Locale-Reminder: An Android Application

GRADUATE PROJECT

Submitted to the Faculty of
The School of Engineering & Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, TX

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Gopal Rao Gurram
Spring 2015

Committee Members

Dr. Ajay Katangur
Committee Chairperson

Dr. Dulal Kar
Committee Member
ABSTRACT

Electronic reminders based on calendars in smart phones are effective and popular, however such reminders are generally activated by time and not by location. In many situations tasks are only meaningful to be performed at a particular location, so it would be useful if reminders for those tasks can be activated when the person to be reminded is physically near or located at that location. In this project, a new application (app) is developed for Android-based smartphones and tablets. With Locale-Reminder app the users can store their tasks and associate them to a particular location, and when the user passes close to it the app reminds them of the task. The app has basic Time Based Reminder as an additional feature. The app alerts users when they are near their location and gives an option to navigate that location using the directions feature. In addition, this app has a ‘get-my-location’ feature, which helps users to set a reminder for unknown locations and to save the location coordinates for creating reminders in the future. Another feature of the app is to share the location, which will help the user to share his/her current location or saved location via messaging or email service. Unlike any other location reminder apps this app has a Safe Driving Mode feature, which helps the user to completely focus on driving, by automatically sending a text message to the caller that the user is driving. The app also alerts users by voice based reminders.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Background and Rationale</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Android</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Android Studio</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Fluid UI</td>
<td>3</td>
</tr>
<tr>
<td>1.4</td>
<td>Location Based Services</td>
<td>3</td>
</tr>
<tr>
<td>1.5</td>
<td>Location Reminders</td>
<td>6</td>
</tr>
<tr>
<td>1.6</td>
<td>Existing Apps</td>
<td>6</td>
</tr>
<tr>
<td>1.7</td>
<td>Solution-Locale-Reminder</td>
<td>7</td>
</tr>
<tr>
<td>2.</td>
<td>Narrative</td>
<td>9</td>
</tr>
<tr>
<td>2.1</td>
<td>Problem Statement</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Motivation</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Project Objective</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Project Scope</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Functionalities of the Project</td>
<td>11</td>
</tr>
</tbody>
</table>
3. Proposed System Design ................................................................. 13
   3.1 Use Case Diagram ................................................................. 14
   3.2 Class Diagram .................................................................... 15
   3.3 Sequence Diagram .............................................................. 17
   3.4 User Interface ....................................................................... 17

4. Implementation of the Application Modules ........................................ 26
   4.1 User Interface Implementation of Locale-Reminder .................... 26
   4.2 Setting Permissions in AndroidManifest.xml file .................... 26
   4.3 Connecting SQLite with Locale-Reminder ............................... 27
   4.4 Location Based Reminder ...................................................... 28
   4.5 Time based reminder ............................................................. 32
   4.6 Voice Reminder ................................................................... 33
   4.7 Get My Location .................................................................... 33
   4.8 Safe Driving Mode ............................................................... 36

5. Testing and Evaluation .................................................................. 37
   5.1 Launching the application ..................................................... 37
   5.2 Time Based Reminder ............................................................ 38
   5.3 Location Based Reminder ...................................................... 43
List of Figures

Figure 1.1 Access to Location Information [11] .......................................................... 5

Figure 3.1 Schematic diagram of the proposed Locale-Reminder application ..................14

Figure 3.2 Locale-Reminder Use case Diagram ..........................................................15

Figure 3.3 Locale-Reminder Class Diagram ..............................................................16

Figure 3.4 Sequence Diagram for User interaction with application............................17

Figure 3.5 Home Screen of Locale-Reminder app .......................................................18

Figure 3.6 Time Based Reminder .............................................................................19

Figure 3.7 Location Based Reminder ........................................................................19

Figure 3.8 Time Based List .......................................................................................20

Figure 3.9 Location Based List ................................................................................21

Figure 3.10 Get My Location ...................................................................................22

Figure 3.11 View Saved Locations ...........................................................................23

Figure 3.12 Safe Driving Mode ...............................................................................24

Figure 3.13 Settings .................................................................................................25

Figure 4.1 Code snippet for setting up permissions in AndroidManifest.xml file ........27

Figure 4.2 Code snippet for creating tables ................................................................28

Figure 4.3 Snippet for places API key generated for the application .........................29
Figure 4.4 Code snippet for creating URL ................................................................. 30
Figure 4.5 Code snippet for parsing the JSON data .................................................. 30
Figure 4.6 Code snippet for showing the parsed data in a list view ............................ 31
Figure 4.7 Code snippet for creating location based reminder .................................... 32
Figure 4.8 Code snippet for creating time based reminder ......................................... 33
Figure 4.9 Code snippet for creating voice based reminder ........................................ 34
Figure 4.10 Code snippet for requesting location updates ........................................ 35
Figure 4.11 Code snippet for getting location coordinates ......................................... 35
Figure 4.12 Code snippet for answering incoming call and sending SMS ..................... 36
Figure 5.1 Launching the application .......................................................................... 37
Figure 5.2 Input validation for time based reminder .................................................... 38
Figure 5.3 Speech to text feature ................................................................................ 39
Figure 5.4 Creating a time based reminder ................................................................. 40
Figure 5.5 Generating notification for time based reminder ......................................... 41
Figure 5.6 Postponing the notification by one hour ..................................................... 42
Figure 5.7 Creating a reminder with repeat option ...................................................... 43
Figure 5.8 Input validation for location based reminder .............................................. 44
Figure 5.9 Input validation for date and time .............................................................. 44
Figure 5.10 Creating a location based reminder with Alert Radius as 2 miles..................45

Figure 5.11 Creating a location based reminder ..........................................................46

Figure 5.12 Creating a location based reminder with Alert Radius as 6 miles..............47

Figure 5.13 Generating the notification for location based reminder .........................48

Figure 5.14 Getting directions for the reminder location ............................................49

Figure 5.15 Creating a location based reminder with time range ...............................50

Figure 5.16 Automatically changing the reminder status to expired after the due date ...51

Figure 5.17 Creating reminder from the current location of the user using Get My Location feature ...........................................................................................................................................52

Figure 5.18 Notification generated for the reminder ....................................................53

Figure 5.19 Creating reminders from saved locations .................................................53

