**CHAPTER-1**

**INTRODUCTION**

Ban Ki-Moon, the secretary general of United Nations stated that “There is one universal truth, applicable to all countries, cultures and communities: violence against women is never acceptable, never excusable, and never tolerable”. Violence against women is a significant public health problem, as well as a fundamental violation of women’s human rights. There are three reasons why mobile technology will reduce violence against women in public places. They are easily accessible, crowd sourcing and affordable scalability.An “app” is a small, specialized software program, easily downloadable and installed onto mobile devices such as Smartphone’s or tablet computers. The use of apps has been popularized by the Apple’sApp Store and also by Google’s Play Store.

The capital city New Delhi in 2012 arrested the attention of not only the people of the nation but also the entire world. A rape incident captured the attention of the entire human kind that occurred on 16th December 2012 at a place Munirka, a neighborhood in south Delhi which was a fatal assault. 23 year old woman physiotherapy (intern) was hit and molested by a gang at 9:30 PM when she was travelling in a private bus with a male counterpart. They were returning after watching the film Life of Pi in saket, south Delhi and boarded an of duty charter bus at Munirka to Dwaraka, which was driven by joy riders at that time. The family members and colleagues of TCS software engineer Esther Anuhaya found her body with the help of a Vijayawada police team. Her parents spent the entire Thursday looking for her in Bhandup (East) as her last call signal on January 5th was from BhandupeshwarKund in Kanjurmarg, which falls under Bhandup (East) jurisdiction. The family had been trying to trace her where about by showing the locals her photographs. Locals said that the spot where her body was found is a hangout for criminals. The body of Anuhaya has been procured by Vijayawada police. A recent article in India claimed that India is the fourth most dangerous place for women’s to take public transport and the second worst for safety while travelling at night.

Another 19-year old Pharmacy student named Ayesha Miran was raped and murdered brutally by 22- year old P. Satyam Babu in a hostel at Vijayawada. A body with stab injuries was found in the bathroom on Dec 27, 2007. A letter dropped by the 'murderer' stated that the girl was raped and murdered for refusing his request for 'love'

This is the small contribution taken which will provide safety android app for women. An Feminine Security using Android Application is an android application which helps women/teenagers to protect themselves by using a Smartphone. It lets your family and friends know your current via GPS tracker if your android device is connected to the network.

**CHAPTER-2**

**AIM & SCOPE**

The project IDE is Android studio and is multiplatform target device. This software works on different platforms with minimum hardware and software requirements as stated.

We describe what features are in the scope of the software and what are not in the scope of the software to be developed.

**In Scope:**

1. It is a daemon application.

2. Information regarding the registered contacts and so on.

3. Giving alerts when a panic button is initiated.

4. The location of the victim and an alert message is sent to registered contacts.

5. User authentication.

6. Also provides a module for changing registered contacts.

**Out Scope:**

1. Never sends an alert message without initiating a panic button.
2. User cannot register with the victim without proper authentication.

**2.1 EXISTING SYSTEM**

The existing system is an android application which contains panic button inside the app. If the victim recognizes that she is in danger, then the victim should open the application should type the alert message and then click the panic button which is available only in the app.

**2.2 PROPOSED SYSTEM**

The proposed system is an android application which contains the panic button on the mobile itself [power button] of the mobile. Initially the victim should register the contacts of near and dear. Whenever the victim is in danger then she should initiate the panic button for 3 times. On initiating the panic button the current address of the victim along with an alert message is sent to the registered contacts.

**2.3 FEASIBILITY STUDY**

Preliminary investigation examines project feasibility; the likelihood the system will beuseful to the organization. The main objective of the feasibility study is to test the Technical,Economical and Social feasibility for adding new modules and debugging old runningsystem. All systems are feasible if they are given unlimited resources and infinite time. Thereare aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Economic Feasibility
* Social Feasibility

2.3.1 TECHNICAL FEASIBILITY

The android app technically possible to implement with little amount of software andhardware costs. The technologies required for this sort of solutions are available in recent days like android phones. We can use this FEMININE SECURITY USING ANDROID APPLICATION in any android phones. This mobile app is feasible to every android phone.

2.3.2 ECONOMIC FEASIBILITY

The developed system is within the budget and this was achieved because most of thetechnologies used are freely available. This Feminine Security Using Android Application is economically feasible because open source technologies like Android SDK and/ SQLite are used in developing this app. A user will not be burdened to use this app.

2.3.3 SOCIAL FEASIBILITY

Feminine Security Using Android Application is developed for Android mobiles, as Android is exploiting most of the mobile market. Many users own Android mobiles these days, than other mobiles such as Windows and IOS. In this regard, Feminine Security Using Android Application app is socially feasible. As now we are providing this app only in English language. In future this app will bedeveloped in different languages.

**CHAPTER-3**

**CONCEPTS & METHODS**

**3.1 PROBLEM DESCRIPTION**

* The drawback of existing system is when emergency situation occurs the user cannot set the alert immediately.
* The existing system contains panic button with in the app itself.
* Whenever the victim is in danger, she should login to the app and press the panic button.

**3.2 PROPOSED SOLUTION**

* Unlike other apps, this app works only when the victim is in danger.
* The proposed system contains panic button on the mobile panel i.e., power button of the mobile.
* If the victim initiates panic button for 3times the address of the victim is send to the registered contacts in the form of SMS.

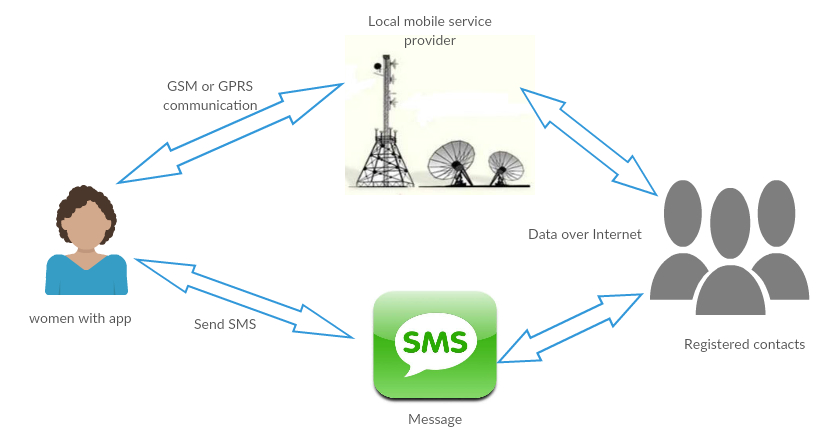
**3.3 SYSTEM ANALYSIS METHODS**

3.3.1 ARCHITECTURE DIAGRAM

Women referred in the diagram represent the victim. Location of the victim is received through GPS.

