Development phase III requirements report

for

RING ME – A Mobile management application

Version 1.0

Prepared by: Team Mode Changer (Venkata Vikas Chirumamilla, Chenchu Sai Krishna Kolli, Siri Gogineni, Revanth Reddy Malreddy, Sai Teja Malle)

University of North Texas

11/26/2018

Table of Contents

1. Development Phase 3 Overview

In this phase we target to finish requirements to monitor the remote phone. In this way we can get the information about our mobile even though if it’s not with us. So, we planned to complete SR3: Requirements to monitor remote phone like fetching the location coordinates of our mobile, viewing the activity logs, and fetching the IMEI number of the mobile. The functional requirements developed in this phase are:

- FR3.6: Feature to read the SMS command and fetch the location coordinates of the phone (Section 3.4.1)
- FR3.7: User activity logs feature (Section 3.4.2)
- FR3.8: Feature to read the SMS command and fetch the IMEI number of the phone (Section 3.4.3)

After this phase, user must be able to get the details of their mobile.

1.1 SR3: Requirements to monitor the remote phone

Monitor the remote phone features allows users to view/monitor his phone location coordinates, battery status, IMEI number from the other phone. This also includes a feature to view activity logs of the user. Feature hierarchy of features to monitor the remote phone is shown in figure 1.1.

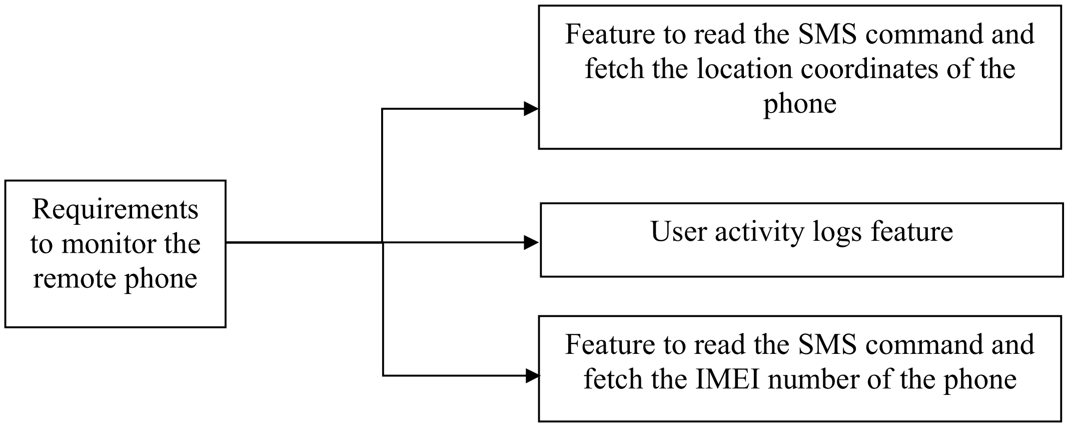

Requirements to monitor the remote phone
Feature to read the SMS command and fetch the location coordinates of the phone
User activity logs feature
Feature to read the SMS command and fetch the IMEI number of the phone

Figure 1.1. Feature hierarchy of features to monitor the remote phone

1.2 FR3.6: Feature to read the SMS command and fetch the location coordinates of the phone

If a phone is misplaced and lost, then user can try to know the location of the phone by getting its location coordinates. All user needs to do is configure a SMS command against ‘LOCATION’ in the Ring me application. This command will be stored in our database. This command will be used by the application to identify and respond accordingly by sending location coordinates to the sender phone.

1.3 FR3.7: User activity logs feature

This feature is provided inside the application which allows user to view his recent activities performed in the app like change of password, change of phone number, change of personal information, change of keywords etc.

1.4 FR3.8: Feature to read SMS command and fetch the IMEI number of the phone

If a phone is misplaced and lost, then user there is no way to trace the location of the phone, then user can try the IMEI feature. He can get the IMEI of his remote phone and this can be used for blacklisting the remote phone. User needs to do is configure a SMS command against 'IMEI' in the Ring me application. This command will be stored in our database. This command will be used by the application to identify and respond accordingly by sending IMEI number of the remote phone to the sender phone.

1.5 Updated requirements

We have completed the *'FR3.6: Feature to read the SMS commands and fetch the location coordinates of the phone (Section 3.4.1)', 'FR3.7: User activity logs feature(Section 3.4.2)' and 'FR3.8: Feature to read the SMS command and fetch the IMEI number of the phone (Section 3.4.3)'.* We also changed a lot of GUI. Due to time constraint, and updated OS version, we couldn't complete some requirements as planned before. The Table 1.1 Phase III development progress table explain further about the phase progress.

Planned features (4)	Developed features (5)	Future Scope (3)
FR 3.3 FR 3.4 FR 3.5 FR 3.6 FR 3.7 FR 3.8	FR 3.6 FR 3.7 FR 3.8	FR 3.3 FR 3.4 FR 3.5

Table 1.1. Phase III development report

1.6 Future Scope:

We have completed all the functional requirements which we had planned to work on in the deliverable – 2 expect for some and they are *'FR3.3: Feature to read the SMS command and turn on/ off GPS(Section 3.3.3'), 'FR3.4: Feature to read the SMS command and lock the phone(Section 3.3.4)'* and *'FR3.5: Feature to read the SMS command and wipe the data on the phone(Section 3.3.5).* We were unable to complete these requirements as there was problem with the update of android OS i.e., Oreo. We faced a problem in getting the exact GPS location of the phone, but were able to get the location coordinates. We would like to complete these other issues also in further days.