Figure 5.20 Automatically changing the reminder status to expired after the due date ...54

Figure 5.21 Deleting a reminder ..................................................................................55

Figure 5.22 Deleting a list of reminders ......................................................................56

Figure 5.23 Creating reminder for current location .....................................................57

Figure 5.24 Saving current location ............................................................................58

Figure 5.25 Sharing the current location of the user ...................................................59

Figure 5.26 List of saved locations ............................................................................60

Figure 5.27 Creating reminder for saved location .......................................................60
Figure 5.28 Getting directions for saved location ............................................................. 61
Figure 5.29 Sharing the saved location ........................................................................... 62
Figure 5.30 Safe Driving Mode with default message ..................................................... 63
Figure 5.31 Safe Driving Mode with custom message .................................................... 64
Figure 5.32 Settings activity ......................................................................................... 65
1. BACKGROUND & RATIONALE

In contemporary society, many people are overwhelmed by the number of tasks that need to be accomplished [1], these tasks are of different types, ranging from everyday based meetings at work, and non-daily based such as buying groceries, paying bills etc. To help ourselves to remember these tasks, basic practices are to take notes on the paper based day planners or use post-its or use a personal task management software on computers and/or smartphones. As smartphones are used as an alternative to personal computers, taking notes on smartphones is a more convenient choice [1].

Similar to programs on a personal computer, applications (app) can be downloaded and installed on a smartphone. Similar to a personal computer there exists a wide selection of proprietary and open source mobile operating system platforms. Out of these smartphone operating systems, the most prominent ones are Apple iOS, Google Android, Symbian from Symbian foundation, RIM blackberry OS and Windows mobile from Microsoft [2].

This project focuses on developing a mobile application, Locale-Reminder for Google’s Android operating system. The majority of smartphone manufacturers such as Samsung, Motorola and HTC have adopted Android as the operating system for their products. As indicated by App Brain statistics, there are more than 1.4 million applications accessible as of February 2015, of which approximately 1.3 million are free and 0.2 million are paid [15]. On July 24, 2013 Google declared that Play store had one million applications and had seen more than 50 billion downloads [3]. Consistently more
than 1 million new Android devices are activated daily around the world [3]. The number of downloads from google play store every month is more than 1.5 billion [3].

This project is developed for Android based smart phones on Android Studio, an IDE for developing applications on the Android platform. The user interface of this project is prototyped by using Fluid UI, a browser based wire framing and prototyping tool. This project utilizes Location Based Services for creating location based reminders. More on Android, Android Studio, Fluid UI and Location Based Services is discussed in sections 1.1, 1.2, 1.3 and 1.4.

1.1 Android

Android is a mobile operating system based on Linux. Android is primarily used for smartphones, tablet computers, watches, smart glasses, home appliances, cars, cameras, game consoles and mirrors. Android is available in 46 languages [3] and powers millions of mobile devices in more than 190 countries around the world [4]. Android provides a world class platform for developing apps and games for Android users everywhere, as well as an open market place for distributing them instantly [4]. The vast majority of Android applications are created in java, which is a standout amongst the most generally used programming languages around the world. Locale-Reminder is compatible on different versions of Android, as starting from the minimum SDK version of Android 4.0 (Ice Cream Sandwich) to recent update Android 5.0 (Lollipop). The development environment used for Locale-Reminder is Android Studio IDE.

1.2 Android Studio

Android Studio is an IDE for developing on the Android platform. It is free and open source software. The Intelligent code editor in the Android Studio helps the user to
be more productive while developing apps. Android Studio makes it easy to develop apps for any Android device. Android Studio’s new view and module support makes it easier to manage app projects and resources.

1.3 Fluid UI

Fluid UI is a browser based wire framing and prototyping tool developed by Fluid software and used to design mobile touch interfaces [6]. Fluid UI support platforms such as Android smart phones, Android tablets, iPhone and iPad. Fluid UI comes with over 2000 ready-made iOS, Android, Windows 8 and wireframe UI widgets for phone and tablet prototyping – all regularly updated with the latest advances in the mobile OS [7]. In this project Fluid UI is used to design the prototypes of user interface.

1.4 Location Based Services

The ability to sense location is one aspect where a smart phone separates itself from a conventional personal computer in terms of benefits and utility [2]. A smart phone can be easily carried by the user unlike a personal computer, a smart phone can be used anywhere for accessing the location based applications such as Google maps for looking up driving directions. The importance and usefulness of location sensing has already been well recognized and accepted with the fame of GPS based navigation systems. Most of the today’s smart phones have built-in location sensing capabilities. Since most people rely on smart phones when on-the-go and far from home or office, apps that influence location based services can add real value to the user [2].

The location of the user is determined by Android’s network location provider using GPS, cell tower signals and Wi-Fi signals. The Location Manager System services take care of the access to location-based services. The Location Manager can be accessed
by using the `getSystemService()` method. The current location of the user can be fetched in following ways.

i. GPS (Global Positioning System)

ii. Network Service Location

iii. Passive

Any of the above providers can be used to fetch the current location of the user (or) user’s device. In this project both GPS and Network Service location are used to fetch the current location of the user, the reason for using both is, it may not be possible to obtain the location using GPS provider indoors, and location using network location providers when the network connectivity is poor.

1.4.1 GPS

The GPS uses a star grouping of satellites and ground stations to compute position and time almost anywhere on earth. A GPS receiver calculates its position by using a satellite ranging technique. To find the current location of the user the smartphones must have an inbuilt GPS receiver. In Android the statement `LocationManager.GPS_PROVIDER` determines the location using satellites.

1.4.2 Network Service Location

Android Network location provider determines user location based on availability of cell tower and Wi-Fi access points. In Android the access to network provider is requested using `LocationManager.NETWORK_PROVIDER`. 
1.4.3 Passive

A special location provider for receiving locations without actually initiating a location fix. This provider can be used to passively receive location updates when other applications or services request them without actually requesting the locations yourself. This provider will return locations generated by other providers. Requires the permission `android.permission.ACCESS_FINE_LOCATION`, although if the GPS is not enabled this provider might only return coarse fixes. With the passive location provider as the independent source, it is difficult to access the current location of the user.

