mobile.

REFERENCES

TEXT BOOK:

1. Java Beginners Guide – Herbert Schildt
2. Android Programming Pushing – Erik Hellman

LINKS:

1. <https://developer.android.com/studio/intro/>
2. <https://developer.android.com/reference/android/location/LocationManager>
3. https://www.tutorialspoint.com/android/android_sending_sms.htm