2. UML Design

2.1 Class Diagram

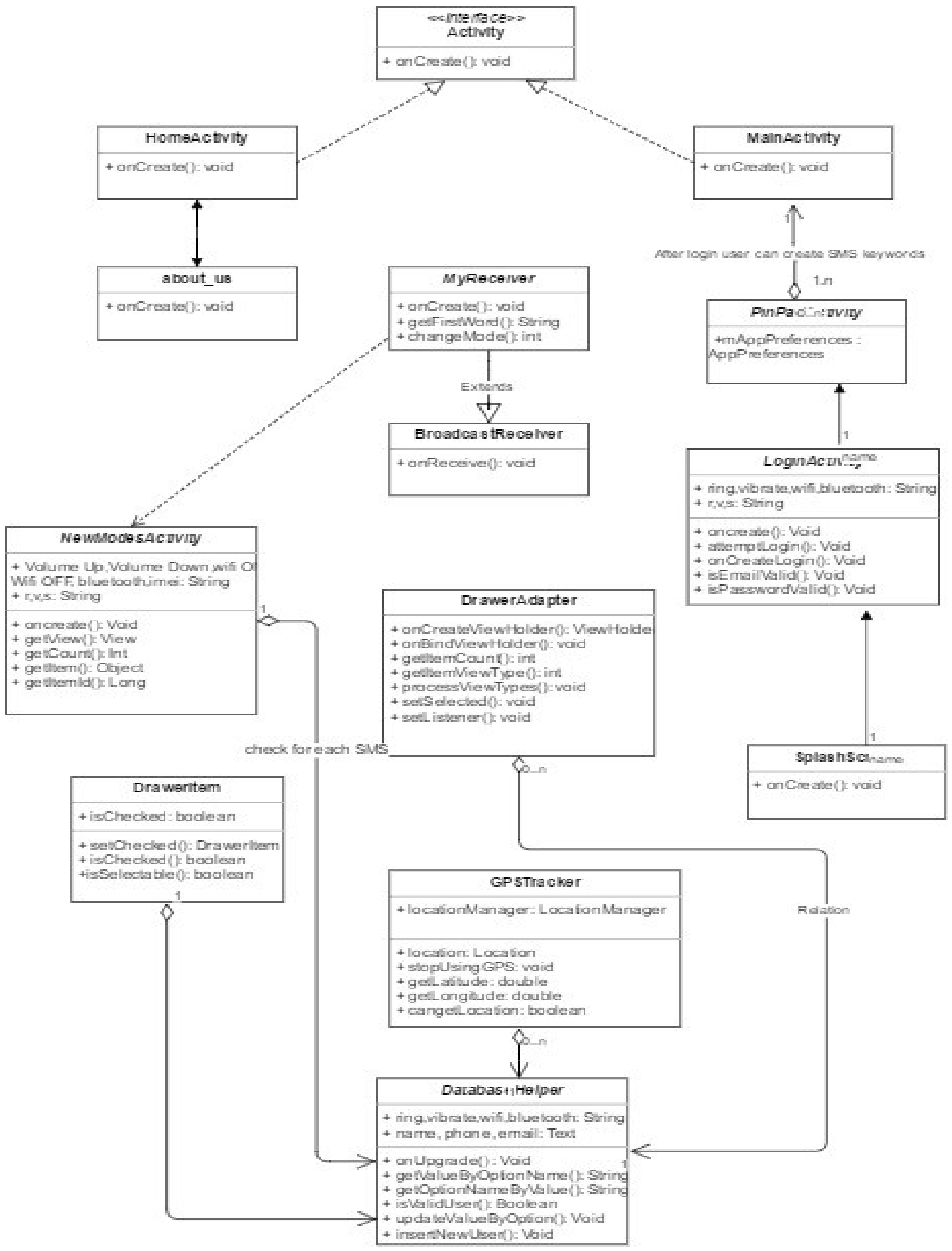


Figure2.1. Class diagram for phase-III requirements

2.2 Sequence Diagram

A sequence diagram is a detailed interaction diagram between the objects in the system and their related operations. They give us an idea about the timely order when the operations occur. The following are the normal case and the error case sequence diagrams (updated) for the Development Phase II requirements of RING ME – A mobile management android application.

As stated, user needs to sign up and login before using the application. Hence in the below sequence Figure 2.2, System always checks for user signup and if he is already signed up, then it asks for Login.

Normal case:

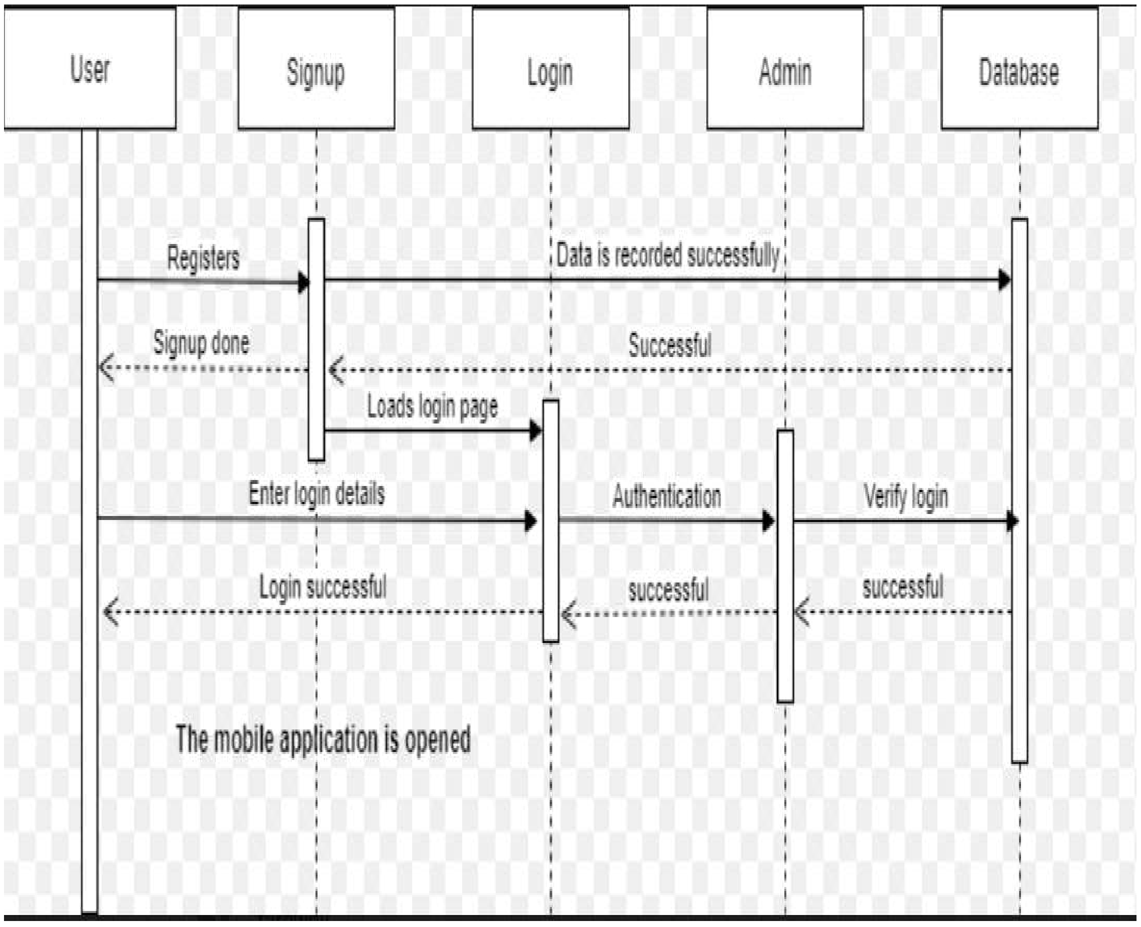


Figure 2.2 Normal case sequence diagram for Phase-III requirements

Error case:

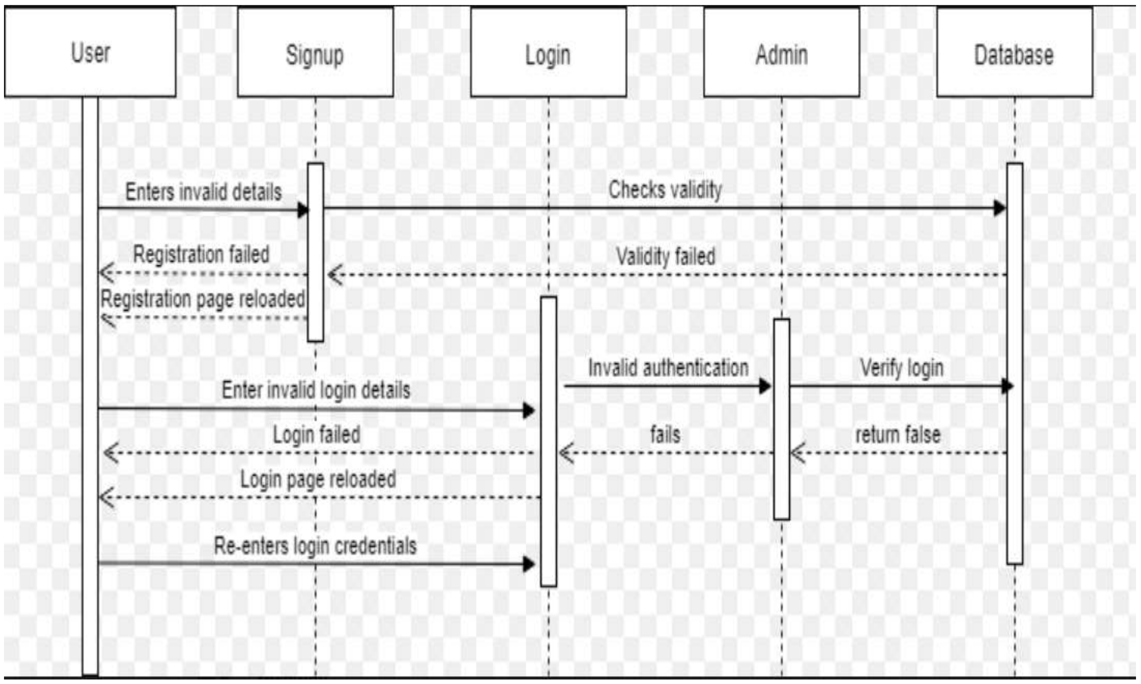


Figure 2.3 Error case sequence diagram for Phase-III requirements

2.3 Use Case Diagram

A use case is a procedure used in a system to distinguish and arrange system requirements. It shows the interactions between the actors and the use cases.

For working case when the user sends the correct keyword, the receiver after receiving the SMS will be able to perform the required function like altering the volume,turning on/off WIFI depending on the keyword sent.

In the error case when user sends the wrong keyword, the receiver after receiving the SMS will not be able to perform any action further.