![Diagram](image)

**Figure 1.1: Access to Location Information [11].**

1.4.4 Finding the location in Android

1. The programmer should set permissions in “Android Manifest.xml” file.

2. The programmer should create a location manager instance as reference to the location service.
3. The programmer can request current location from location manager.

4. The programmer receives location update from location listener on change of location.

Once the location information is changed, we can use geocoding and reverse geocoding.

1.4.5 Geocoding and Reverse Geocoding

Geocoding allows the user to translate between street address and latitude/longitude map coordinates. The Geocoder class provides access to two geocoding functions. The process of converting address into latitude/longitude is called Forward Geocoding. The process of converting latitude and longitude into address is called Reverse Geocoding.

1.5 Location Reminders

Reminders are generally based on time. Frequently, however, time is not sufficient to configure the context in which the user needs to be reminded. A location reminder system allows users to set reminders taking into account their location. Using location to activate reminders is a valuable piece of content that can enhance the way people use reminders [2].

1.6 Existing Applications

1.6.1 Google Keep – notes and lists

Google Keep is a note taking app developed by Google. In Google Keep, the user can set time or location based reminders.

Advantages

i. Create and access notes from anywhere and anytime.
ii. Changes made by the user will be instantly saved to the cloud and synced with all the other devices of the user.

Disadvantages

i. No voice based alert.

ii. Limited features.

iii. Unorganized.

1.6.2 Geobells

Geobells is an Android app that reminds the user about his/her task based on location.

Advantages

i. Voice based alert.

ii. The user can add repeat option for his/her reminders.

Disadvantages

i. No Time Based Reminder.

ii. Confusing graphical user interface

iii. Paid app.

1.7 Solution – Locale-Reminder App

It was described in section 1.6 that there are some applications that help the user to set a reminder based on location and time. There has been no single location reminder application yet that has a unique solution for all the following problems listed below:

- An app that allows the user to set a reminder even if he/she does not know the name of the location at which they want to be reminded.
- An app that has repeat option for time based reminders.
- An app that has voice based alert.
- An app that allows the user to save his/her current location coordinates for creating reminders in future. An app that helps the user navigate to his/her saved location by using the *Get Directions* feature.
- An app that allows the user to share his/her current and saved locations via messaging or email service.
- An app that allows the user to create reminders with his/her voice.

The intent and main purpose of this project is to come up with an Android application which efficiently and effectively solves all the problems listed above. This application is developed on Android so as to reach the vast majority of the people, as Android covers over 78% of the smartphone market.
2. NARRATIVE

2.1 Problem Statement

In the current Android market there are some location based reminder applications, but most of them are not really helpful to users. Most of the existing location based reminder applications alert the user with a pop-up notification, which may mislead the concentration of the user while driving and can cause an accident. There are no apps that help the user to set a reminder based on location that is unknown to the user. There are no apps that will allow the user to save his/her current location coordinates for creating reminders in future and will help the user navigate to his/her saved location by using Get Directions feature.

2.2 Motivation

In general to set a location based reminder the user needs to know the name of the location, where he/she wants to be reminded. This does not work all the time, because if the user is at an unknown location and he/she wants to be reminded, it will be difficult for the user to set a reminder in this situation. Moreover, location based reminders are mainly generated when the user is approaching a location, in other context they are mostly generated when he/she is driving on the road. If a location reminder app alerts the user with a normal text based notification, this may distract the concentration of the user driving and can cause an accident. These difficulties motivated the development of an application that would eliminate such efforts by bringing these functionalities to a single application and thus can enhance the way people use reminders.
2.3 Project Objective

The main objective of the proposed system is to enhance the way in which people use reminders. This is achieved by helping the people to create both time and location based reminders. The application also allows the people to create reminder for unknown locations. In addition the application has voice based alert functionality, along with regular pop-up notifications. The application allows the user to save his/her current location for creating reminders in future, and helps the user navigate to their saved location by using Get Directions feature. The application allows the user to share his/her current and saved locations via messaging or email services. The application has a Safe Driving Mode functionality, which helps the user to completely focus on driving, by automatically sending a text message to the caller that the user is driving. In summary, this application helps people organize their tasks effectively by taking the burden of remembering things and reminding them at a specific time or location.

2.4 Project Scope

Technology has improved a lot over the last few decades. One of the best and biggest technological advancements is the invention of smart phone. A smart phone is a device which offers more advanced computing and connectivity than regular mobile phones [12]. In the last one or two years the smart phone users have rapidly increased and the count is still on. People using smart phones demand for better applications and updates for existing ones, which in turn creates a huge scope of Android mobile application development [14]. Android is a fully open-source platform which was created completely for smart phones and similar devices like tablets [13]. This project is developed for Android based smart phones. User can only avail the services of the
Locale-Reminder app when his/her smart phone’s GPS is switched on or when they are connected to the internet, because as the name suggests, Locale-reminders are notifications or alerts that are triggered when the user is at specific place. The app should keep an eye on the GPS information collected by user’s smart phone to trigger the alert when the user approaches a destination. According to data collected during a 7-day period ending on February 2, 2015 by Google there are 90.4% of Android users who use Android versions 4.0 to 5.0. Locale-Reminder app is compatible with different versions of Android, starting from SDK version of Android 4.0 (Ice Cream Sandwich) to the recent update Android 5.0 (Lollipop).

2.5 Functionalities of the Project

This project presents Locale-Reminder, a mobile app targeted for Google’s Android platform. The following are the functionalities provided by this app.

1. A reminder system that allows users to create location reminders and be alerted when they are approaching the location. Based on the distance selected, the user will be reminded when he gets into the distance range.
2. It also has a feature to select a range of dates in which he/she should be alerted when approaching a location.
3. Ability to create location reminders for unknown locations.
4. Ability to create time based reminders, and repeating reminders.
5. Ability to save locations for creating reminders in future.
6. Ability to share current and saved locations of the user by messaging or email services.
7. Ability to navigate to saved locations by using Get Directions.
8. Ability to create voice based reminders.

9. Beyond text, Locale-Reminder alerts the user with voice.

10. *Safe Driving Mode* feature helps the user to completely focus on driving, by automatically sending a text message to the caller that the user is driving.