In case if GPS in the mobile is turned off, it receives location from the local network provider.

After the victim initializing a panic button, the location of the victim along with an alert message is sent to the registered contacts. GSM will send the message to the registered contacts.

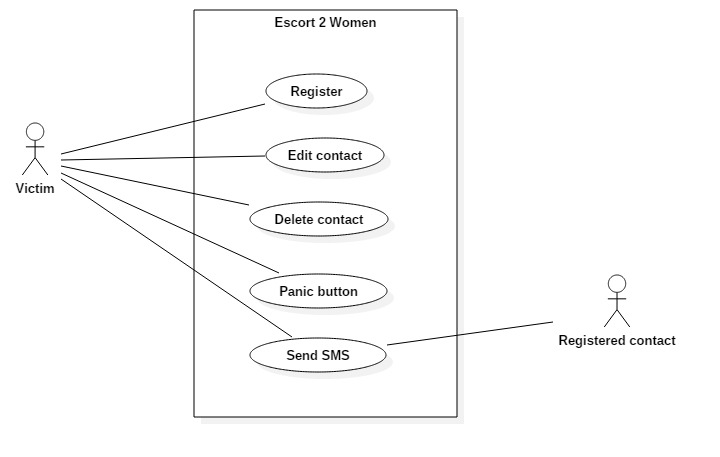
****

**Fig 3.1 Architecture diagram**

Panic button is provided for emergency alert. When Panic button is pressed GSM will urgently send alert message.

3.3.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

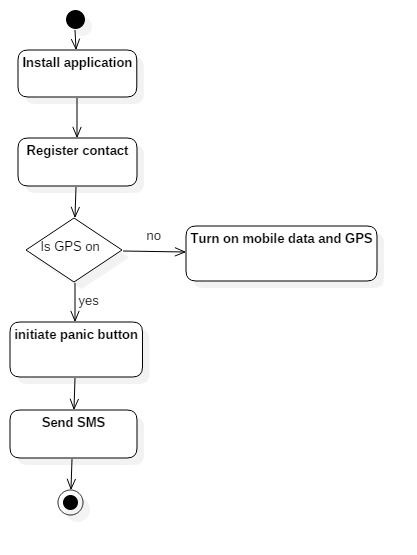


**Fig 3.2 Use Case diagram**

There are different types of operations like Register, Edit contact, Delete contact, Panic button, Send SMS. Victim initially registers some contacts. If he wishes to change the contact then he can edit the contacts. If he wants to delete the contact, then he can also delete the contact. Victim is only responsible for initiating the panic button. After initiating the panic button the system is responsible for sending the SMS. Registered contact can only access the message sent by the victim. Registered contact does have permission to initiate panic button or modify any contacts.

3.3.2 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by- step workflows of components in a system. An activity diagram shows the overall flow of control.



**Fig 3.3 Activity diagram**

After installing the application the victim is asked to register some contacts of his/her friends, family or trusted persons. GPS and mobile data should be always turned on in order to send the SMS. After turning on mobile data and GPS and victim initializes panic button then an alert message along with the current address is sent in the form of SMS to the initially registered contacts.

* 1. **SYSTEM REQUIREMENTS**

## *Hardware Requirements*

No external hardware is involved other than android based mobile phone.

*Software Requirements*

* Android studio 2.3
* SDK version – 26
* Minimum SDK version – 19
* JDK version - jdk1.8.0\_101
* Database- SQLite

**3.5 SYSTEM FEATURES**

## *3.5.1 Change Contacts*

3.5.1.1 *Description and Priority*

This feature describes about the changing the selected contacts.

3.5.1.2 *Stimulus/Response Sequences*

1. If the user wants to change the registered contacts then the user must initiate change button.

2. User should initiate the change button beside the contact he wants to change.

3. Then it will redirect to the contacts pagein user mobile.

3.5.1.3 *Functional Requirements*

*3.5.1.3.1 Introduction:*

If the user wishes to change the initially registered contacts then he must initiate the change button.

*3.5.1.3.2 Input:*

This application takes the input from only the list of contacts available in the mobile.

*3.5.1.3.3 Processing:*

* User should initiate change button beside the contact he want to change.
* Then it redirects to the contacts page.
* User should select the required contact.

*3.5.1.3.4 Output:*

It replaces the already existing contact with the new contact selected.

## *3.5.2 Sending SMS*

*3.5.1.1 Description and Priority*

This feature describes about sending a default message to the selected contacts whenever a panic button is pressed.

*3.5.1.2 Stimulus/Response Sequences*

1. As soon as user initiates panic button it sends a default message to the selected contacts.

2. A default message may be like “I am in danger, please save me”.

*3.5.1.3 Functional Requirements*

*3.5.1.3.1 Introduction:*

In order to send an SMS to the selected contacts, when the victim is in danger.

*3.5.1.3.2 Input:*

Victim must initiate panic button continuously for 4 times in order to send an SMS.

*3.5.1.3.3 Processing:*

* Victim should be aware that he/she is in danger.
* Then victim should initiate the panic button.
* Then it sends a default message to the selected contacts.
* A default message be like “I am in danger, please help me”.

*3.5.1.3.4 Output:*

A message is sent to the registered contacts.

## *3.5.3 Sending Location*

*3.5.3.1 Description and Priority*

This feature describes about sending location to the selected contacts whenever a panic button is pressed.

*3.5.3.2 Stimulus/Response Sequences*

1. As soon as user initiates panic button it sends current location to the selected contacts.

2. Location is sent in the form of latitudinal and longitudinal values.

*3.5.3.3 Functional Requirements*

*3.5.3.3.1 Introduction:*

In order to send location to the selected contacts,when the victim is in danger.