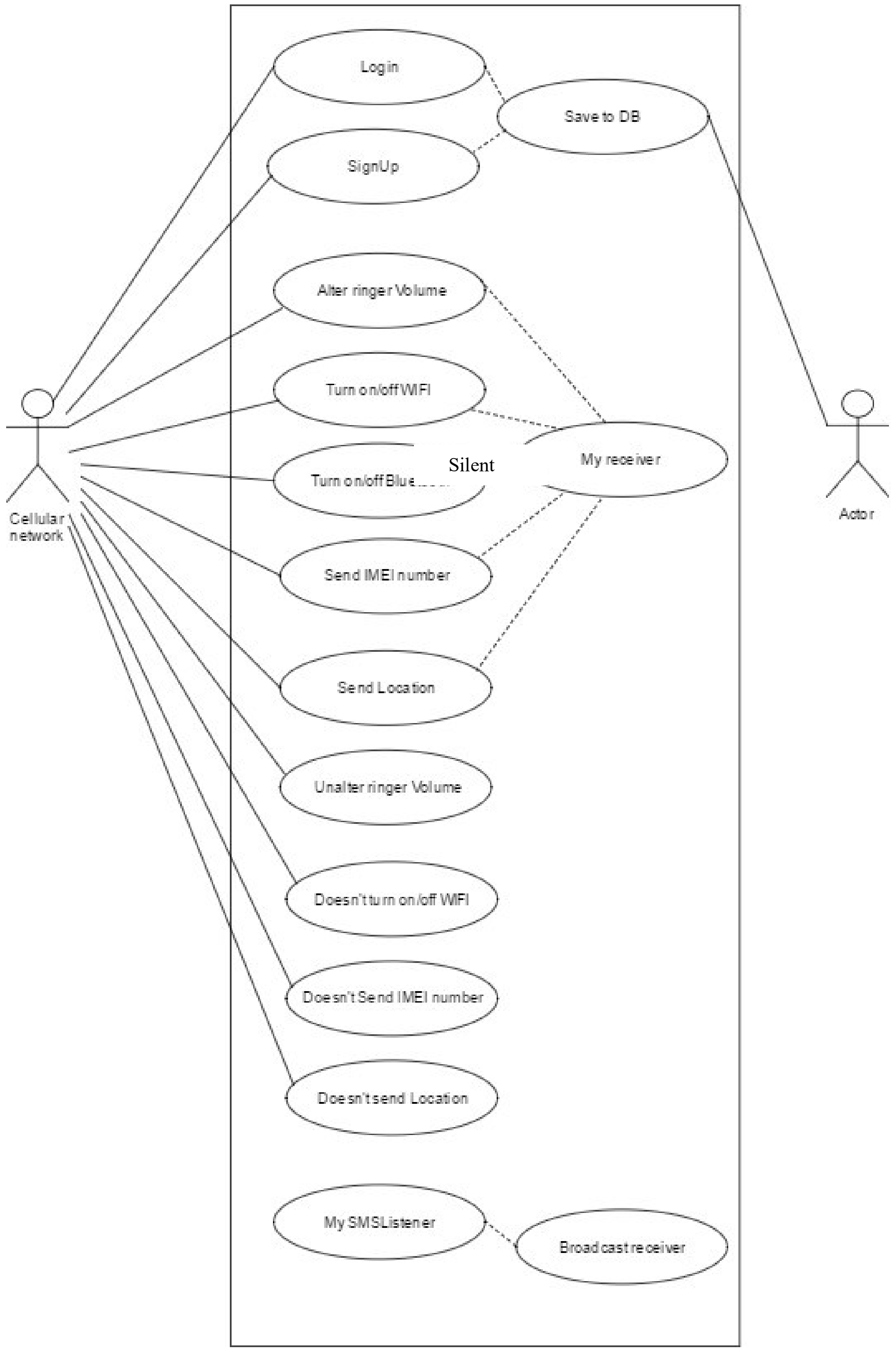


Figure 2.4 Use case diagram for Phase-III requirements

3. Test Cases

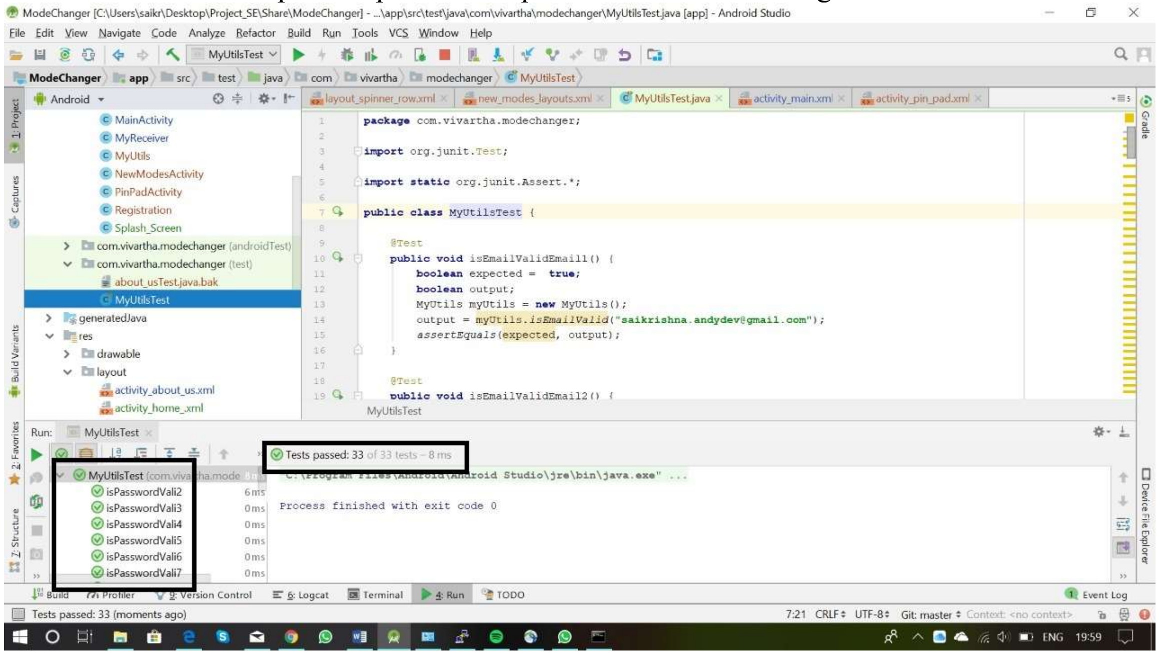
3.1 Unit test cases report

Phase III requirements are focused on monitoring the remote mobile- Fetching the location coordinates, activity logs and fetching IMEI number. The test cases which we have performed are almost the same as we implemented in the Phase II requirements. They were mainly on the User authentication- Registration and Login. Efforts were made to identify more unit cases around these requirements. A total of 33 Unit test cases were found and tested. In this phase, we mainly concentrated on the GUI and so there was no need for any extra test cases.

https://github.com/Sai11262246/ModeChanger/tree/master/Test%20cases

Number of Unit test case identified	33
Number of Unit test case passed	33

Unit test case execution report for phase III requirements is shown in figure 3.1



```
package com.vivartha.modechanger;

import org.junit.Test;

import static org.junit.Assert.*;

public class MyUtilsTest {

    @Test
    public void isEmailValidEmail1() {
        boolean expected =  true;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output = myUtils.isEmailValid("saikrishna.andydev@gmail.com");
        assertEquals(expected, output);
    }

    @Test
    public void isEmailValidEmail2() {
```

```
Tests passed: 33 of 33 tests – 8 ms
MyUtilsTest (com.vivartha.mode...
  isPasswordVali2    6 ms
  isPasswordVali3    0 ms
  isPasswordVali4    0 ms
  isPasswordVali5    0 ms
  isPasswordVali6    0 ms
  isPasswordVali7    0 ms

"C:\Program Files\Android\Android Studio\jre\bin\java.exe" ...

Process finished with exit code 0

Tests passed: 33 (moments ago)
```

Figure 3.1 Unit test cases execution report for phase III requirements.

3.2 Unit test cases for classes

- **MyUtilsTest: Junit code for MyUtils.java class**

```
package com.vivartha.modechanger;
import org.junit.Test;
import static org.junit.Assert.*;
public class MyUtilsTest {
    @Test
    public void isEmailValidEmail1() {
        boolean expected =  true;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("saikrishna.andydev@gmail.com");
        assertEquals(expected, output);
    }
    @Test
    public void isEmailValidEmail2() {
        boolean expected =  true;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("saikrishnaandydev@gmail.com");
        assertEquals(expected, output);
    }
    @Test
    public void isEmailValidEmail3() {
        boolean expected =  false;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("saikrishnaandydevgmail.com");//no @
        assertEquals(expected, output);
    }
    @Test
    public void isEmailValidEmail4() {
        boolean expected =  false;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("saikrishnaandydev@gmailcom");//no
.com
        assertEquals(expected, output);
    }
    @Test
    public void isEmailValidEmail5() {
        boolean expected =  true;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("saikrishnaandydev@gmail.in");// .in
        assertEquals(expected, output);
    }
    @Test
    public void isEmailValidEmail6() {
        boolean expected =  true;
        boolean output;
        MyUtils myUtils = new MyUtils();
        output =
myUtils.isEmailValid("krishna@gmail.com");// small length