11. It also has additional feature like ability to list most frequent tasks.
3. PROPOSED SYSTEM DESIGN

The schematic diagram of the proposed system Locale-Reminder application is shown in Figure 3.1. Most smart phones are built-in with both a GPS receiver and a Wi-Fi network interface card, which can get radio signals from GPS satellites and Wi-Fi Access Points (APs), individually. Taking into account the GPS readings and the data from the Wi-Fi APs, the application can perform geolocation to estimate the current location of the user. The database is designed to store location based tasks, time based tasks and unknown locations of the user, which are stored in different tables. If a location-based task exists in the database, then the application will compare the current sensed location with the location associated with the task. At the point when the user is physically near to the predefined location, the reminder then will be activated to remind the client of his/her task. If a time-based task exists in the database, then the application will compare the current date and time in the system with the date and time associated with the task. When both the date and times are matched, a reminder is triggered to remind the user of his/her task.
3.2 Use Case Diagram

Figure 3.2 shows the use case diagram for Locale-Reminder application. In the use case diagram of the Locale-Reminder, all the ovals represent the functionalities of the user. The Add Reminder functionality helps the user to create time or location based reminders. The Get My Location functionality locates the current location of the user with location address, latitude, longitude and altitude coordinates. It helps the user to create location based reminder for the current location and save his current location for creating reminders in future. The View Saved Locations functionality lists all the saved locations of the user, it helps the user to create location based reminders and share his saved location by messaging or email services, The user can navigate to his saved location by using the Get Directions property in View Saved Locations. The My To-Do’s
functionality lists all the reminders that are created by the user. The *Safe Driving Mode* helps the user to focus on driving by blocking all incoming calls, and informs everyone through automatic text replies that the user is driving. While driving the user is alerted using voice based reminders to create a safe environment. The *Settings* functionality helps the user to change alert settings like, choosing alert tone turning on vibration. The user can use the Alert Radius feature in Settings to set a distance at which he/ she should be alerted before a location.

![Diagram](image)

**Figure: 3.2 Locale-Reminder Use case Diagram**

### 3.3. Class Diagram

The class diagram consists of the interfaces, methods, variables and relationship between them. Figure 3.3 is the class diagram for Locale-Reminder app which describes the major functionalities of the user like creating *Location Based Reminder, Time Based*
Reminder and saving them in the database. Apart from these functionalities, users can get the current location by using Get My Location and can save the location coordinates in the database for creating reminders in future. The reminder database has three main tables, Location Reminder table, Time Reminder table and Location Value table for storing and retrieving the data.

![Class Diagram](image)

Figure: 3.3 Locale-Reminder Class Diagram
3.4. Sequence Diagram

A sequence diagram describes how the communication happens between the user, application, and local data base. Figure 3.4 describes the user sequence with the application and the application handling the user request. The application processes the user request and makes a request to the device’s local SQLite database to store and retrieve the data.

Figure: 3.4 Sequence Diagram for User interaction with application

3.5. User Interface

The main activities in the application are home screen, Location Based Reminder, Time Based Reminder, My To-Do’s, Safe Driving Mode, Get My Location, View Saved Locations and Settings. All these modules implementation is discussed in detail in chapter 4.
Figure 3.5 shows the home screen of the Locale-Reminder app. From the home screen the user can interact with all functionalities in the app.

![Image of Locale-Reminder home screen]

**Figure: 3.5 Home Screen of Locale-Reminder app**

### 3.5.1 Add Reminder

The user will be navigated to the Time Based Reminder activity on touching the ADD REMINDER icon as shown in Figure 3.6. In *Time Based Reminder* the user can create a reminder with respect to time. When the user clicks on Save button, the input will be validated and saved in the *Time Reminder* table of the device’s local database. Alarm Manager will trigger the alarm at user specified date and time. When the user swipes the Time Based Reminder screen fragment from right to left, app will navigate the user to Location Based Reminder screen as shown in the Figure 3.7.
Figure: 3.6 Time Based Reminder

Figure: 3.7 Location Based Reminder
In *Location Based Reminder* the user can create a reminder with respect to location. When the user clicks on Save button, the input will be validated and saved in the *Location Reminder* table of the device’s local database. Location Manager is used to trigger the alarm when is user is approaching the location.

### 3.5.2 My To-Do’s

When the user touches on the MY TO-DO’S icon in the home screen as shown in Figure 3.5, he/she will be navigated to the Time Based List, where a list of all time based reminders are shown as in Figure 3.8.

![Time Based List](image)

**Figure: 3.8 Time Based List**

In Time Based List the user can see “Upcoming” and “Completed” status tags for reminders, where the tag “Upcoming” means the task is not yet completed, and the user will be alerted on the specified date and time. The tag “Completed” means the task is
already completed and the user will not be alerted any more. When the user swipes the Time Based List screen fragment from right to left, the user will be navigated to the Location Based List screen as shown in the Figure 3.9.

![Location Based List](image)

**Figure: 3.9 Location Based List**

In Location Based List the user can see “Upcoming”, “Completed” and “Expired” status tags for reminders, where the tag “Upcoming” means the task is not yet completed, and the user will be alerted when the user is close to the specified location. The tag “Completed” means the task is already completed and the user will not be alerted any more. The tag “Expired” means the task has been never triggered and the user will not be alerted any more.
3.5.3 Get My Location

When the user touches on GET MY LOCATION icon in the home screen as shown in Figure 3.5, he/she will be navigated to the my location activity, where the user can see a list of location coordinates like latitude, longitude, altitude and address for current location as shown in Figure 3.10.

![My Location Activity](image)

**Figure: 3.10 Get My Location**

In Get My Location activity, the user can create a reminder for his current location by clicking on the Add Reminder button or save his current location for creating reminders in the future or share his current location with others via messaging or email services by clicking on the navigation image shown in Figure 3.10.
3.5.4 View Saved Locations

When the user touches on VIEW SAVED LOCATIONS icon in the home screen as shown in Figure 3.5, he/she will be navigated to the saved locations activity, where the user can see a list of previously saved locations as shown in Figure 3.11(a).