*3.5.3.3.2 Input:*

Victim must initiate panic button continuously for 4 times inorder to send the current location.

*3.5.3.3.3 Processing:*

* Victim should be aware that he/she is in danger.
* Then victim should initiate the panic button.
* Then it sends current location to the selected contacts.
* Location is sent in the form of latitudinal and longitudinal values.

*3.5.3.3.4 Output:*

Current location of the victim is send to the selected contacts.

## *3.5.4 Panic Button*

*3.5.4.1 Description and Priority*

This feature is used to send current location and default message to the registered contacts.

*3.5.4.2 Stimulus/Response Sequences*

1. As soon as user initiates panic button it sends current location and default message to the selected contacts.

2. Location is sent in the form of latitudinal and longitudinal values.

3. A default message is sent to the selected contacts.

*3.5.4.3 Functional Requirements*

*3.5.4.3.1 Introduction:*

Panic button is used to send location and message to the selected contacts.

*3.5.4.3.2 Input:*

Victim must initiate panic button continuously for 4 times.

*3.5.4.3.3 Processing:*

* Victim should be aware that he/she is in danger.
* Then victim should initiate the panic button.
* Then it sends current location to the selected contacts.
* It also sends an alert message.
* Location is sent in the form of latitudinal and longitudinal values.

*3.5.4.3.4 Output:*

Current location of the victim is send to the selected contacts.

**3.6 NON FUNCTIONAL REQUIREMENTS**

### *Availability*

If the internet service gets disrupted while sending location and SMS to the user, the information can be sent again only to the particular user.

### *Usability*

As the system is easy to handle and navigates in the most expected way with no delays. In that case the system program reacts accordingly and transverses quickly between its states.

### *Correctness*

This application gives correct and accurate location to the registered contacts. In the same way is also sends the location and alert message to the correct contact. It provides 100% accuracy.

### *3.6.4Scalability*

Even though the number of users has increased the performance of the system does not decreases. The present system has the ability to handle increase in load without affecting the performance of the system.

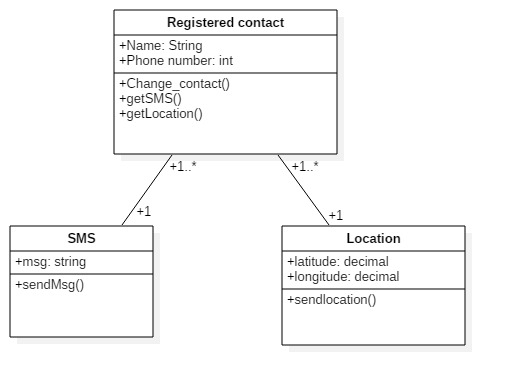
***3.6.5 Efficiency***

The implementation requires fewer resources in terms of disk space, memory and bandwidth. As the databases in the app are updated regularly at frequent intervals, the resultsof the query gives more updated information.

**3.7 SYSTEM DESIGN**

*3.7.1 CLASS DIAGRAM*

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

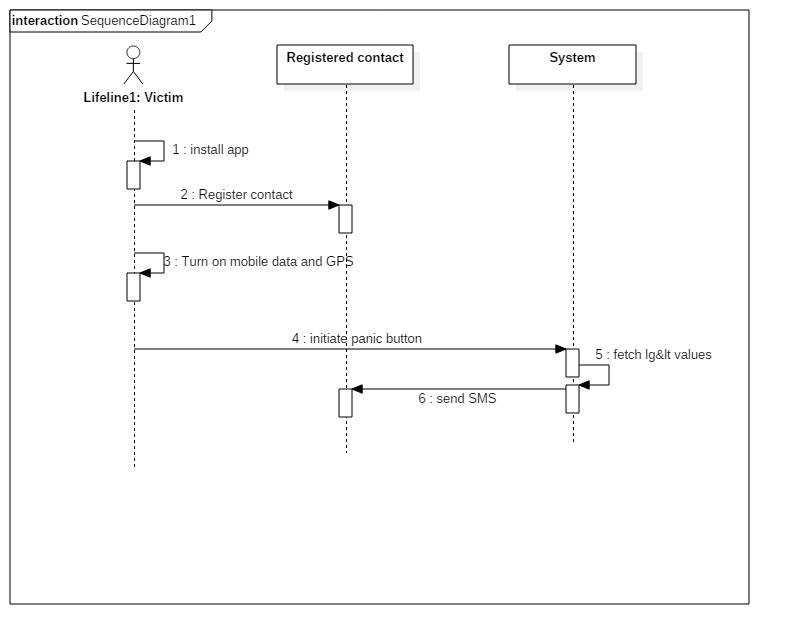


**Fig 3.4 Class diagram**

This application consists of three classes. One class for storing registered contacts, one class for storing SMS and another class for location.The Registered contact class consists of attributes name and phone number of the registered contact; name is stored in the form of string and phone number in the form of number. It consists of operations like Change contact, get sms from the victim and get location from the victim. The SMS class consists of attributes like msg which takes string value as input, stores a default message. This contains single operation for sending SMS to the contacts. The Location consists of attributes like latitude and longitude which takes decimal values as input data type. It consists of only one operation to send the location to the contacts.

*3.7.2 SEQUENCE DIAGRAM*

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

******

**Fig 3.5 Sequence diagram**

After installing the application the victim is asked for registration of some contacts. After registration the victim should be aware that whether the mobile data and GPS are turned on or not. Whenever the victim is in danger the she should initiate the panic button. As soon as the panic button is initiated the location is fetched in the form of latitude and longitude values from the Google server. The lg&lt values are again redirected to the Google server to fetch the location in the form of address, and then address is sent to the registered contacts in the form of SMS.

* 1. **DATABASE DESIGN**

**TABLE DESCRIPTION**

The following table is retrieved from the database once the user enters the details.

SQLite database is used in Feminine Security Using Android Application App.