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("Saikrishna@123");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordVali2() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("sAikrishna@123");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordVali3() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saikrishnA#123");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordVali4() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saiKrishna#0");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordVali5() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output = myUtils.isPasswordValid("");
        assertEquals(expected, output);

    }
    @Test
    public void isPasswordVali6() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saikrishna");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordVali7() {
        boolean expected = false;
```

```
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isPasswordValid("123456789");
            assertEquals(expected, output);
        }
        @Test
        public void isPasswordVali8() {
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isPasswordValid("MSDHONI#123");
            assertEquals(expected, output);
        }
        @Test
        public void isPasswordVali9() {
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isPasswordValid("MsDhoni#123");
            assertEquals(expected, output);
        }


// Registration
      @Test
        public void isNameValid1(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("SaiKrishna");
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid2(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("");
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid3(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("SK");
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid4(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isNameValid("SaiKrishna12");
```

```
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid5(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("Sai");
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid6(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("SaiK");
            assertEquals(expected, output);
        }
        @Test
        public void isNameValid7(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isNameValid("S");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid(" ");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber1(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("123");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber2(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isPhoneValid("9848022338");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber3(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("784569321");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber4(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("00000000");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber5(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("1234567");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber6(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("12345");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber7(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output =
    myUtils.isPhoneValid("9985785724");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber8(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("998-578-
    5724");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber9(){
            boolean expected = true;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("998 578
    5724");
            assertEquals(expected, output);
        }
        @Test
        public void isValidPhoneNumber10(){
            boolean expected = false;
            MyUtils myUtils = new MyUtils();
            boolean output = myUtils.isPhoneValid("143143");
```

```
            assertEquals(expected, output);
        }
    }
```