![Saved Locations](image)

(a)

![My Location](image)

(b)

**Figure: 3.11 View Saved Locations**

In View Saved Locations, when the user touches on any of his saved location as shown in Figure 3.11(a), he/ she will be navigated to the saved location activity as shown in Figure 3.11(b). The user can create a reminder for his saved location by clicking on Add Reminder button, share his saved location with others via messaging or email services by clicking on the navigation image as shown in Figure 3.11(b) and can navigate to his saved location by clicking on Get Directions button.
3.5.5 Safe Driving Mode

When the user touches on SAFE DRIVING MODE icon in the home screen as shown in Figure 3.5 he/she will be navigated to the Safe Driving Mode activity as shown in the Figure 3.12.

Figure: 3.12 Safe Driving Mode

Safe Driving Mode helps the user to completely focus on driving by answering all the incoming calls with automatic text replies. Safe Driving Mode by default has a custom message displaying ‘I am driving right now, I will call you later’. The application allows the users to change the default message to their own custom message.
3.5.6 Settings

When the user touches on SETTINGS icon in the home screen as shown in Figure 3.5, he/she will be navigated to the Settings activity, where the user can see a list of settings like NOTIFICATION TYPE, Alert Radius as shown in Figure 3.13.

![Figure: 3.13 Settings](image)

In Settings, the user can choose the alert tone, enable or disable Vibrate option, set Voice Reminder option and increase or decrease the Alert Radius. If the user enables the Voice Reminder option, the user will be alerted with voice based notification.
4. IMPLEMENTATION OF THE APPLICATION MODULES

4.1 User Interface Implementation of Locale-Reminder

All Functionality icons used in the home screen are Image Views. In Locale-Reminder app two libraries have been used from GitHub [17]. MaterialEditText is a library which contains functionality for text formatting and spell checking. DateTimePicker is a library which contains functionality or selecting date and time. The following statements should be added to the build.gradle file and settings.gradle file to make use of these libraries,

In build.gradle file add:

```java
compile project(':library:datetimepickerlibrary')
compile project(':library:EditTextlibrary')
```

In settings.gradle file add:

```java
include ':library:datetimepickerlibrary'
include ':library:EditTextlibrary'
```

4.2 Setting Permissions in AndroidManifest.xml file

Every application has an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about the application to the Android system. This information is essential for the system to run any of the application’s code. All permissions used in the app are set in the manifest file. Figure 4.1 shows the permissions used in the Locale-Reminder app.
<uses-permission android:name="android.permission.receive_boot_completed" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permissionINTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.CALL_PHONE" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

Figure: 4.1 Code snippet for setting up permissions in AndroidManifest.xml file

4.3 Connecting SQLite with Locale-Reminder

In Locale-Reminder, the SQLite database is used to store and retrieve the data. Using SQLite database in Android does not require a setup procedure or administration of the database. The user has to define the SQL statements for creating and updating the database. The database is automatically managed by the Android platform. The Application Program Interface (APIs) that are required for using a SQLite database on Android are available in the android.database.sqlite package.

In Locale-Reminder, three tables are created for storing and retrieving the user data. Time Reminder table stores the data related to time based tasks, Location Reminder table stores the data related to the location based tasks, Location Value table stores the location coordinates. The code snippet for creating all these tables is shown in Figure 4.2.
public void onCreate(SQLiteDatabase db)
{
    try
    {
        // Creating Table for Time based Data
        db.execSQL("CREATE TABLE TIME_REMINDER (_ID INTEGER PRIMARY KEY, TIME_TITLE VARCHAR," + "TIME_NOTE VARCHAR, TIME_DATE VARCHAR, TIME_TIMESTAMP VARCHAR, TIME_IMAGE VARCHAR, STATUS VARCHAR, ISREPEAT VARCHAR)";
        // Creating Table for Location Based data
        db.execSQL("CREATE TABLE LOCATION_REMINDER (_ID INTEGER PRIMARY KEY NOT NULL , LOCATION_TITLE VARCHAR," + "LOCATION_NOTE VARCHAR, START_DATE VARCHAR, DUE_DATE VARCHAR, DUE_TIME VARCHAR, START_TIME VARCHAR," + "LOCATION_IMAGE VARCHAR, LATITUDE DOUBLE, LONGITUDE DOUBLE, STATUS VARCHAR, LOCATION_NAME VARCHAR)";
        // Creating Table for Storing location values.
        db.execSQL("CREATE TABLE LOCATION_VALUE (_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL , TITLE VARCHAR," + "LATITUDE DOUBLE," + "LONGITUDE DOUBLE, ALTITUDE DOUBLE, LOCATION_NAME VARCHAR)";
        Log.e("Sqlite Status","Successfully Created");
    }

    Figure: 4.2 Code snippet for creating tables.

4.4 Location Based Reminder

This module is specific to creating reminder with respect to location. When the user wants to create a location based reminder, he/ she can create a reminder by typing the name of the location in the address bar and selecting an address from a list view which is generated based on the matches to the typed characters in the address bar. When the user wants to create a reminder for his saved location, he/ she can create a reminder by clicking on Get Saved Location button, which shows a list of saved locations and the user can select a location from the list at which he/ she wants to be reminded. When the user wants to create a reminder for his current location, he or she can create a reminder by pulling his/her current location coordinates using Get My Location feature in the application, and clicking on Add Reminder button.
In this application places API is used to display the list of matched locations when the user types the first few characters of the location at which he/she wants to be reminded. The data from places API can be accessed after registering at console.developers.google.com and generating a key for places API. This key is used to access the data from the places API. Figure 4.3 shows the places API key generated for Locale-Reminder.

![Key for server applications](image)

**Figure: 4.3 Snippet for places API key generated for the application.**

In order to access the places data from the places API a URL should be created. Figure 4.4 shows the code snippet for creating the URL.

