**Development phase I 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

10/29/2018

# Table of Contents

1. [SR1: Mode Change requirements 3](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250027)
   1. [FR1.1: Feature to read the SMS command and change phone mode to ringer mode 3](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250026)
   2. [FR1.2: Feature to read the SMS command and change phone mode to vibration mode .. 3](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250025)
   3. [FR1.3: Feature to read the SMS command and change phone mode to silent mode 3](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250024)
   4. FR1.5: [Application Home Screen](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250023) 3
   5. [FR1.6: Updated requirements 4](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250023)
2. [UML Design](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250022) 4
   1. [Class Diagram](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250021) 4
   2. [Sequence Diagram 6](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250020)
   3. [Use case Diagram working model 7](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250019)
   4. [Use case Diagram Error case](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250018) 8
3. [Test Cases](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250016) 9
   1. [Test Cases for FR1.1](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015) 9
   2. [Test Cases for FR1.2 9](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015)
   3. [Test Cases for FR1.3](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015) 9
   4. [Test Cases for FR2.4](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015) 9
   5. [Test Cases for FR1.5](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015) 10
4. [Contribution 1](file:///C:\\Users\\Revanth\\Downloads\\Deliverable-2.docx" \l "_TOC_250011)5
   1. [Requirements](file:///C:\\Users\\Revanth\\Downloads\\Deliverable-2.docx" \l "_TOC_250015) 15
   2. [Components/Classes](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250015) 15
5. [User Manual 1](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250011)6
6. Installation [instructions 1](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250011)9
7. [Peer review session feedback](file:///C:\Users\Revanth\Downloads\Deliverable-2.docx#_TOC_250011) 20

# SR1: Mode changer requirements

Mode changer features allows user to control ringer modes of the mobile phone. In Android we have three basic ringer modes namely Ringer, Vibration and Silent. This feature aims to develop the functionalities to read the SMS command and alter the ringer modes. This also includes the development of application home screen and alter the ringer volume. Feature hierarchy of Mode changer features is shown in figure 3.2.

# FR1.1: Feature to read the SMS command and change phone mode to ringer mode

Ringer mode is any mobile phone alerts the user with sound by which user will come to know about his incoming phone calls or notifications. This feature is to change the mode of his phone to Ringer as a response to the command received in the SMS. User will be able to select this function ‘Ring’ and configure a SMS command under it. This command will be stored in our database. This command will be used by the application to identify and respond accordingly by change the mode of the phone to ringer.

# FR1.2: Feature to read the SMS command and change phone mode to vibration mode

Vibration mode does not alert the user with any sound but vibrates the phone for all incoming phone calls and notifications. This feature is to change the mode of his phone to Vibrate as a response to the command received in the SMS. User will be able to select this function ‘Vibration’ and configure a SMS command against it. This command will be stored in our database. This command will be used by the application to identify and respond accordingly by change the mode of the phone to vibration.

# FR1.3: Feature to read the SMS command and change phone mode to silent mode

Silent mode in a mobile phone is like DND (Do Not Disturb) mode in which user is not alerted by any sound or vibration for incoming phone calls and notifications. This feature is to change the mode of his phone to Silent as a response to the command received in the SMS. User will be able to select this function ‘Silent’ and configure a SMS command against it. This command will be stored in our database. This command will be used by the application to identify and respond accordingly by change the mode of the phone to silent.

# FR1.5: Application Home screen

This is Home GUI screen will allow user to create new SMS commands for Ring me application. User will be displayed with all the featured supported by Ring me. User will be displayed with all the commands configured by him in this Home screen page.

* 1. **Updated requirements**

We have moved the *‘FR1.4: Feature to read the SMS command and alter the ringer volume (Section 3.1.4)’* to development phase II and swapped *‘FR2.4: Feature to edit SMS commands (Section 3.2.4)’* as edit requirement should be integrated for all the requirements so that user can add and edit SMS commands. Hence this requirement is needed to be prioritized. This is the reason why we have swapped *FR1.4 (Section 3.1.4)* with *FR2.4 (Section 3.2.4)*

The development phases have been re-planned accordingly as below.