4. Contribution

4.1 Requirements

Contribution	Developer name
FR3.6: Feature to read the SMS command and fetch the location coordinates of the phone	Vikas Chirumamilla
FR3.7: User activity logs feature	Siri Gogineni
FR3.8: Feature to read the SMS command and fetch the IMEI number of the phone	Revanth Malreddy
Develop the GUI with the new Functional Requirements	Sai Krishna Kolli
Testing and Quality Assurance	Sai Teja Reddy Malle

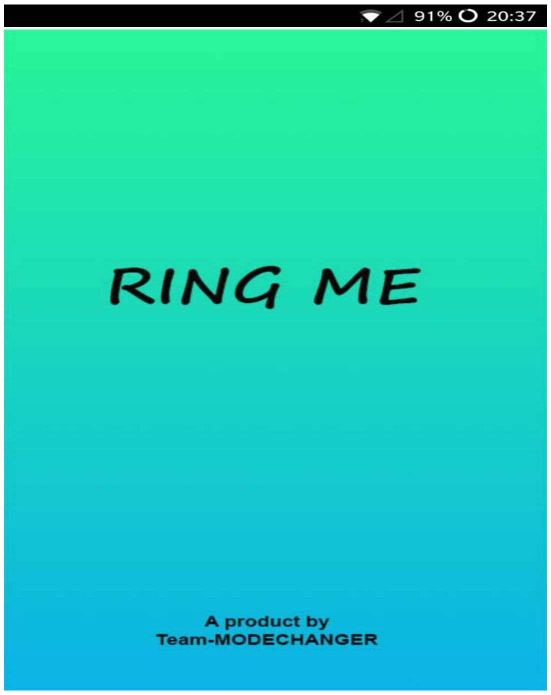
4.2 Components/Classes

Name	Component/Classes
Vikas Chirumamilla	activity_about_us.xml, activity_home_.xml, activity_login.xml, activity_main.xml, cards_layout.xml
Sai Krishna Kolli	GPSTracker.java, MainActivity.java, MyReceiver.java, MyUtils.java, NewModesActivity.java
Revanth Malreddy	about_us.java, ActivityLogModel.java, ChnageFragmentListener.java, DataBaseHelper.java, DrawerAdapter.java
Siri Gogineni	AppController.java, AppPreferences.java, ChnageFragmentListener.java, Home_Activity.java, LoginActivity.java
Sai Teja Malle	PinPadActivity.java, Registration.java, SimpleItem.java, SpaceItem.java,

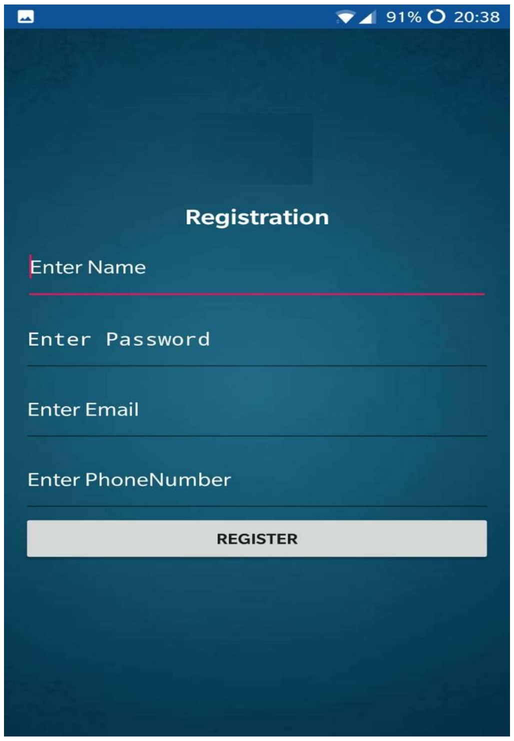
	Splash_Screen.java

5. User Manual

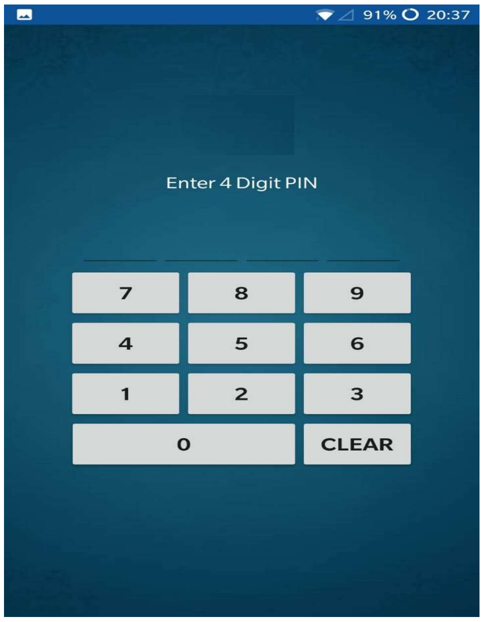
- This is how the splash screen of our mobile management Android application. As soon as we click on this application, the following screen opens.



- For, security reasons, it is important that every user should create an account in this application. The user when opens the app for the first time, he is taken to the "Registration" page where he needs to fill the details. The details include Enter Name, Enter Password, Enter Email, Enter Phone Number.



- After registering, the user needs to choose and enter the PIN with which he can access the application. The user needs to re-enter the PIN for confirmation.
- After choosing the PIN, it redirects to login where user needs to enter PIN for application to be launched.



- After logging to application, it takes user to the home screen where we get Menu Inflator.