![Code snippet for creating URL](image)

The data from the places API is returned in JSON format. This data should be parsed in order to be displayed it to the user in the form of a list view. Figures 4.5 and 4.6 shows the code snippets for parsing the JSON data and displaying it in the form of a list view.
//Google Browser API KEY
String key = "key=" + Constant.API_KEY;
// place type to be searched
String types = Constant.LOCATION_TYPES;
// Sensor enabled
String sensor = "sensor=false";
// Building the parameters to the web service
String parameters = input + "&" + types + "&" + sensor + "&" + key;
// Output format
String output = "json";
// Building the url to the web service
String url = "https://maps.googleapis.com/maps/api/place/autocomplete/" + output + "?" + parameters;
Log.e("At line 1068 USRSELLOC.java","url is "+url);
try
{
// Fetching the data from we service
data = downloadUrl(url);
}

Figure: 4.4 Code snippet for creating URL

List<HashMap<String, String>> places = null;
PlaceJsonParser placeJsonParser = new PlaceJsonParser();
try
{
  jObject = new JSONObject(jsonData[0]);
  // Getting the parsed data as a List construct
  places = placeJsonParser.parse(jObject);
}
catch (Exception e)
{
  Log.d("Exception", e.toString());
} return places;

Figure: 4.5 Code snippet for parsing the JSON data
String[] from = new String[]{"description"};
int[] to = new int[]{android.R.id.text1};
locationHashMap = result;
//Log.e("result value ", from.toString());
Log.e("result value ", "+from);
// Creating a SimpleAdapter for the AutoCompleteTextView
SimpleAdapter adapter = new SimpleAdapter(context, result,
android.R.layout.simple_list_item_1, from, to);
// Setting the adapter
locationSearch.setAdapter(new ArrayAdapter<String>(context,
android.R.layout.simple_list_item_1,resultlist));

Figure: 4.6 Code snippet for showing the parsed data in a list view.

After choosing location from the list view, forward geocoding is performed to get
the corresponding latitude and longitude coordinates for the selected location. Latitude
and longitude coordinates for the selected location are retrieved by passing the location
address to the Location Manager class. If the user selected location has the latitude and
longitude values, the Location Manager will return the location object for retrieving the
corresponding latitude and longitude values. If the user selected location does not have
the corresponding latitude and longitude values in the Location Manager class, it will
return a null object. When a Location Manager returns a null object for any location, the
user is alerted by an error message that displays a reminder cannot be created for that
location.

When creating reminders from saved locations and by using Get My Location, the
corresponding latitude and longitude values for that location are used rather than
performing forward geocoding, this feature makes Locale-Reminder to create reminders
for locations that do not have a valid name.
After retrieving the corresponding latitude and longitude values for the location, these values are passed to the `addProximityAlert()` method in the Location Manager class. Figure 4.7 shows code for setting the location based reminder.

```java
// if no time range is given
mLocationManager = (LocationManager)
context.getSystemService(context.LOCATION_SERVICE);
PendingIntent proximityIntent = PendingIntent.getBroadcast(context, uniquekey, intent,
Intent.FLAG_ACTIVITY_NEW_TASK);

mLocationManager.addProximityAlert(
    latitude, // the latitude of the central point of the alert region
    longitude, // the longitude of the central point of the alert region
    POINT_RADIUS, // the radius of the central point of the alert region, in meters
    PROX_ALERT_EXPIRATION, // time for this proximity alert, in milliseconds, or -1 to
    indicate no expiration
    ProximityIntent // will be used to generate an Intent to fire when entry to or exit from
    the alert region is detected
);
```

**Figure: 4.7 Code snippet for creating location based reminder.**

### 4.5 Time based reminder

This module is specific to creating reminders with respect to time. To create a time based reminder, the user should enter the date and time on which he/she wants to create a reminder, a title and description for the reminder. Locale-Reminder allows the user to add repeat option for the reminder. If the user selects the repeat option the reminder will be repeated on the same day and time of every week. Locale-Reminder also allows the user to add an image for his reminder. The user can take a new picture by using his mobile camera or can upload an existing picture from his device. Once the application receives the due date and time of the reminder, the application creates an alarm by using the Alarm Manager class as shown in the Figure 4.8. Alarm Manager
class is a service which runs in the background and compares the system time with the alarm time, and when the time matches it will alert the user.

```java
//Setting the Alarm.
if (calendar.getTimeInMillis() > System.currentTimeMillis())
{
    if (uniqueRequestCode == 0)
    {
        uniqueRequestCode = getRequestCode();
    }
    saveTimeReminderDataInDatabase(uniqueRequestCode);
    Toast.makeText(timeView.getContext(), "Your Reminder has been Scheduled ", Toast.LENGTH_SHORT).show();
    Log.e("TimeBasedReminder.java, at line 171", " reminder set on " + calendar.getTime() + " with isRepeat value as " + isRepeat.isChecked());
    boolean alarmValue = alarm.setAlarm(timeView.getContext(), calendar, title, description, uniqueRequestCode, isRepeat.isChecked());
    // registering the alarm to alarm Manager Class
    PendingIntent pi = PendingIntent.getBroadcast(context, uniqueRequestCode, intent, PendingIntent.FLAG_UPDATE_CURRENT);
    AlarmManager am = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
    if (!isRepeat)
    {
        am.set(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), pi);
        Log.e("AlarmMgrHelper.java, at line 98 ", " reminder time value without repeat is " + calendar.getTime());
    }
    else
    {
        am.setInexactRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), 7 * 24 * 60 * 60 * 1000, pi);
        Log.e("AlarmMgrHelper.java, at line 103 ", " reminder time value with repeat is " + calendar.getTime());
    }
}
```

**Figure: 4.8 Code snippet for creating time based reminder.**

### 4.6 Voice Reminder

Voice reminder helps the user to completely focus on driving by alerting the user with a voice based reminder. This feature is achieved by using the `getSpeech()` method in the TextToSpeech class as shown in the Figure 4.9.
public void getSpeech()
{
    tts = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener()
    {
        @Override
        public void onInit(int status)
        {
            tts.setLanguage(Locale.ENGLISH);
            tts.setSpeechRate(.5f);
            tts.speak("Approaching " + message + " Don't Forget to " + locationTitle,
                    TextToSpeech.QUEUE_FLUSH, null);
        }
    });
}

Figure: 4.9 Code snippet for creating voice based reminder

4.7 Get My Location

Get My Location is used to retrieve the current location of the user. The user can create reminder for his current location, by using the Get My Location feature. Get My Location feature helps the user to create reminders even if he/ she doesn’t know the name of the location or if the location does not have a proper name. Here the reminder is created by using the latitude and longitude values. The user can save his current location by clicking on the Save button. The purpose of saved location feature is to use this location to create reminders in future or navigate to the saved location by using Get Directions feature. The user can share his current and saved location by using the navigation image as shown in Figure 3.11(b).