1. User registration table stores the phone number and name of the user.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Data type** | **Constraints** |
|  |  |  |
| Phone Number | Number | NOT NULL |
|  |  |  |
| Name | Varchar2(40) | NOT NULL |
|  |  |  |

**Table 3.1 User Registration**

**CHAPTER-4**

**IMPLEMENTATION**

**4.1 TOOLS USED**

* Android studio 2.3
* SDK version – 26
* Minimum SDK version – 19
* JDK version - jdk1.8.0\_101

Android is an open source and Linux-based **Operating System** for mobile devices such as smartphones and tablet computers. Android was developed by the *Open Handset Alliance*, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

**Android IDEs**

There are so many sophisticated Technologies are available to develop android applications, the familiar technologies, which are predominantly using tools as follows

* [Android Studio](https://www.tutorialspoint.com/android/android_studio.htm)
* [Eclipse IDE(Deprecated)](https://www.tutorialspoint.com/android/android_eclipse.htm)

Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.



**Fig 4.1 Architecture diagram for Android OS**

**Linux kernel**

At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

**Libraries**

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

**Android Runtime**

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android.

The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**Application Framework**

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

The Android framework includes the following key services −

* **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
* **Content Providers** − Allows applications to publish and share data with other applications.
* **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
* **Notifications Manager** − Allows applications to display alerts and notifications to the user.
* **View System** − An extensible set of views used to create application user interfaces.

**4.2 PSEUDO CODE**

## CREATING THE BROADCAST RECEIVER

A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and overriding the onReceive() method where each message is received as a **Intent** object parameter.

public class MyReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

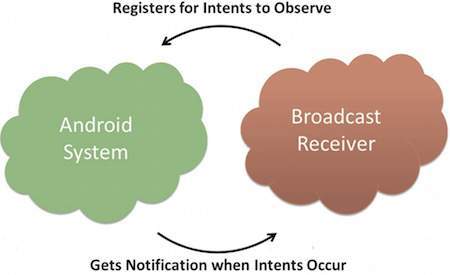
Toast.makeText(context, "Intent Detected.", Toast.LENGTH\_LONG).show();

}

}

## REGISTERING BROADCAST RECEIVER

An application listens for specific broadcast intents by registering a broadcast receiver in *AndroidManifest.xml* file. Consider we are going to register *MyReceiver* for system generated event ACTION\_BOOT\_COMPLETED which is fired by the system once the Android system has completed the boot process.



#### Fig 4.2 Broadcast-Receiver

<application

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

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

android:theme="@style/AppTheme" >

<receiver android:name="MyReceiver">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED">

</action>

</intent-filter>

</receiver>

|  |  |
| --- | --- |
| **S.No** | **Event Constant & Description** |
| 1 | **android.intent.action.BATTERY\_CHANGED**  Sticky broadcast containing the charging state, level, and other information about the battery. |
| 2 | **android.intent.action.BATTERY\_LOW**  Indicates low battery condition on the device. |
| 3 | **android.intent.action.BATTERY\_OKAY**  Indicates the battery is now okay after being low. |

</application>

Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver *MyReceiver* and implemented logic inside *onReceive()* will be executed.

There are several system generated events defined as final static fields in the **Intent** class. The following table lists a few important system events.

# GPSTRACKER

|  |  |
| --- | --- |
| Constants | |
| [String](https://developer.android.com/reference/java/lang/String.html) | [GPS\_PROVIDER](https://developer.android.com/reference/android/location/LocationManager.html#GPS_PROVIDER)  Name of the GPS location provider. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [KEY\_LOCATION\_CHANGED](https://developer.android.com/reference/android/location/LocationManager.html#KEY_LOCATION_CHANGED)  Key used for a Bundle extra holding a Location value when a location change is broadcast using a Pending  Intent. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [KEY\_PROVIDER\_ENABLED](https://developer.android.com/reference/android/location/LocationManager.html#KEY_PROVIDER_ENABLED)  Key used for a Bundle extra holding an Boolean status value when a provider enabled/disabled event is broadcast using a Pending Intent. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [KEY\_PROXIMITY\_ENTERING](https://developer.android.com/reference/android/location/LocationManager.html#KEY_PROXIMITY_ENTERING)  Key used for the Bundle extra holding a boolean indicating whether a proximity alert is entering (true) or exiting (false).. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [KEY\_STATUS\_CHANGED](https://developer.android.com/reference/android/location/LocationManager.html#KEY_STATUS_CHANGED)  Key used for a Bundle extra holding an Integer status value when a status change is broadcast using a Pending Intent. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [MODE\_CHANGED\_ACTION](https://developer.android.com/reference/android/location/LocationManager.html#MODE_CHANGED_ACTION)  Broadcast intent action when [LOCATION\_MODE](https://developer.android.com/reference/android/provider/Settings.Secure.html#LOCATION_MODE) changes. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [NETWORK\_PROVIDER](https://developer.android.com/reference/android/location/LocationManager.html#NETWORK_PROVIDER)  Name of the network location provider. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [PASSIVE\_PROVIDER](https://developer.android.com/reference/android/location/LocationManager.html#PASSIVE_PROVIDER)  A special location provider for receiving locations without actually initiating a location fix. |
| [String](https://developer.android.com/reference/java/lang/String.html) | [PROVIDERS\_CHANGED\_ACTION](https://developer.android.com/reference/android/location/LocationManager.html#PROVIDERS_CHANGED_ACTION)  Broadcast intent action when the configured location providers change. |

Location.getlatitude() is used to get the latitude value from the Google server.

Location.getlongitude() is used to get the longitude value from the Google server.

Location manager provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location, or to fire an application-specified [Intent](https://developer.android.com/reference/android/content/Intent.html) when the device enters the proximity of a given geographical location.

**GPS\_PROVIDER**

[String](https://developer.android.com/reference/java/lang/String.html) GPS\_PROVIDER

Name of the GPS location provider.

This provider determines location using satellites. Depending on conditions, this provider may take a while to return a location fix. Requires the permission

[ACCESS\_FINE\_LOCATION](https://developer.android.com/reference/android/Manifest.permission.html#ACCESS_FINE_LOCATION).