- After clicking Menu Inflator, we get different options

Home

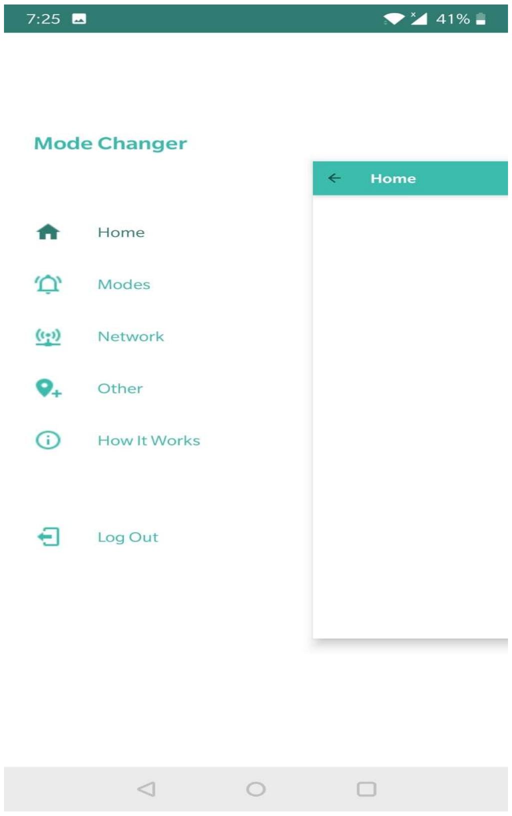
Modes

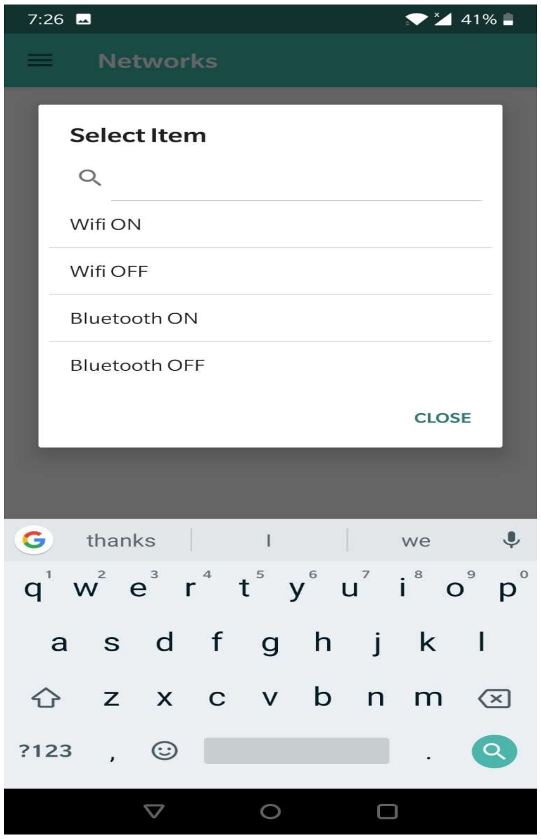
Network

Other

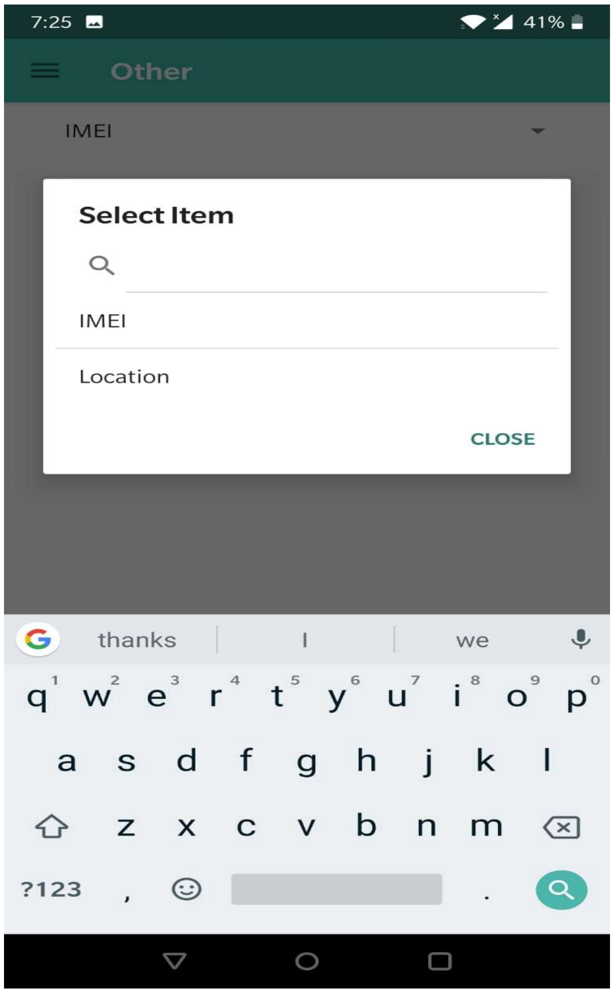
How it works

- Logout option allows us to Logout from the device.