In order to receive the location coordinates the user should check for GPS provider or Network provider to get the location updates. The code snippet for requesting location updates from GPS provider and Network provider is shown in Figure 4.10.
mLocationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
gps_enabled = mLocationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
network_enabled = mLocationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
if (gps_enabled) {
mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, mLocationListener);
Log.e("Provider", "GPS PROVIDER");
cLocation = mLocationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
} else if (network_enabled) {
mLocationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, mLocationListener);
Log.e("Provider", "NETWORK PROVIDER");
cLocation = mLocationManager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
}

**Figure: 4.10 Code snippet for requesting location updates.**

Once the application receives the location object, it can extract the current latitude, longitude, altitude and address of the location using the methods shown in the Figure 4.11.

if (address.size() > 0 && address != null) {
Log.e("Called onPostExecute ","at line 322");
// latitude value
myLatitude.setText(String.valueOf(mLocation.getLatitude()));
// longitude value
myLongitude.setText(String.valueOf(mLocation.getLongitude()));
// altitude value
myAltitude.setText(String.valueOf(mLocation.getAltitude()) + "m");
// address value
myAddress.setText(address.get(0).getAddressLine(0)+","+address.get(0).getAddressLine(1)+","+address.get(0).getAddressLine(2));
}

**Figure: 4.11 Code snippet for getting location coordinates.**
4.8 Safe Driving Mode

*Safe Driving Mode* helps the user to completely focus on driving by answering all his incoming calls with an automatic text reply. The user should enable *Safe Driving Mode* by swiping the switch from off state to on as shown in Figure 5.30(a). In *Safe Driving Mode*, the default message ‘I am driving right now, I will call you later’ is displayed. The user can change the default message by entering his own custom message in the text box and clicking on the Save button. The code snippet for answering the incoming call and sending the SMS is shown in Figure 4.12.

```java
Public Boolean endCallIfBlocked(String callingNumber)
{
    boolean value;
    try
    {
        //End Call
        TelephonyManager tm = (TelephonyManager)
        context.getSystemService(context.TELEPHONY_SERVICE);
        Class c = Class.forName(tm.getClass().getName());
        Method m = c.getDeclaredMethod("getITelephony");
        m.setAccessible(true);
        com.android.internal.telephony.ITelephony telephonyService = (ITelephony) m.invoke(tm);
        telephonyService.endCall();
        Log.e("Call is blocking ", "Yes working, count is : " + count);
        value = true;
    }
    // Send SMS
    SharedPreferences.Editor edit = sharedpreferences.edit();
    edit.putString("missedcall", number);
    edit.commit();
    String message1 = sharedpreferences.getString("message", "I am driving right now, " +"I will call you Later");
    SmsManager sms = SmsManager.getDefault();
    sms.sendTextMessage(number, null, message1, psent, pdelivered);
}
```

**Figure: 4.12 Code snippet for answering incoming call and sending SMS.**
5. TESTING AND EVALUATION

This chapter deals with the functional evaluation of the application. The application is tested by installing it on a Moto G mobile phone with android version 4.4.4. To install the application, “install from other locations” setting has to be enabled on the device under the developer options. The application supports minimum Android version 4.0.4 to recent update 5.0.1. Each module of the application is tested with all possible test cases.

5.1 Launching the application

Figure 5.1 shows the launching screen of the application when the user clicks Locale-Reminder icon on the home screen. Figure 5.1(a) shows the launcher activity and Figure 5.1(b) shows the application’s home screen.

![Figure: 5.1 Launching the application](image)
5.2 Time Based Reminder

In *Time Based Reminder* the input is validated. If the user clicks on the Save button without entering values in the fields or entering an end date before start date, it shows an error message on the fields as shown in the Figure 5.2. Figure 5.2(a) shows validating the input when all the fields are empty and Figure 5.2(b) shows validating the input when the user enters invalid date and time.

![Figure: 5.2 Input validation for time based reminder](image-url)
The application also supports giving input through voice using speech recognition. Figure 5.3 shows converting user speech to text. By clicking on the mic icon as shown in Figure 5.3(a), a popup dialog will be displayed asking the user to speak and once the user speaks, the application will display a list of matching text as shown in Figure 5.3(b). Once the user selects a value in the list, the value selected will be placed as data in the field as shown in the Figure 5.4(a).

![Figure: 5.3 Speech to text feature](image)
Figure 5.4(a) shows the data in the fields given through speech input. Figure 5.4(b) shows the status of the reminder in *My To-Do’s* once it gets created. In the Figure 5.4(b) the status “Upcoming” means the reminder is scheduled to occur in the future at the specified date and time.

![Figure: 5.4 Creating a time based reminder](image-url)
Figure 5.5(a) shows the notification generated for a *Time Based Reminder* and Figure 5.5(b) shows the status of the reminder in *My To-Do’s* as “Completed”. The status “Completed” means the reminder has been generated (or) the event occurred.

![Generating notification for time based reminder](image)
Figure 5.6(a) shows postponing the reminder to one hour and Figure 5.6(b) shows the updated time value of the reminder after postponing it by one hour. The user can also cancel the notification by clicking on the Cancel button as shown in Figure 5.6(a).

(a)                                                                    (b)

Figure: 5.6 Postponing the notification by one hour
Figure 5.7(a) shows creating a reminder for 4-27-2015 at 9:32 pm with repeat option and Figure 5.7(b) shows the updated date as 5-4-2015, after the reminder has been generated once on 4-27-2015.

![Figure: 5.7 Creating a reminder with repeat option](image)

5.3 Location Based Reminder

In *Location Based Reminder* the input is validated. If the user clicks on the Save button without entering values in the fields or entering an invalid address or entering an end date before start date, it shows an error message on the fields as shown in the Figure 5.8. Figure 5.8 shows validating the input without entering anything in any of the fields and Figure 5.9(a) shows validating the input for invalid address. Figure 5.9(b) shows validating the input for invalid date and time.
Figure: 5.8 Input validation for location based reminder

(a) 

(b) 

Figure: 5.9 Input validation for date, time and address
Figure 5.10(a) shows the default Alert Radius value before creating the reminder and Figure 5.10(b) shows a list of location suggestions to the user based on his input.

