

Development phase II requirements report

for

RING ME – A Mobile management application

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

University of North Texas

11/11/2018

Table of Contents

1. Development Phase 2 Overview	3
1.1 SR2: User authentication requirements	3
1.2 FR2.1: User registration.....	3
1.3 FR2.2: User Login	4
1.4 FR1.4: Feature to read SMS command and alter the ringer volume	4
1.5 FR3.1: Feature to read SMS command and turn on/off Wi-Fi.....	4
1.6 FR3.1: Feature to read SMS command and turn on/off Bluetooth.....	4
1.7 Updated Requirements.....	5
2. UML Design.....	5
2.1 Class Diagram.....	5
2.2 Sequence Diagram (Working and Error case)	7
2.3 Use case Diagram	8
3. Test Cases.....	9
3.1 Unit Test Cases Report	9
3.2 Unit Test Cases for Classes	10
4. Contribution	16
4.1 Requirements	16
4.2 Components/Classes	16
5. User Manual	17
6. Installation instructions	20
6.1 Enabling APK Installations	20
6.2 Installing RING ME from .apk file.....	20
6.3 Testing the application.....	21
7. References	21

1. Development Phase 2 Overview

In this phase we target to finish user authentication features as security is primary concern of any application. So, we planned to complete SR2: User authentication features (Section 3.2 of deliverable-3.pdf) like User register, Login. The functional requirements developed in this phase are:

- FR2.1: User registration (Section 3.2.1)
- FR2.2: User login (Section 3.2.2)
- FR1.4: Feature to read the SMS command and alter the ringer volume (Section 3.1.4)
- FR 3.1: Feature to read the SMS command and turn on/off WIFI (Section 3.3.1)
- FR 3.9: Feature to read the SMS command and turn on/off Bluetooth (New feature)

After this phase, user must register/login to use Ring me application.

1.1 SR2: User authentication requirements

User authentication features aim to collect the user personal information by registration in the application. This feature allows user to Register, Login and Edit his personal details. Feature hierarchy of User authentication requirements is shown in figure 1.1.

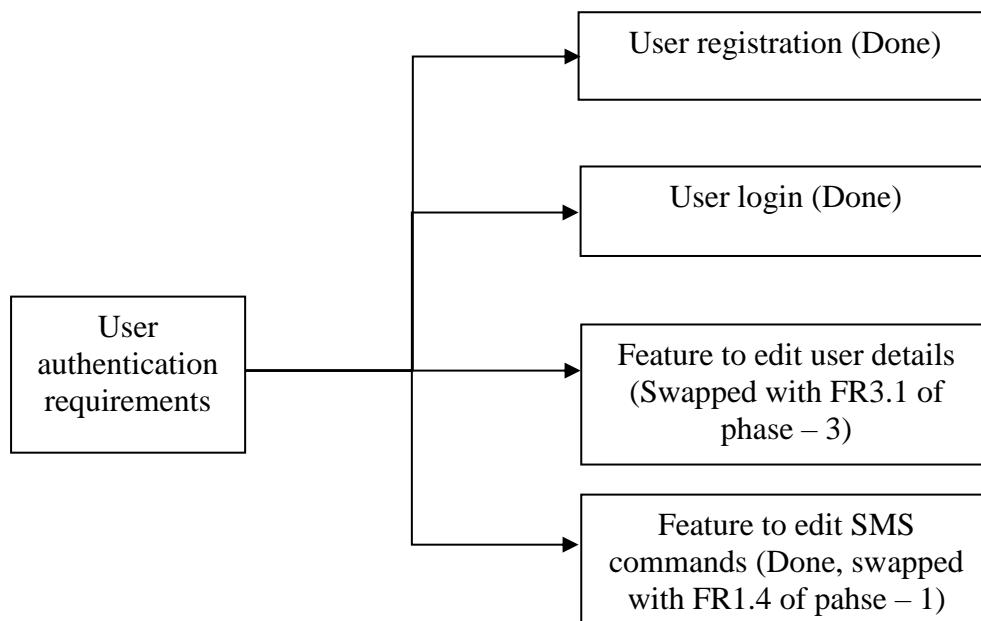


Figure 1.1. Feature hierarchy of User authentication requirements

1.2 FR2.1: User registration

New user needs to register for using Ring me application. The registration details of the user are important as a part of application security. Users should be able to register using their email ID and phone number. All users' needs to provide details

like - name, email ID, password and contact number. All users' information will be their stored in the User details table in the database. Upon successful registration, the user will be logged into the application and redirected to Application Home screen.