The extras Bundle for the GPS location provider can contain the following key/value pairs:

* satellites - the number of satellites used to derive the fix

Constant Value: "gps"

if(isGPSEnabled)

{

if(location==null)

{

if(location==null)

{

try {

locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER, MIN\_TIME\_BW\_UPDATES, MIN\_DISTANCE\_CHANGE\_FOR\_UPDATE, 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();

}

}

}

# 

# NETWORK PROVIDER

if(isNetworkEnabled)

{

try {

locationManager.requestLocationUpdates(LocationManager.NETWORK\_PROVIDER, MIN\_TIME\_BW\_UPDATES, MIN\_DISTANCE\_CHANGE\_FOR\_UPDATE, this);

Log.d("N/W Enabled", "Network Enabled");

if (locationManager != null) {

location = locationManager.getLastKnownLocation(LocationManager.NETWORK\_PROVIDER);

if (location != null) {

latitude = location.getLatitude();

longitude = location.getLongitude();

}

}

}

}

If GPS is unavailable for the mobile, then we can trace the location using network service provider. By using network provider we can get the last known location. NETWORK PROVIDER is available in the LocationManager class.

PANIC BUTTON

booleanscreenOn = intent.getBooleanExtra("screen\_state", false);

if (!screenOn)

{

cnt++;

if (cnt == 2)

{

startAction();

cnt = 0;

}

}

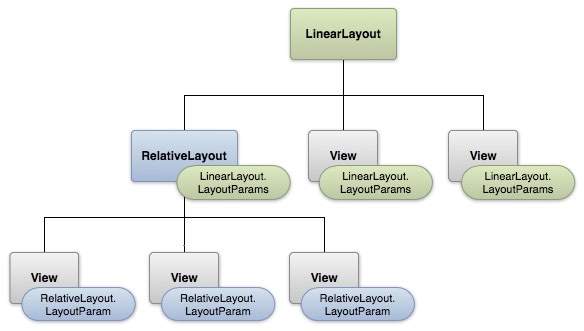
Here panic button is the power button of the mobile. Instead of counting the number of times the button is pressed, here we are counting the number of times the screen is turned on and off. Initially the screen state is in off state which indicates false. If screen is turned on for 2 times then it indicates that panic button is clicked and message is sent.

# Android - UI Layouts

The basic building block for user interface is a **View** object which is created from the View class and occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components like buttons, text fields, etc.

The **ViewGroup** is a subclass of **View** and provides invisible container that hold other Views or other ViewGroups and define their layout properties.

At third level we have different layouts which are subclasses of ViewGroup class and a typical layout defines the visual structure for an Android user interface and can be created either at run time using **View/ViewGroup** objects or you can declare your layout using simple XML file **main\_layout.xml** which is located in the res/layout folder of your project.



#### Fig 4.3 Layout Params

This tutorial is more about creating your GUI based on layouts defined in XML file. A layout may contain any type of widgets such as buttons, labels, textboxes, and so on. Following is a simple example of XML file having Linear Layout −

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

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical">

<TextViewandroid:id="@+id/text"

android:layout\_width="wrap\_content"

|  |  |
| --- | --- |
| **Sr.No** | **Layout & Description** |
| 1 | [**Linear Layout**](https://www.tutorialspoint.com/android/android_linear_layout.htm)  LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally. |
| 2 | [**Relative Layout**](https://www.tutorialspoint.com/android/android_relative_layout.htm)  RelativeLayout is a view group that displays child views in relative positions. |
| 3 | [**Table Layout**](https://www.tutorialspoint.com/android/android_table_layout.htm)  TableLayout is a view that groups views into rows and columns. |
| 4 | [**Absolute Layout**](https://www.tutorialspoint.com/android/android_absolute_layout.htm)  AbsoluteLayout enables you to specify the exact location of its children. |
| 5 | [**Frame Layout**](https://www.tutorialspoint.com/android/android_frame_layout.htm)  The FrameLayout is a placeholder on screen that you can use to display a single view. |
| 6 | [**List View**](https://www.tutorialspoint.com/android/android_list_view.htm)  ListView is a view group that displays a list of scrollable items. |
| 7 | [**Grid View**](https://www.tutorialspoint.com/android/android_grid_view.htm)  GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid. |

android:layout\_height="wrap\_content"

android:text="This is a TextView"/>

<Buttonandroid:id="@+id/button"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="This is a Button"/>

<!-- More GUI components go here -->

</LinearLayout>

Once your layout has created, you can load the layout resource from your application code, in your *Activity.onCreate()* callback implementation as shown below −

publicvoidonCreate(BundlesavedInstanceState){

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

## ANDROID LAYOUTTYPES

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

## LAYOUT ATTRIBUTES

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and their are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

|  |  |
| --- | --- |
| **Sr.No** | **Attribute & Description** |
| 1 | **android:id**  This is the ID which uniquely identifies the view. |
| 2 | **android:layout\_width**  This is the width of the layout. |
| 3 | **android:layout\_height**  This is the height of the layout |
| 4 | **android:layout\_marginTop**  This is the extra space on the top side of the layout. |
| 5 | **android:layout\_marginBottom**  This is the extra space on the bottom side of the layout. |
| 6 | **android:layout\_marginLeft**  This is the extra space on the left side of the layout. |
| 7 | **android:layout\_marginRight**  This is the extra space on the right side of the layout. |
| 8 | **android:layout\_gravity**  This specifies how child Views are positioned. |

# Android - SQLite Database

SQLite is an opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC,ODBCe.t.c

## DATABASE - PACKAGE

The main package is android.database.sqlite that contains the classes to manage your own databases

## DATABASE - CREATION

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object.Its syntax is given below

SQLiteDatabasemydatabase = openOrCreateDatabase("your database name",MODE\_PRIVATE,null);

## DATABASE - INSERTION

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

mydatabase.execSQL("CREATE TABLE IF NOT EXISTS TutorialsPoint(Username VARCHAR,Password VARCHAR);");

mydatabase.execSQL("INSERT INTO TutorialsPoint VALUES('admin','admin');");

Apart from this, there are other functions available in the database package, which does this job. They are listed below

|  |  |
| --- | --- |
| **Sr.No** | **Method & Description** |
| 1 | **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandlererrorHandler)**  This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN\_READWRITE OPEN\_READONLY |
| 2 | **openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)**  It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases |
| 3 | **openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)**  It not only opens but create the database if it not exists. This method is equivalent to openDatabase method. |
| 4 | **openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory)**  This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath() |

## DATABASE - FETCHING

|  |  |
| --- | --- |
| **Sr.No** | **Method & Description** |
| 1 | **getColumnCount()**  This method return the total number of columns of the table. |
| 2 | **getColumnIndex(String columnName)**  This method returns the index number of a column by specifying the name of the column |
| 3 | **getColumnName(intcolumnIndex)**  This method returns the name of the column by specifying the index of the column |