**Development Phase I: (Deadline – 10/24/2018)**

In this phase we concentrate on the main functionality which we decided to prioritize. Mode changer features are like base features which needs to be supported from the first. Hence, we have decided to complete Mode changer features (Section 3.1) first. This phase would be intended to develop Mode changer features in full stack. The functional requirements which are implemented here are:

• FR1.1: Feature to read the SMS command and change phone mode to ringer mode (Section 3.1.1)

• FR1.2: Feature to read the SMS command and change phone mode to vibrate mode (Section 3.1.2)

• FR1.3: Feature to read the SMS command and change phone mode to silent mode (Section 3.1.3)

• FR2.4: Feature to edit SMS commands (Section 3.2.4)

• FR1.5: Application Home screen (Section 3.1.5)

**Development Phase II: (Deadline – 11/07/2018)**

In this phase we target to finish user authentication features as security is primary concern of any application. So, we shall complete User authentication features (Section 3.2) like User register, Login and edit personal data. The functional requirements in this phase are:

• FR2.1: User registration (Section 3.2.1)

• FR2.2: User login (Section 3.2.2)

• FR2.3: Feature to edit user details (Section 3.2.3)

• FR1.4: Feature to read the SMS command and alter the ringer volume (Section 3.1.4)

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

1. **UML Design**
   1. **Class Diagram**

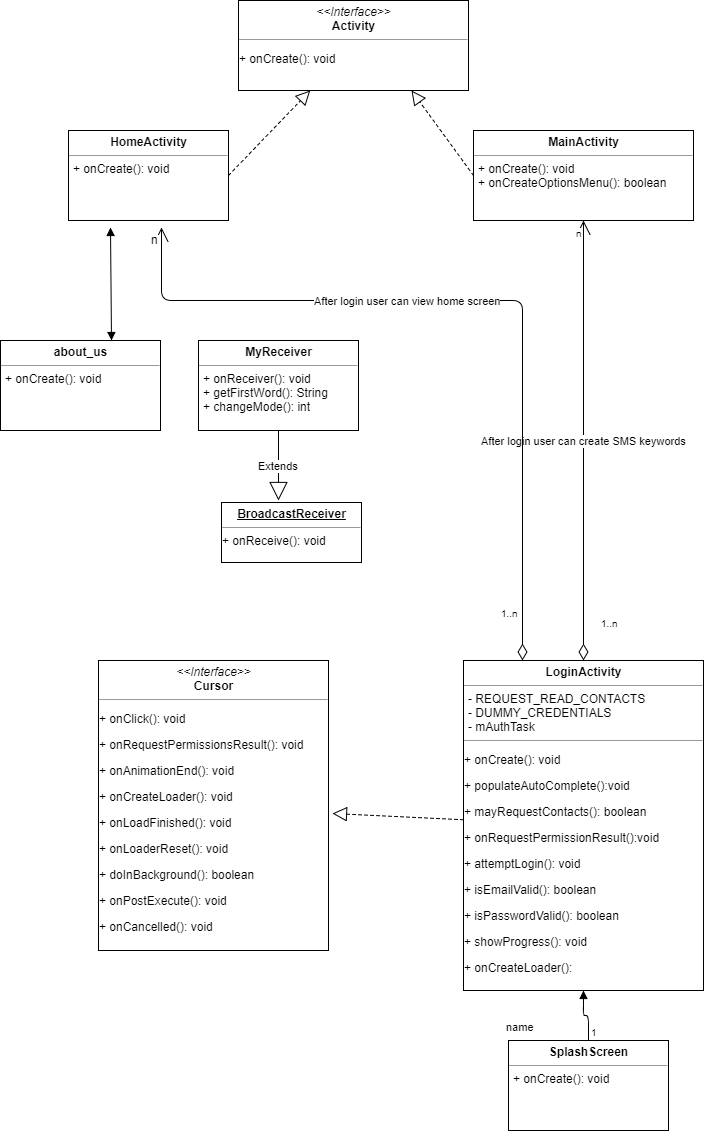
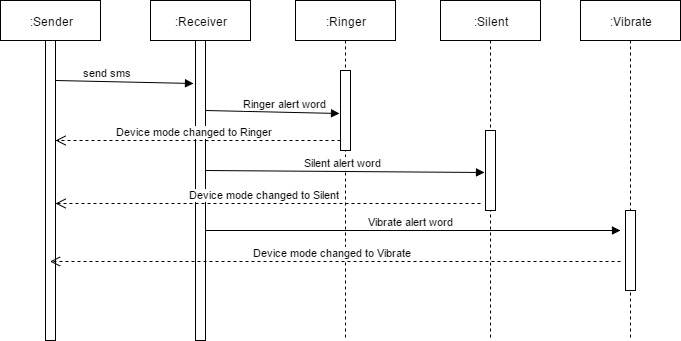
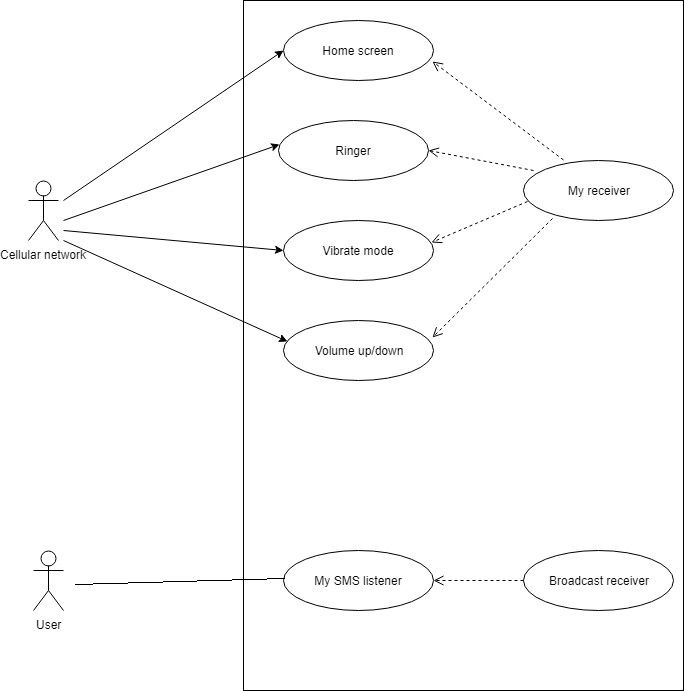
****