1.3 FR2.2: User Login

Every signed-up user needs to login before using the application every time. User should provide a valid email/phone number and his password given at the time of registration. When login details for a user are entered, they are validated by the application with database. If the details entered are matched, the user will be able to application home page; else a failure prompt will be displayed.

1.4 FR1.4: Feature to read SMS command and alter the ringer volume

Android has a feature to increase or decrease its ringer volume. This feature is important as there will few situations where user can keep neither in high ringer volume nor in silent. This feature is to alter the ringer volume of his phone to maximum or minimum based on the need. User will be able to select these functions 'Ringer high' and 'Ringer low' and configure a SMS commands against them. This command will be stored in our database. These commands will be used by the application to identify and respond accordingly by altering the ringer volume.

1.5 FR3.1: Feature to read SMS command and turn on/off Wi-Fi

Wi-Fi in a mobile phone is used to connect to internet. If user wants to turn on/off his Wi-Fi for various reasons, he can do it using this function. There are many applications which needs Wi-Fi to be turned on to locate the phone. If user phone is lost and he wants to turn on his Wi-Fi, he can do it by making use Ring me application SMS commands feature. This feature supports turn on and turn off Wi-Fi based on the command received in the SMS. User will be able to select these functions 'WIFI ON' or 'WIFI OFF' and configure a SMS command against them. These commands will be stored in our database. These commands will be used by the application to identify and respond accordingly by turning on/off the Wi-Fi in the remote mobile phone.

1.6 FR3.1: Feature to read SMS command and turn on/off Bluetooth

Bluetooth is a wireless technology standard for exchanging data over short distances. [1] If user wants to turn on/off his Bluetooth for various reasons, he can do it using this function. There are many applications which needs Bluetooth to be turned on to locate the phone. If user phone is lost and he wants to turn on his Bluetooth, he can do it by making use Ring me application SMS commands feature. This feature supports turn on and turn off Bluetooth based on the command received in the SMS. User will be able to select these functions 'Bluetooth ON' or Bluetooth OFF' and configure a SMS command against them. These commands will be stored in our database. These commands will be used by the application to identify and respond accordingly by turning on/off the Bluetooth in the remote mobile phone.

1.7 Updated requirements

We have moved the '*FR2.3: Feature to edit user details* (Section 3.2.3)' to development phase III and swapped with '*FR3.1: Feature to read the SMS command and turn on/off WIFI* (Section 3.3.1)' as edit user details requires lot of GUI changes to be done. Due to time constraint, this requirement is moved to phase III. The Table 1.1 Phase II development progress table explain further about the phase progress.

Planned features (4)	Developed features (5)	Re-planned (1)	New feature/s(1)
FR 2.1 FR 2.2 FR 2.3 FR 1.4	FR 2.1 FR 2.2 FR 1.4 FR 3.1 FR 3.9	FR 2.3	FR 3.9: Feature to read the SMS command and turn on/off Bluetooth (New feature)

Table 1.1. Phase II development report

2. UML Design

2.1 Class Diagram

One user registers, for the first time, it will ask to login with email and password (LoginActivity). Then user needs to create login PIN, which will be used for further logins (PinPadActivity). Each time when user logs in or SMS command is received (MyReceiver), application check with database for authentication (DatabaseHelper). The class diagram for phase II is depicted using Figure 2.1.