We can retrieve anything from database using an object of the Cursor class. We will call a

method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Cursor resultSet = mydatbase.rawQuery("Select \* from TutorialsPoint",null);

resultSet.moveToFirst();

String username = resultSet.getString(0);

String password = resultSet.getString(1);

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

## DATABASE - HELPER CLASS

For managing all the operations related to the database, a helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

publicclassDBHelperextendsSQLiteOpenHelper{

publicDBHelper(){

super(context,DATABASE\_NAME,null,1);

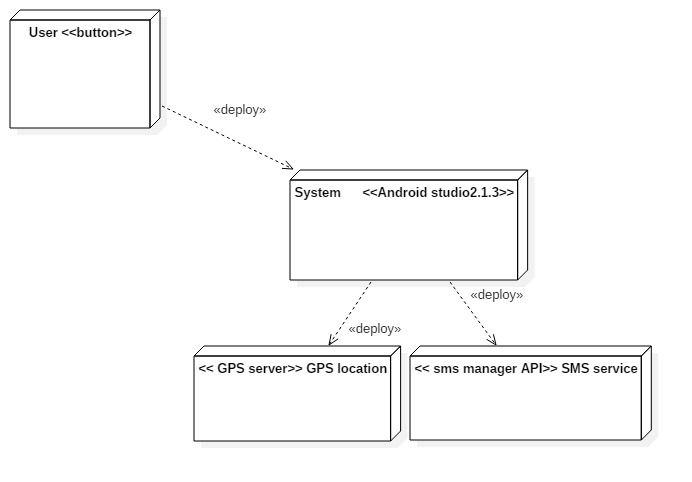
}

publicvoidonCreate(SQLiteDatabasedb){}

publicvoidonUpgrade(SQLiteDatabasedatabase,intoldVersion,intnewVersion){}

}

**4.4 DEPLOYMENT DIAGRAM**



**Fig 4.4 Deployment diagram**

In this application, the system used is Android studio 2.1.3. The latitude and longitude values are fetched from the Google server and again they are redirected to the server to fetch address corresponding to the latitude and longitude values. sms manager API is used send the SMS.

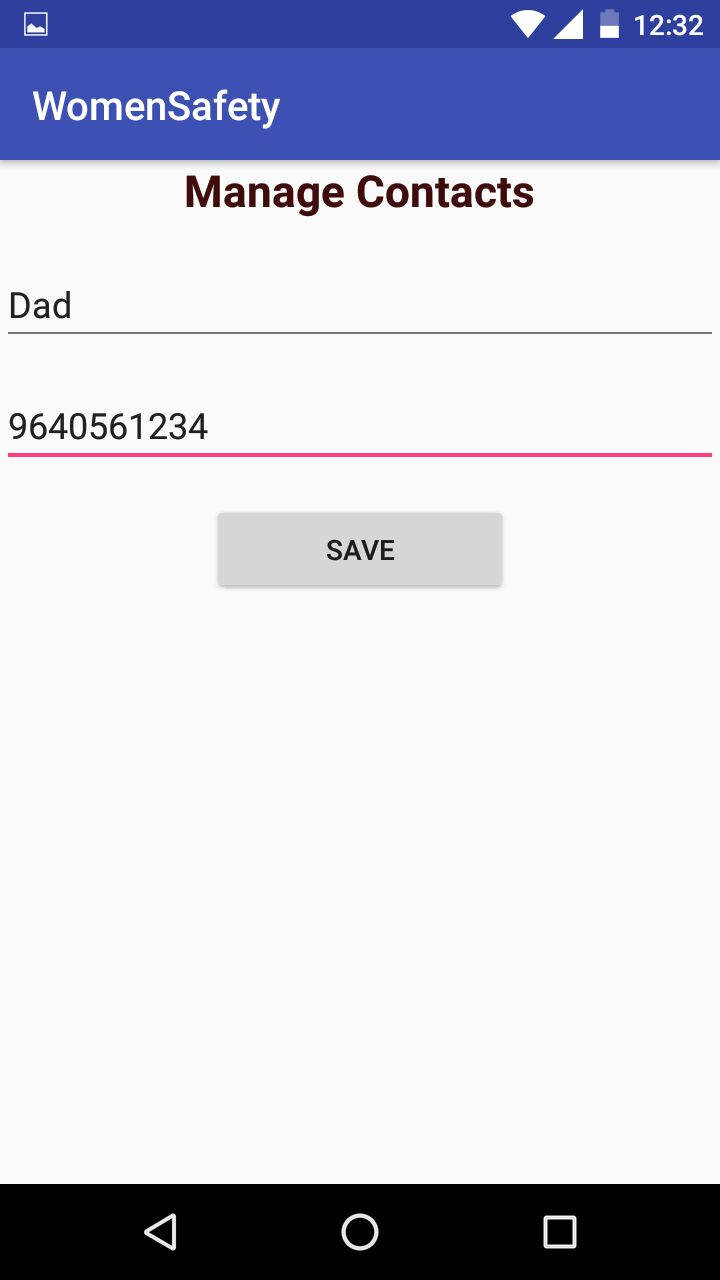
**4.5 SCREENSHOTS**

This is the initial screen appears after installation of application. As soon as the user clicks on contacts icon, it prompts to register contacts screen.