Figure2.1. Class diagram for phase-I requirements

* 1. **Sequence Diagram**

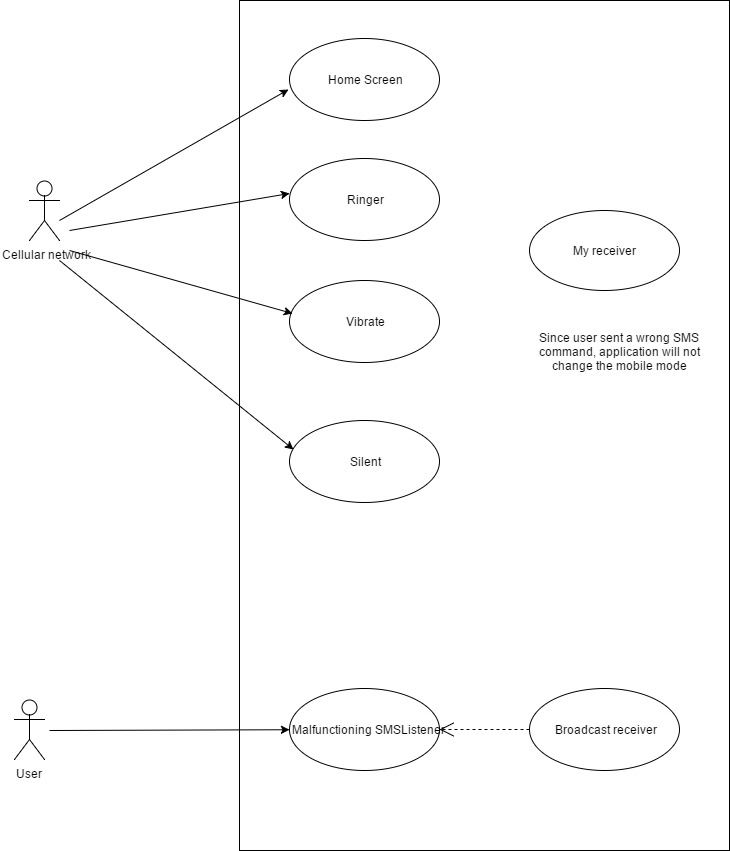
****

* 1. **Use Case Diagram**

****

Silent

* 1. **Use Case Diagram Error case**

****

1. **Test Cases**

[**https://github.com/Sai11262246/ModeChanger/tree/master/Test%20cases**](https://github.com/Sai11262246/ModeChanger/tree/master/Test%20cases)

* 1. **Functional Test cases for FR1.1\_Feature to read the SMS command and change phone mode to ringer mode**



* 1. **Functional Test Cases for FR1.2\_Feature to read the SMS command and change phone mode to vibrate mode**



* 1. **Functional Test cases for FR1.3\_Feature to read the SMS command and change phone mode to silent mode**



* 1. **Functional Test Cases for FR2.4\_Feature to edit SMS commands**



* 1. **Functional Test Cases for FR1.5\_Application Home screen**



* 1. **Unit test cases for classes**
* **Home\_ActivityTest: Junit code for Home\_Activity.java class**

**package** com.vivartha.modechanger;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.design.widget.FloatingActionButton;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** org.junit.Test;  
  
**public class** Home\_ActivityTest{  
   
 @Test  
 **public void** testOnCreate()  
 {  
 Home\_Activity act = **new** Home\_Activity();  
 Bundle savedInstanceState = **new** Bundle();  
 act.onCreate(savedInstanceState);  
 }  
}

* **LoginActivityTest: Junit code for LoginActivity.java class**

**package** com.vivartha.modechanger;  
  
**import** android.animation.Animator;  
**import** android.animation.AnimatorListenerAdapter;  
**import** android.annotation.TargetApi;  
**import** android.content.Intent;  
**import** android.content.pm.PackageManager;  
**import** android.database.sqlite.SQLiteCursor;  
**import** android.database.sqlite.SQLiteCursorDriver;  
**import** android.support.annotation.NonNull;  
**import** android.support.design.widget.Snackbar;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.app.LoaderManager.LoaderCallbacks;  
  
**import** android.content.CursorLoader;  
**import** android.content.Loader;  
**import** android.database.Cursor;  
**import** android.net.Uri;  
**import** android.os.AsyncTask;  
  
**import** android.os.Build;  
**import** android.os.Bundle;  
**import** android.provider.ContactsContract;  
**import** android.text.TextUtils;  
**import** android.view.KeyEvent;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.view.inputmethod.EditorInfo;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.AutoCompleteTextView;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.TextView;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** org.junit.Test;  
  
**import static** android.Manifest.permission.***READ\_CONTACTS***;  
  
*/\*\*  
 \* A login screen that offers login via email/password.  
 \*/***public class** LoginActivityTest {  
   
 @Test  
 **public void** testOnCreate()  
 {  
 LoginActivity act = **new** LoginActivity();  
 Bundle savedInstanceState = **new** Bundle();  
 act.onCreate(savedInstanceState);  
 }  
  
 @Test  
 **public void** testOnRequestPermissionsResult()  
 {  
 **int** requestCode = 1;  
 String[] permissions = {**"1"**,**"0"**,**"1"**};  
 **int**[] grantResults = **new int**[]{1,2,3};  
 *//grantResults = {1,2,3};* LoginActivity act = **new** LoginActivity();  
 act.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 }   
   