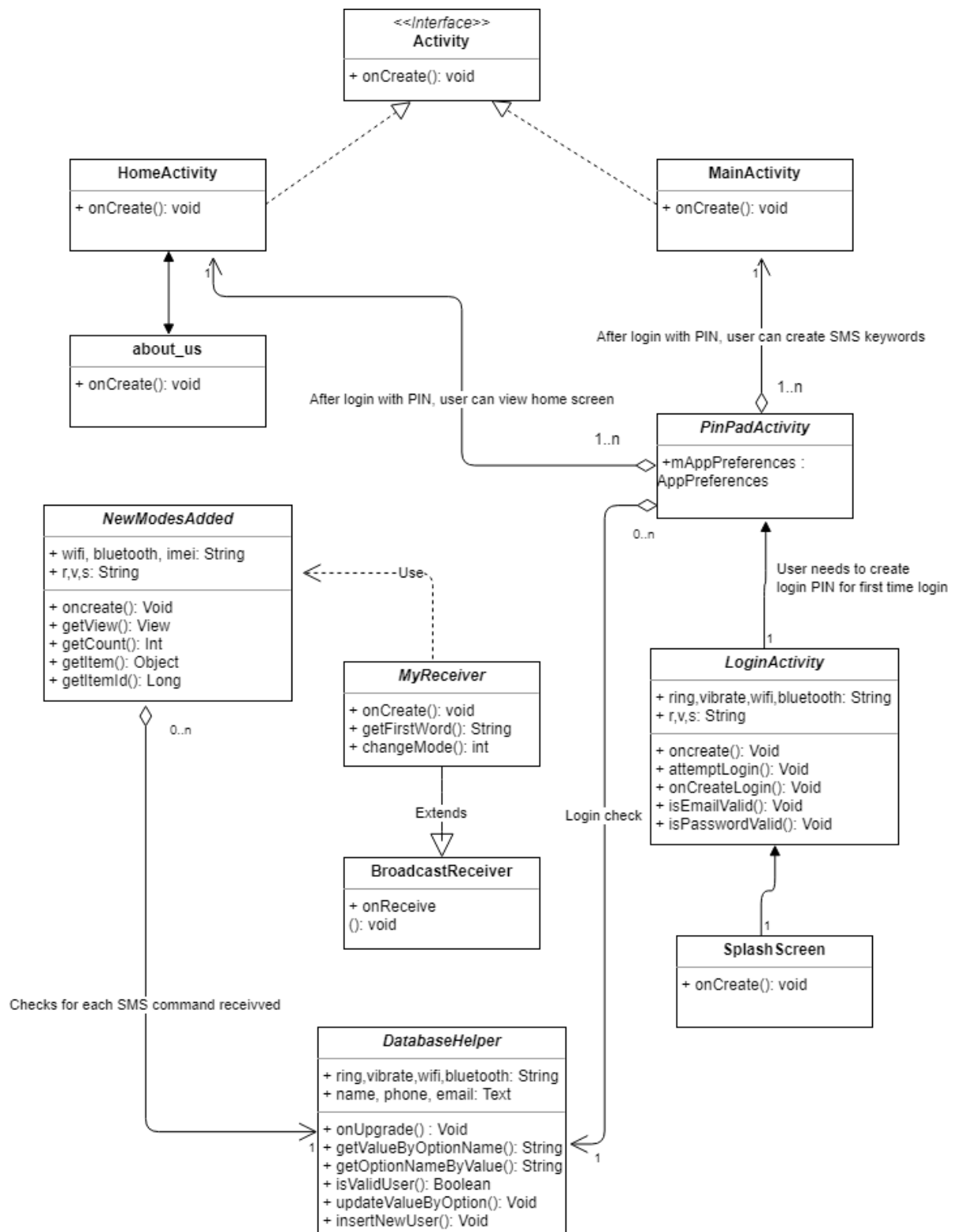


Figure2.1. Class diagram for phase-II requirements

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:

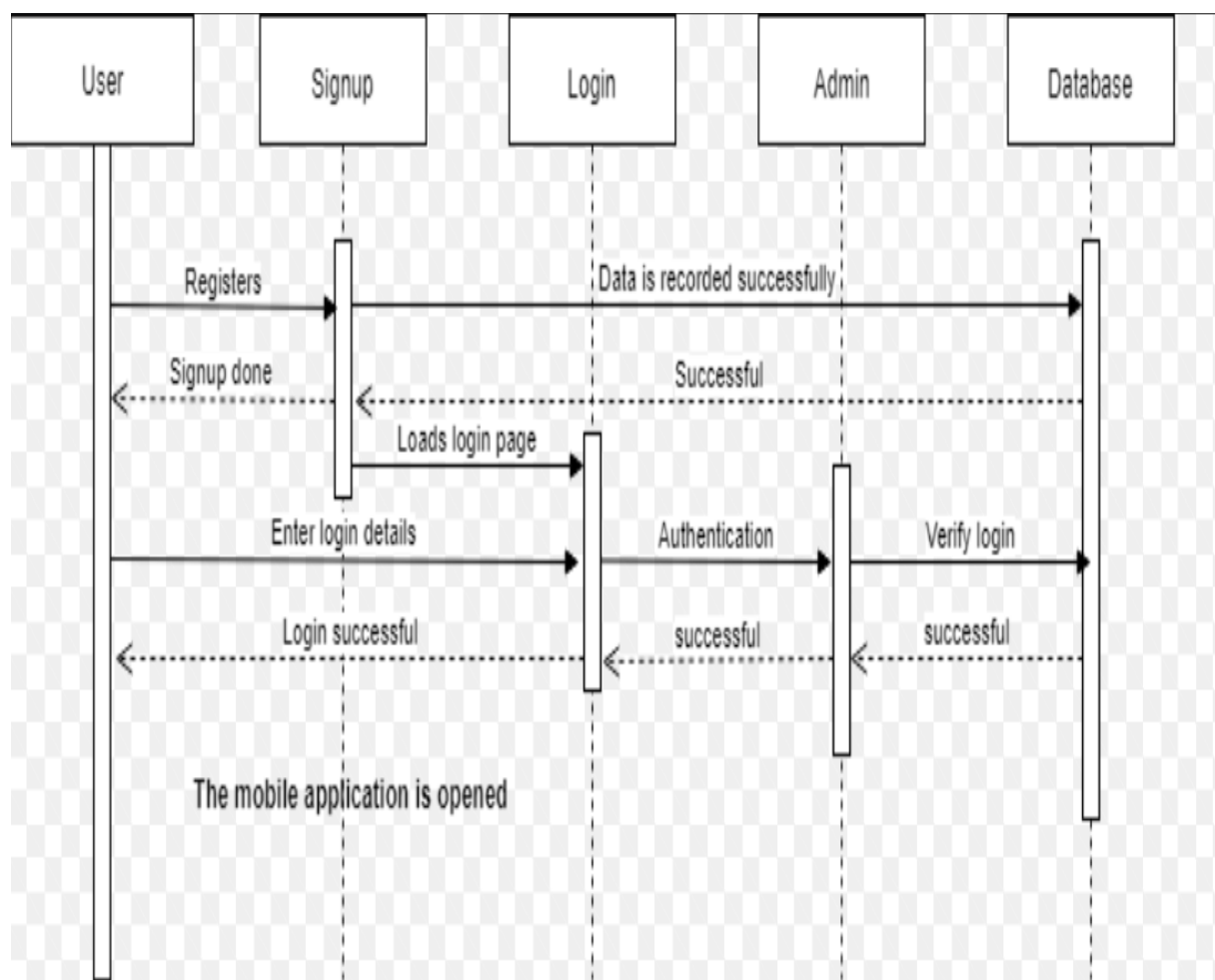


Figure 2.2 Normal case sequence diagram for Phase-II requirements

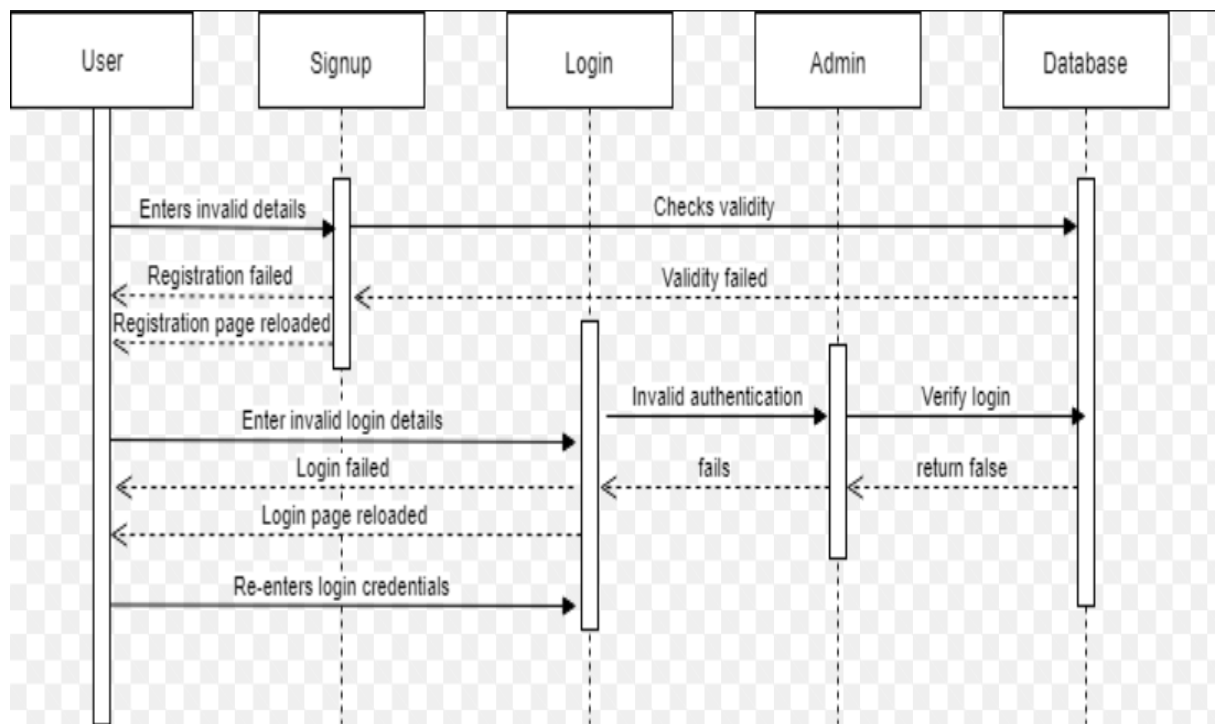
Error case:

Figure 2.3 Error case sequence diagram for Phase-II 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.

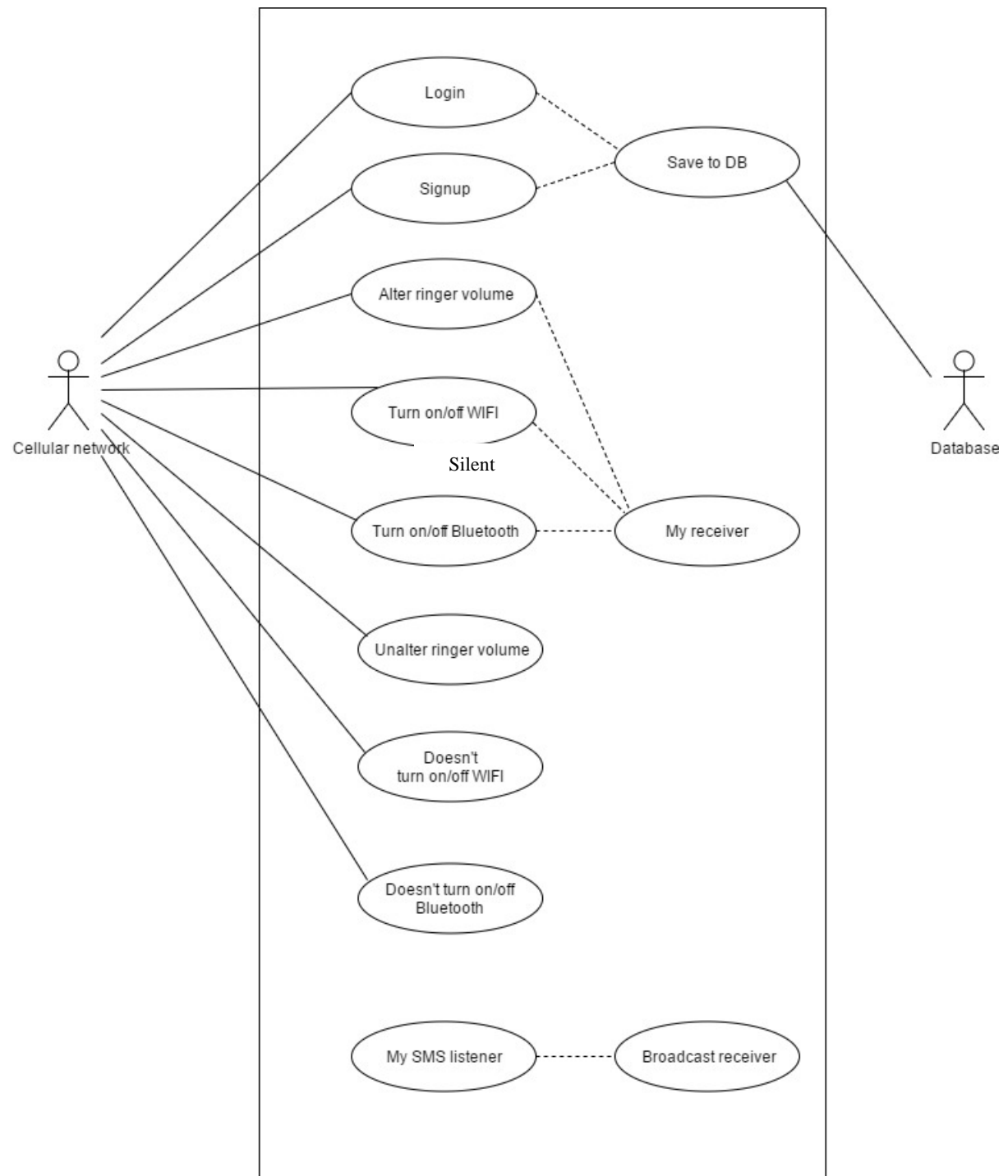


Figure 2.4 Use case diagram for Phase-II requirements

3. Test Cases

3.1 Unit test cases report

Phase II requirements are focused on User authentication – Registration and Login. Hence efforts are made to identify more unit cases around these requirements. A total of 33 Unit test cases were found and tested.