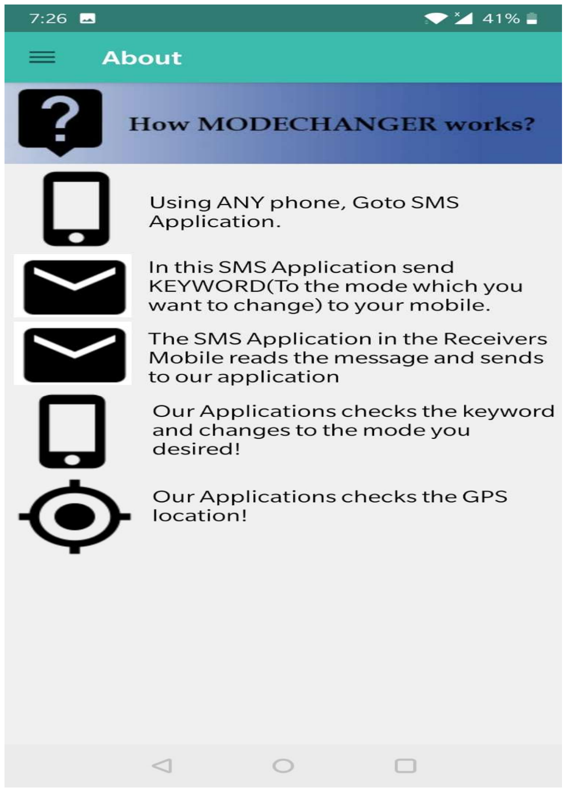

7:25
41%
Mode Changer
Home
Home
Modes
Network
Other
How It Works
Log Out


7:26
41%
Networks
Select Item
Wifi ON
Wifi OFF
Bluetooth ON
Bluetooth OFF
CLOSE
thanks
I
we

- When you select 'Modes' option, it takes you to a screen where you have the option to edit the keyword which you want to use for Ring, Vibrate, Silent, Volume Up and Volume Down.
- Similarly, we have for 'Networks' where we have WIFI off, WIFI On, Bluetooth Off, Bluetooth On.
- When we select 'Other', we have IMEI, Location which can be used to get IMEI number and location co-ordinates.
- You can edit the keyword which you would like to use and click on save.
- There is an option 'Close' to go back. When you select it, it takes you back to the home page.



- In the home page, user will have an option which says how it works. When you select this option, a page appears which will give detailed information about how this android application works. This helps the new users on how to use it.



7:26 41%

About

How MODECHANGER works?

Using ANY phone, Goto SMS Application.

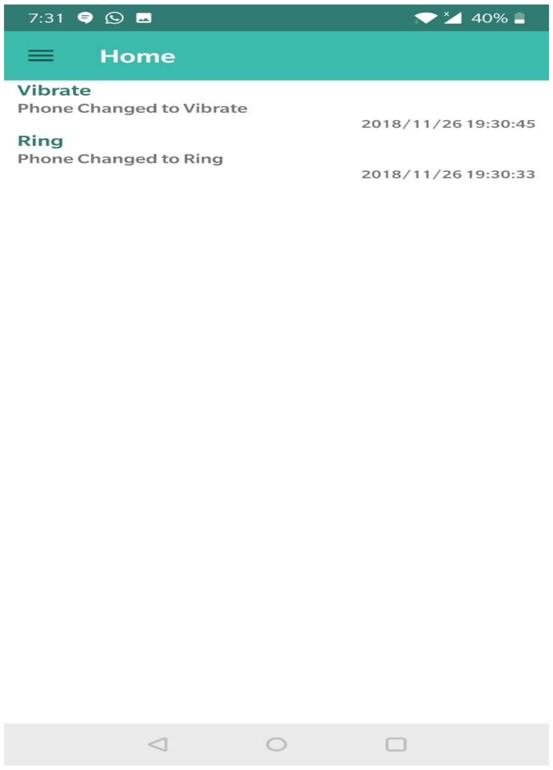
In this SMS Application send KEYWORD(To the mode which you want to change) to your mobile.

The SMS Application in the Receivers Mobile reads the message and sends to our application

Our Applications checks the keyword and changes to the mode you desired!

Our Applications checks the GPS location!

- When the user opens app, he also gets activity log where he can check activities regarding the Mode changes that have been done previously.



7:31 40%

Home

Vibrate
Phone Changed to Vibrate
2018/11/26 19:30:45

Ring
Phone Changed to Ring
2018/11/26 19:30:33

6. Installation instructions

6.1 Enabling APK Installations

- Open your Android's Settings. Use two fingers to swipe down from the top of the screen, then tap the "Settings" Image titled Android7settings.png gear icon in the top-right corner of the resulting drop-down menu.
- Scroll down and tap Apps and notifications. Doing so opens the Apps and Notifications menu.
- If you have an Android running Nougat (Android 7.0), skip to the last step in this part instead.
- Tap Install unknown apps. It should be in the middle of the menu. This option may instead say Install other apps. On some Androids, you may first have to tap Special access.

6.2 Installing RING ME from .apk file

- Download .apk file into your Android mobile which meets the system requirements.
- Open your Android's file manager app, select your default storage location, tap the Downloads folder, and tap the APK file that you downloaded. You can then tap SETTINGS when prompted and enable installations from unknown sources for your file manager app.
- Tap INSTALL. This option is in the bottom-right corner of the screen. Doing so will prompt the APK file to begin installing; once it completes, you'll see an OPEN option appear in the bottom-right corner of the screen. This will open RING ME application.

6.3 Testing the application

- Please find the user manual under section 5 of this document which helps to find out the more details of the application features.
- You can make use of section 1 to get knowledge on the requirements scope.
- Functional test cases have been updated to GitHub.
  https://github.com/Sai11262246/ModeChanger/tree/master/Test%20cases

  Test cases.zip
- Unit testing has been uploaded to
  https://github.com/Sai11262246/ModeChanger/tree/master/SourceCode/app/src/test/java/com/vivartha/modechanger
- Run each of the Junit cases in Android studio after importing the code to Android studio from GitHub.
  https://github.com/Sai11262246/ModeChanger/tree/master/SourceCode

7. References

[1] https://en.wikipedia.org/wiki/Bluetooth