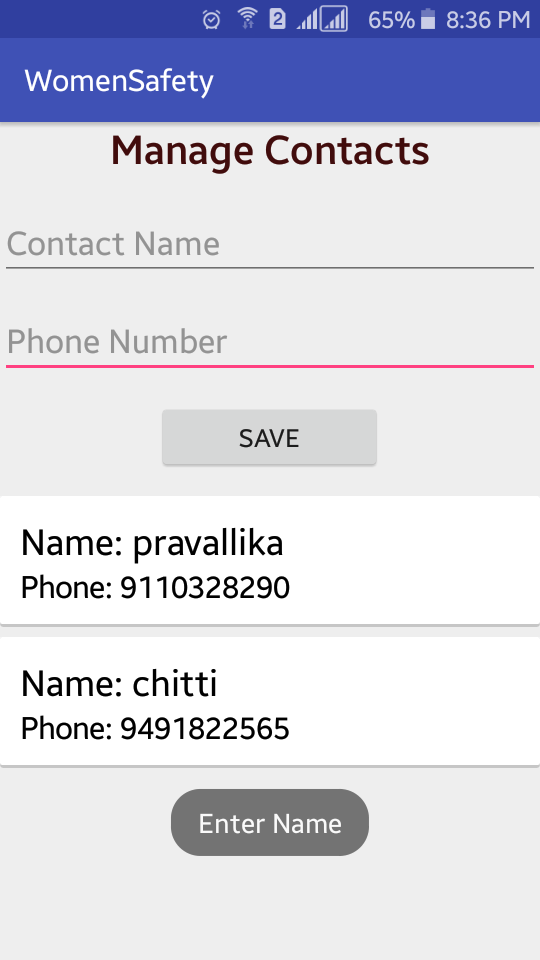
**Fig 4.6 Welcome Screen**

After initiating the contacts button the screen appears in this manner. Here the user is asked to register some contacts of near and dear.



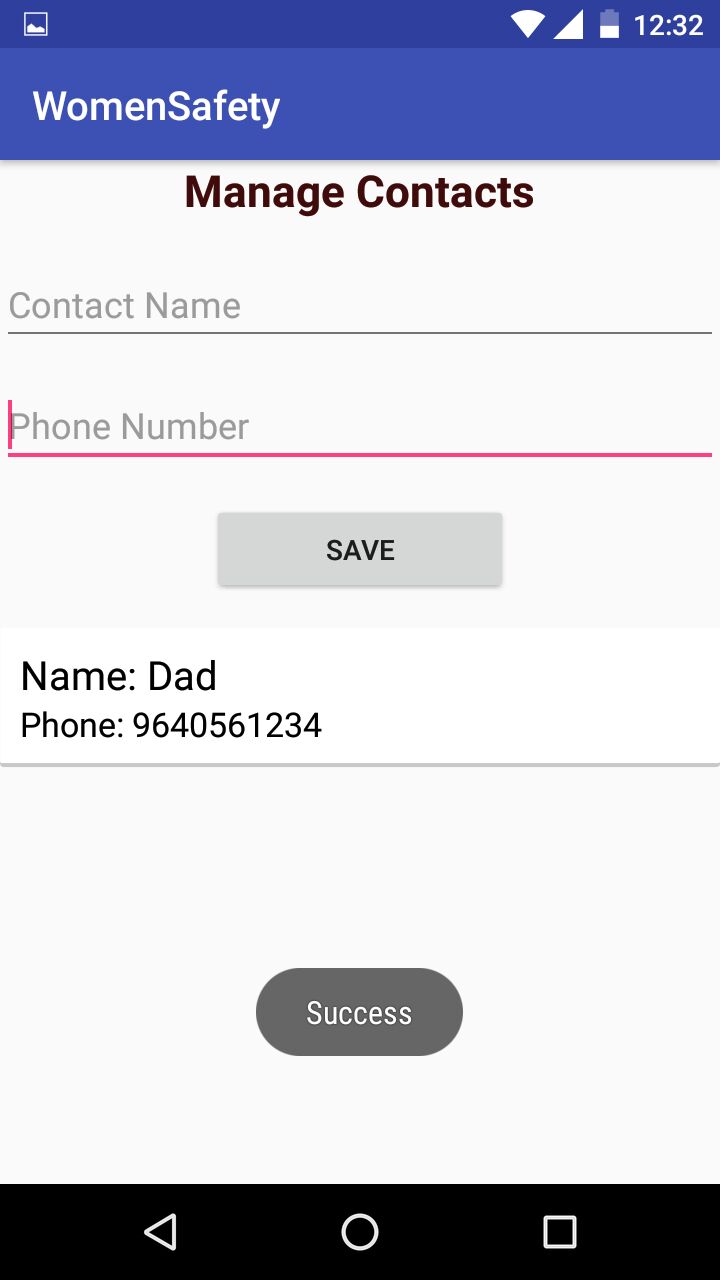
**Fig 4.7 Registration screen**

Any number of contacts can be registered at a time based on user requirement. Here the input for name is given in the form of text, whereas the input for phone number is given in the form of number. After ensuring all the details are correct the user should initiate save button.



**Fig 4.8 Registration mismatch screen**

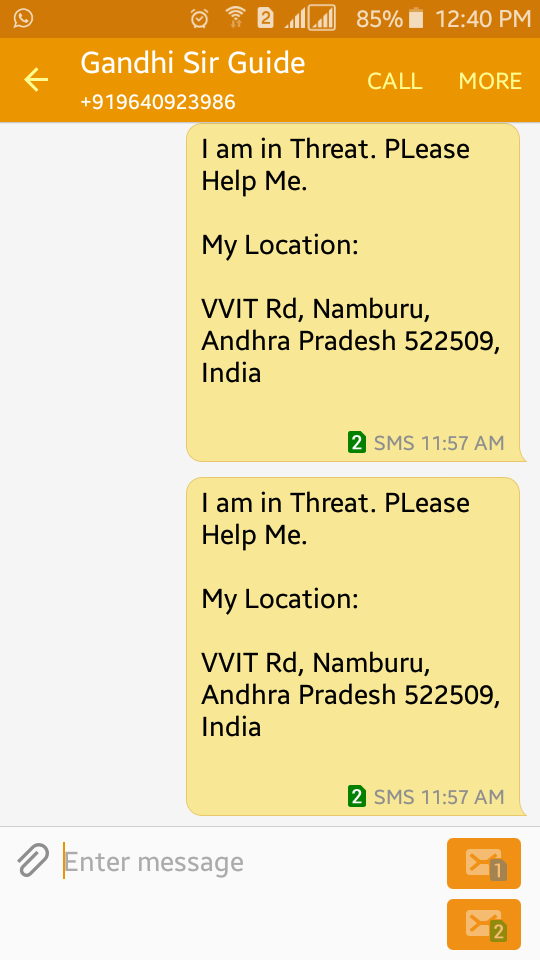
If the user initiates the save without giving necessary input then here displays a toast message which asks the user to enter necessary details. For example if user clicks save button without entering name the a toast message “Enter Name” is displayed. Similarly, if the user clicks save button without entering phone number then a toast message “Enter mobile number is displayed”.