<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 II requirements is shown in figure 3.1

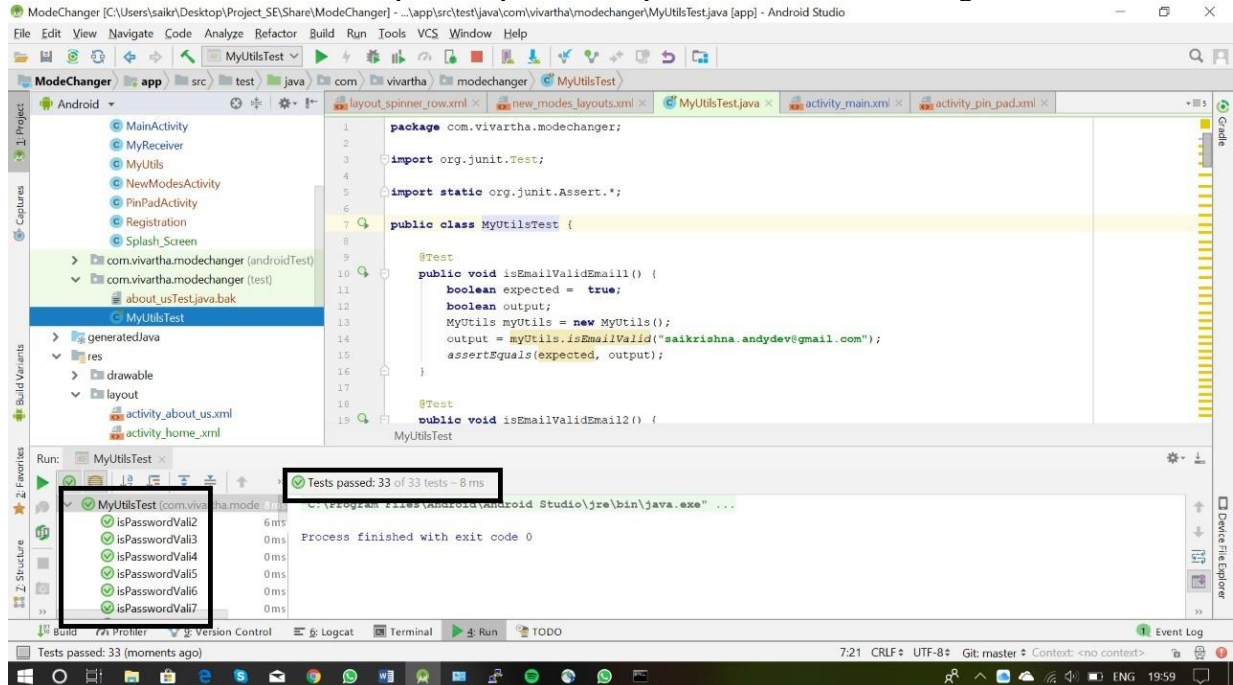


Figure 3.1 Unit test cases execution report for phase II requirements.