 @Test  
 **public void** testAttemptLogin()  
 {  
 LoginActivity act = **new** LoginActivity();  
 *//act.attemptLogin();* }  
   
 @Test  
 **public void** testOnCreateLoader()  
 {  
 LoginActivity act = **new** LoginActivity();  
 Bundle bundle = **new** Bundle();  
 act.onCreateLoader(1,bundle);  
 }  
   
 @Test  
 **public void** testOnLoadFinished()  
 {  
 Loader<Cursor> cursorLoader=**new** Loader<Cursor>(**null**);  
 Cursor cursor = **new** SQLiteCursor(**null**, **null**, **null**);  
 *// // public SQLiteCursor (SQLiteCursorDriver driver, String editTable, SQLiteQuery query)* LoginActivity act = **new** LoginActivity();  
 act.onLoadFinished(cursorLoader, cursor);  
   
 }  
  
}

* **MainActivityTest: Junit code for MainActivity.java class**

**package** com.vivartha.modechanger;  
  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.os.Bundle;  
**import** android.view.ContextMenu;  
**import** android.view.Menu;  
**import** android.view.SubMenu;  
**import** android.view.View;  
**import** android.view.View.OnClickListener;  
**import** android.widget.Button;  
**import** android.widget.EditText;  
**import** android.widget.Toast;  
  
**import** org.junit.Test;  
  
**public class** MainActivityTest {  
   
 @Test  
 **public void** testOnCreate()  
 {  
 MainActivity act = **new** MainActivity();  
 Bundle savedInstanceState = **new** Bundle();  
 act.onCreate(savedInstanceState);  
 }  
   
 @Test  
 **public void** testOnCreateOptionsMenu()  
 {  
 MainActivity act = **new** MainActivity();  
 *//Menu menu = PowerMockito.mock(Menu.class);* Menu menu = **null**;  
 **boolean** result = act.onCreateOptionsMenu(menu);  
 }   
  
}

* **MyReceiverTest: Junit code for MyReceiver.java class**

**package** com.vivartha.modechanger;  
  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.ContextWrapper;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.media.AudioManager;  
**import** android.os.Bundle;  
**import** android.telephony.SmsMessage;  
**import** android.widget.Toast;  
  
**import** org.junit.Test;  
  
**public class** MyReceiverTest{  
   
 @Test  
 **public void** testOnReceive()  
 {  
 MyReceiver obj = **new** MyReceiver();  
 Context context = **new** ContextWrapper(**null**);  
 Intent intent = **new** Intent();  
*// obj.onReceive(context, intent);* }  
  
}

* **Splash\_ScreenTest: Junit code for Splash\_Screen.java class**

**package** com.vivartha.modechanger;  
  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.app.Activity;  
  
**import** org.junit.Test;  
  
**public class** Splash\_ScreenTest {  
   
 @Test  
 **public void** testOnCreate()  
 {  
 Splash\_Screen obj = **new** Splash\_Screen();  
 Bundle savedInstanceState = **new** Bundle();  
 obj.onCreate(savedInstanceState);  
 }  
  
}

--- nn

**package** com.vivartha.modechanger;  
  
**import** android.os.Bundle;  
**import** android.app.Activity;  
**import** org.junit.Test;  
**import static** org.junit.Assert.\*;  
**import** org.junit.runner.RunWith;  
  
**public class** about\_usTest{  
   
 @Test  
 **public void** testOnCreate()  
 {  
 about\_us obj = **new** about\_us();  
 Bundle savedInstanceState = **new** Bundle();  
 obj.onCreate(savedInstanceState);  
   
 }  
}

1. **Contribution**
   1. **Requirements**

|  |  |
| --- | --- |
| **Contribution** | **Developer name** |
| FR1.1\_Ringer mode requirement | Vikas Chirumamilla |
| FR1.5\_Application home screen | Sai Krishna Kolli |
| FR1.3\_Silent mode requirement | Revanth Malreddy |
| FR1.2\_Vibrate mode requirement | Siri Gogineni |
| FR2.4\_Edit SMS commands requirement | Sai Teja Reddy Malle |