**Fig 4.9 Registration submitting screen**

After assuring that entered details are correct, user should initiate “SAVE” button. Then a toast message is displayed as “SUCCESS”.

If user found any details are incorrect then he can also edit the contact.



**Fig 4.10 Alert message screen**

On initiating the panic button a message which contains current locationis sent to the registered contacts.The message appears to the registered contacts in this way.

**CHAPTER-5**

**TESTING**

**5.1. TESTING TECHNOLOGIES**

**System Testing**

Testing is a process of finding faults with the package. To test the software there are

several testing methodologies.

**Testing Objectives**

The main objective of testing is to uncover a host of errors, systematically and with

minimum effort and time. Stating formally, we can say,

* Testing is a process of executing a program with the intent of finding an error.
* A successful test is one that uncovers an as yet undiscovered error.
* A good test case is one that has a high probability of finding error, if it exists.
* The tests are inadequate to detect possibly present errors.
* The software more or less confirms to the quality and reliable standards.

**Levels of Testing**

In order to uncover the errors present in different phases we have the concept of levels

of testing. The basic levels of testing are as shown below…

User input validation

Testing

Client need

Requirements System Testing

Integration Testing

Design

Unit Testing

Code

**Fig 5.1 Levels of Testing**

**Types of Testing**

* Unit Testing
* System Testing
* User Input Validation Testing

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software i.e. themodule. Using the detailed design and the process specifications testing is done touncover errors within the boundary of the module. All modules must be successful inthe unit test before the start of the integration testing begins.

In this project each service can be thought of a module. There are modules like User Registration and Panic button. Givingdifferent sets of inputs has tested each module. When developing the module as wellas finishing the development so that each module works without any error.

**System Testing**

Here the entire software system is tested. The reference document for thisprocess is the requirements document, and the goal to see if software meets itsrequirements.

**User input validation**

The User input must be validated to confirm to expected values. The fieldsshould also not be empty.

**5.2 TEST CASES**

**Test Case 1**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Test case 1** |  | **Registration** |
|  |  |  |  |
| **2.** | **Precondition** |  | Enter Name. |
|  |  |  | Enter Phone Number. |
|  |  |  |  |
|  |  |  |  |
| **3.** | **Description** |  | User has to enter a valid Name, |
|  |  |  | Phone number. |
|  |  |  |  |
| **4.** | **Test Steps** |  | Fill all the details which are given in the |
|  |  |  | Registration screen. |
|  |  |  |  |
| **5.** | **Expected Output** |  | If all the details are given by the user are |
|  |  |  | Valid then register activity is performed. |
|  |  |  |  |
| **6.** | **Actual Output** |  | If the user enters all the details in the |
|  |  |  | register screen then the next activity is |
|  |  |  | Performed. |
|  |  |  |  |
| **7.** | **Status** |  | **Pass** |
|  |  |  |  |

**Table.5.1 Registration Success**

**Test case 2**

|  |  |  |  |
| --- | --- | --- | --- |
| **1.** | **Test case 2** |  | **Panic Button** |
|  |  |  |  |
| **2.** | **Pre-condition** |  | Click the Panic button. |
|  |  |  |  |
| **3.** | **Description** |  | User has to click the panic button  for 3 times. |
|  |  |  |  |
|  |  |  |  |
| **4.** | **Test Steps** |  | User has to click the Panic button for 3 times whenever she is in danger. |
|  |  |  |  |
|  |  |  |  |
| **5.** | **Expected Output** |  | If the panic button is clicked properly then an alert is sent. |
|  |  |  |  |
|  |  |  |  |
| **6.** | **Actual Output** |  | If the Panic button is clicked properly then an alert message is sent by which the |
|  |  |  | Registered user can be able to trace the victim. |
|  |  |  |  |
|  |  |  |  |
| **7.** | **Status** |  | **Pass** |
|  |  |  |  |

**Table.5.2 Panic button**

**CHAPTER-6**

**CONCLUSION & FUTURE SCOPE**

**Feminine Security Using Android Application App** isdeveloped for an Android Phone to ensure the security of the women in the society which sends the alert message and current address to the registered contacts which allows tracing the victim’s present location.

In future we can extend the project by sending a notification to the registered contacts along with the text message. After registered contact initiating the notification it automatically redirects to the Google Maps for live tracing of the victim’s location. Live tracing of the victim is available to the registered contacts until the victim clicks STOP button in the application. FEMININE SECURITY USING ANDROID APPLICATION is developed for Android mobiles, as Android is exploiting most of the mobile market. Many users own Android mobiles these days, than other mobiles such as Windows and IOS. In this regard, FEMININE SECURITY USING ANDROID APPLICATION is socially feasible.

**BIBLIOGRAPHY**

1. Prof.R.A.Jain “Women’s safety using IOT”, International Research Journal of Engineering and Technology. Vol.04, Issue.05, May-2017

2. B.Chougula, “Smart girls security system,” International Journal of Application orInnovation in Engineering & Management, Volume 3, Issue 4,April 2014.

3. IOSR Journal of Computer Engineering (IOSR-JCE) “A Mobile Based Women Safety Application” Volume 17, Issue 1.

4. https://www.aiktcdspace.org:8080/jspui/handle/123456789/2053

5. <https://developer.android.com/studio/intro/index.html>

6. <https://developer.android.com/reference/packages.html>

7. <https://www.gadgetsnow.com/featured/10-best-women-safety-apps/articleshow/57535322.cms>

8. <https://developer.android.com/reference/android/location/LocationManager.html>

9.

**USER MANUAL**

**FEMININE SECURITY USING ANDROID APPLICATION can be used in the following manner**

1. After installing the App the user has to see the home screen first.

2. After installation the app is provide with home page. The home screen contains module to Register Contact.



3. After splash screen Register Screen will be appear. User has to enter the details which are given in the register form like name, phone number which is mandatory and click on the Save button.

**SAVE**

4. After clicking on save button the list of saved contacts will be displayed on the screen.

5. When the victim is in danger, then she should initiate panic button.

****

**PUBLICATION DETAILS**