3.2 Unit test cases for classes

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

```
package com.vivarthamodechanger;
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 isPasswordValid2() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
```

```
        boolean output =
myUtils.isPasswordValid("sAikrishna@123");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid3() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saikrishna#123");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid4() {
        boolean expected = true;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saiKrishna#0");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid5() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output = myUtils.isPasswordValid("");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid6() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("saikrishna");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid7() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("123456789");
        assertEquals(expected, output);
    }
    @Test
    public void isPasswordValid8() {
        boolean expected = false;
        MyUtils myUtils = new MyUtils();
        boolean output =
myUtils.isPasswordValid("MSDHONI#123");
        assertEquals(expected, output);
    }
```

```
    }
    @Test
    public void isPasswordValid9() {
        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
FR2.1: User registration	Sai Teja Reddy Malle
FR2.2: User login	Sai Krishna Kolli
FR1.4: Feature to read the SMS command and alter the ringer volume	Revanth Malreddy
FR3.1: Feature to read the SMS command and turn on/off WIFI	Siri Gogineni
FR3.9: Feature to read the SMS command and turn on/off Bluetooth	Vikas Chirumamilla

4.2 Components/Classes

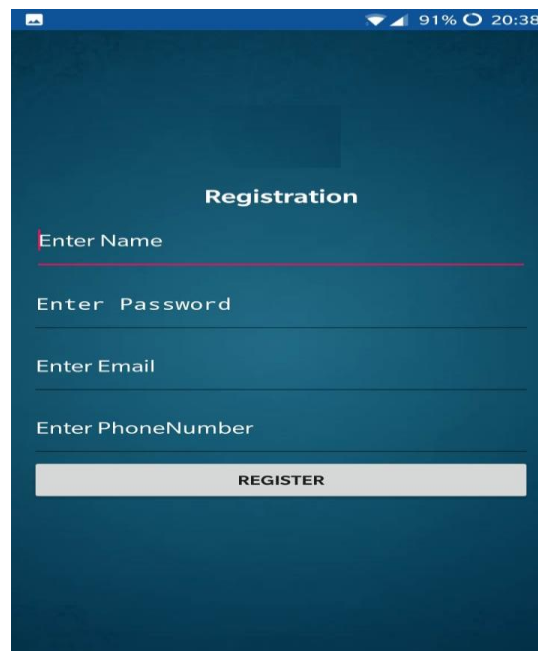
Name	Component/Classes
Vikas Chirumamilla	Home_activity.java, activity_home.xml, Home_ActivityTest.java, MyUtils.java, MyUtilsTest.java, PinPadActivity.java
Sai Krishna Kolli	Activity_pin_pad.xml, ActivityLogin.java, LoginActivityTest.java, DatabaseHelper.java, ApplicationController.java
Revanth Malreddy	MainActivity.java, activity_main.xml, MainActivityTest.java, AppPreferences.java
Siri Gogineni	MyReceiver.java, activity_login.xml, MyReceiverTest.java
Sai Teja Malle	Registration.java, Activity_registration.xml

5. User Manual

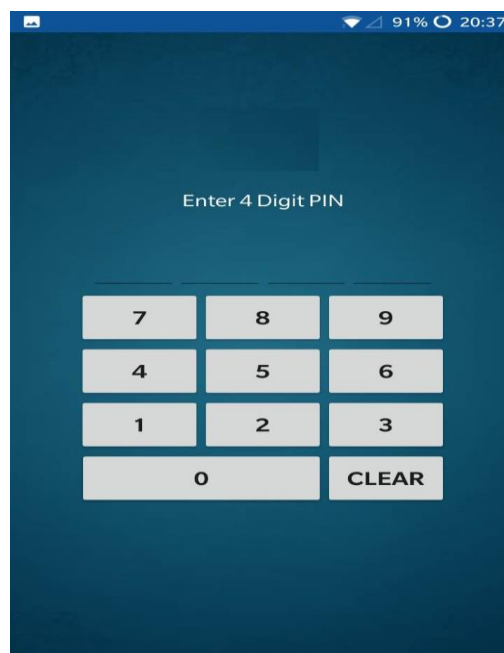
- This is how the splash screen of our mobile management Android application is going to look like. As soon as we click on this application, the following screen open first.



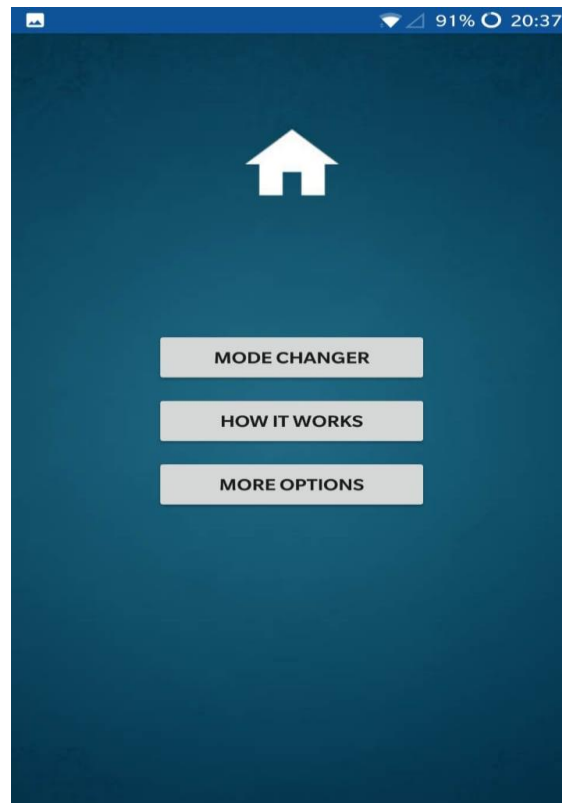