* 1. **Components/Classes**

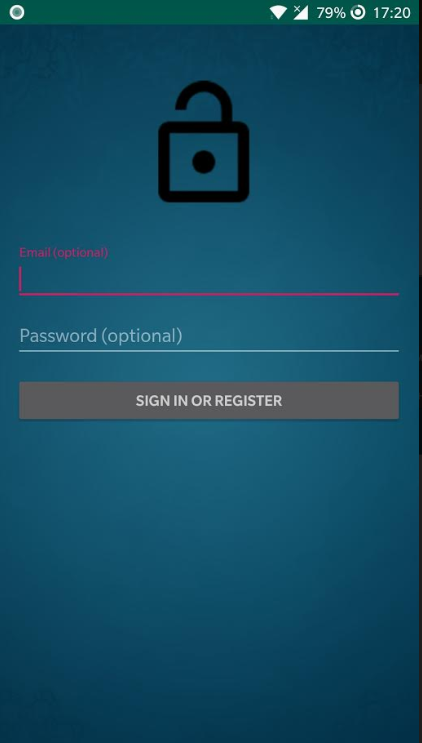
|  |  |
| --- | --- |
| **Name** | **Component/Classes** |
| Vikas Chirumamilla | Home\_activity.java, activity\_home\_.xml, Home\_ActivityTest.java |
| Sai Krishna Kolli | Android base framework setup, SpalshScreen.java, activity\_splash\_screen.xml, ActivityLogin.java, LoginActivityTest.java |
| Revanth Malreddy | MainActivity.java, activity\_main.xml, MainActivityTest.java |
| Siri Gogineni | MyReceiver.java, activity\_login.xml, MyReceiverTest.java |
| Sai Teja Malle | about\_us.java, activity\_about\_us.xml, about\_usTest.java |

1. **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. Since, we have done only the Development Phase- 1, we have not yet authenticated the sign-in and register options. You can see this in the screenshot below. We have provided optional option for this phase.
* As soon as we select the sign in option, our application takes you to another screen and starts working.



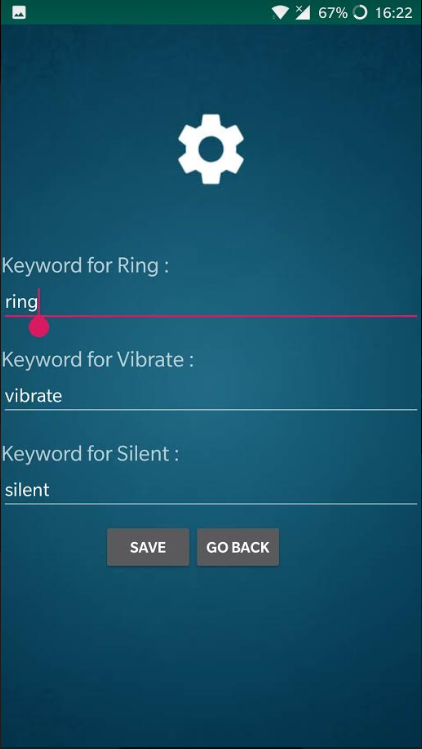
* It takes you to the home screen, where we have provided you with two options:

Mode changer

How it works



* 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, you 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.

1. **Installation instructions**
2. 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.
3. 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.
4. 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>



* + 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>

1. **Peer review session feedback**

We have received 2 feedback points from our peer team.

1. Table of contents are not up to 3rd level:

We have first included table of contents for 3rd level headings. Later in peer review received feedback to include up to 3rd level which will be easier for people to navigate.

1. Requirements not clear:

FR3.3.3, FR3.3.5 are not clearly mentioned. Also asked to check the feasibility of developing these requirements at Android system level.