- For, security reasons, it is important that every user should create an account in this application. The user when opens the app for the first time, he is taken to the “Registration” page where he needs to fill the details. The details include Enter Name, Enter Password, Enter Email, Enter Phone Number.

A screenshot of a mobile application's registration screen. The background is a dark blue gradient. At the top, there's a status bar with a battery icon, signal strength, 91% battery, and the time 20:38. Below the status bar, there's a small logo. The title "Registration" is centered in white. Below the title, there are four input fields with labels "Enter Name", "Enter Password", "Enter Email", and "Enter PhoneNumber". Each field has a red underline. At the bottom, there is a white button with the text "REGISTER" in black.

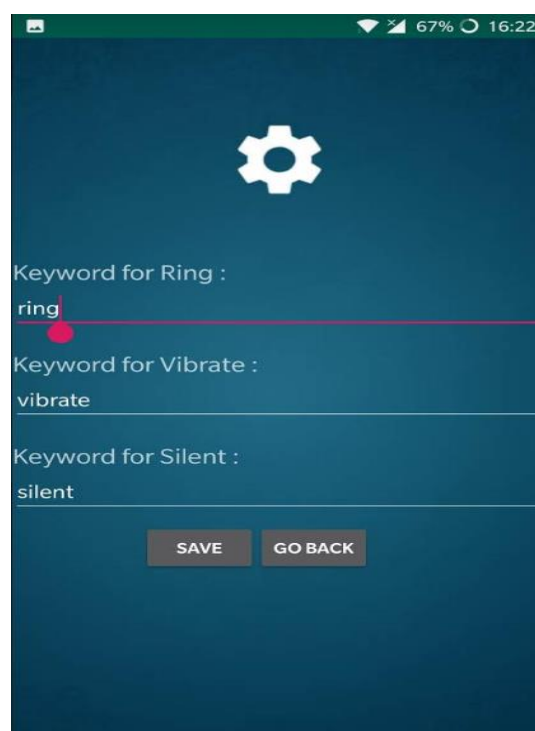
- After registering, the user needs to choose and enter the PIN with which he can access the application. The user needs to re-enter the PIN for confirmation.
- After choosing the PIN, it redirects to login where user needs to enter PIN for application to be launched.

A screenshot of a mobile application's PIN entry screen. The background is a dark blue gradient. At the top, there's a status bar with a battery icon, signal strength, 91% battery, and the time 20:37. Below the status bar, there's a small logo. The title "Enter 4 Digit PIN" is centered in white. Below the title, there is a numeric keypad with buttons for digits 0-9 and a "CLEAR" button. The buttons are arranged in a 4x3 grid: the first three rows have digits 7-9, 4-6, and 1-3 respectively, and the fourth row has 0 and CLEAR.

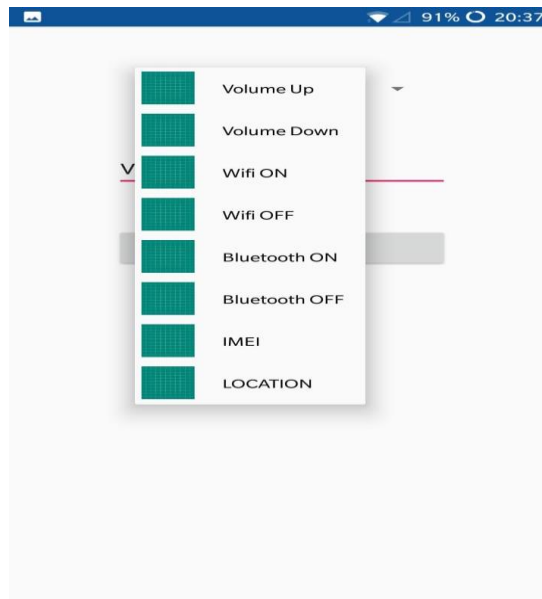
- After logging to application, it takes user to the home screen where there are three options provided.
Mode changer
How it works
More Options



- When you select Mode changer option, it takes you to a screen where you have the option to edit the keyword which you want to use for ring, vibrate and silent. In this Development Phase- 1, we have only developed the code for these options. In the later development phases, we will have more requirements filled in.
- You can edit the keyword which you would like to use and click on save.
- There is another option 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.



- When the user selects the 'More Options', a dropdown menu appears where user can choose different options like Volume Up, Volume Down, Wifi OFF, Wifi ON etc. User can choose a different keyword for the respective option.

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/vivarthamodechanger>
- 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>