Figure: 5.10 Creating a location based reminder with Alert Radius as 2 miles
Figure 5.11(a) shows creating a location based reminder with Alert Radius (distance to be reminded) as 2 miles and Figure 5.11(b) shows the status of the reminder in My To-Do’s after it has been created. In Figure 5.11(b), the status “Upcoming” means the reminder is scheduled to occur.

Figure: 5.11 Creating a location based reminder
Figure 5.12(a) shows the Alert Radius increased to 6 miles and Figure 5.12(b) shows creating a location based reminder after increasing the Alert Radius.

Figure: 5.12 Creating a location based reminder with Alert Radius as 6 miles
Figure 5.13(a) shows the notification generated when the user is near 6 miles to the destination location and Figure 5.13(b) shows the status of the reminder in *My To-Do’s* once the notification has been generated. In Figure 5.13(b) the status “Completed” means the reminder has been generated and will no longer generate notifications for that task.

![Figure: 5.13 Generating the notification for location based reminder](image-url)
Figure 5.14(a) shows touching on Get Directions image view for navigating to the location that has been used for creating the reminder and Figure 5.14(b) shows directions for the destination location in google maps.

(a)                                                                    (b)

Figure: 5.14 Getting directions for the reminder location
Figure 5.15(a) shows creating a location based reminder within a time range and Figure 5.15(b) shows the notification generated for that reminder.

Figure: 5.15 Creating a location based reminder with time range.
Figure 5.16(a) shows creating a location based reminder with a time range and Figure 5.16(b) shows status of the reminder in *My To-Do’s* as “Expired”. Figure 5.16(b) the status “Expired” means, the user did not visit that location within the time range, so the application automatically stops the notification service for that reminder after the time range and changes the status of the reminder to “Expired” state and will no longer alert the user for that reminder.

![Figure 5.16(a)](image1)

![Figure 5.16(b)](image2)

**Figure: 5.16** Automatically changing the reminder status to expired after the due date
Figure 5.17(a) shows the current location of the user and Figure 5.17(b) shows creating a reminder for the current location of the user.

![Image](a) ![Image](b)

**Figure: 5.17 Creating reminder from the current location of the user using Get My Location feature**

Figure 5.18 shows the notification generated for the reminder created from the current location of the user.

Figure 5.19(a) shows that the user has clicked on get saved locations to create reminder from one of his saved locations and Figure 5.19(b) shows the list of saved locations of the user.
Figure: 5.18 Notification generated for the reminder

Figure: 5.19 Creating reminders from saved locations
Figure 5.20(a) shows that the user has clicked on one of the saved locations from the list to create a reminder and Figure 5.20(b) shows the notification generated for the reminder created from the saved locations of the user.

(a)                                                                    (b)

Figure: 5.20 Automatically changing the reminder status to expired after the due date
5.4 Deleting Reminders

Figure 5.21(a) shows deleting the reminder and Figure 5.21(b) shows the response as a toast notification generated after deleting the reminder.

(a)                                                                    (b)

Figure: 5.21 Deleting a reminder
Figure 5.22(a) shows selecting multiple reminders for deletion and Figure 5.22(b) shows the response as a toast notification generated after deleting multiple reminders.

![Figure: 5.22 Deleting a list of reminders](image)
5.5 Get My Location

The *Get My Location* feature helps the user to know his current location, create a reminder for his current location, save his current location for creating reminders in future, sharing his current location via messaging or email services.

By clicking on Add Reminder button as shown in Figure 5.23(a) the user is navigated to the Location Based Reminder activity as shown in Figure 5.23(b). The user can click on the Save button by entering all details and the location based reminder is created.

![Figure 5.23 Creating reminder for current location](image)

(a)  
(b)
By clicking on Save button as shown in Figure 5.24(a) a pop up dialog box is shown to the user to enter the title of the location as shown in Figure 5.24(b).

![Image of My Location app](image1)

(a) [Image of My Location app](image2)

(b)

Figure: 5.24 Saving current location

By clicking on navigation image as shown in Figure 5.25(a) a pop up dialog box is shown to the user with a list of messaging and email services that are running in the device. Figure 5.25(b) shows sharing the current location of the user via messaging service.
5.6 View Saved Locations

The View Saved Locations activity contains a list of locations saved by the user using the *Get My Location* feature. As discussed in section 5.5, the user can create a reminder for his saved location, *Get Directions* to a saved location, share a saved location by using messaging or email services by touching on navigation image as shown in Figure 5.25(a).

Figure 5.26(a) shows the list of saved locations by the user and Figure 5.26(b) shows creating a reminder for one of the saved location. When the user clicks on Add Reminder button he/ she will be navigated to Location Based Reminder activity along with the saved location address details. This address is used for creating a location based reminder as shown in Figure 5.27.
Figure: 5.26 List of saved locations

Figure: 5.27 Creating reminder for saved location
By clicking on Get Directions button as shown in Figure 5.28(a) the user is navigated to the google maps activity as shown in Figure 5.28(b).

Figure: 5.28 Getting directions for saved location
By clicking on navigation image as shown in Figure 5.29(a) a pop up dialog box is shown to the user with a list of messaging and email services that are available in the device. Figure 5.29(b) shows sharing the current location of the user via Email service.

![Image of My Location](image.png)

**Home**

<table>
<thead>
<tr>
<th>My Latitude</th>
<th>27.71802251</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Longitude</td>
<td>-97.35092277</td>
</tr>
<tr>
<td>My Altitude</td>
<td>-11.0m</td>
</tr>
<tr>
<td>My Address</td>
<td>5445 S Alameda St, Corpus Christi, TX 78412, USA</td>
</tr>
</tbody>
</table>

**Compose**

gopairao000@gmail.com

**To**

gopairao@gmail.com

Our Meeting Place

5445 S Alameda St, Corpus Christi, TX 78412, USA

https://maps.google.com/maps?q=27.71802251,97.35092277

![Image of Email Sharing](image2.png)

**Figure: 5.29 Sharing the saved location**
5.7 Safe Driving Mode

When the user activates *Safe Driving Mode* as shown in Figure 5.30(a) a toast notification is displayed to the user, all their incoming calls are automatically answered with a default text message as shown in Figure 5.30(b).

![Figure 5.30 Safe Driving Mode with default message](image-url)
In *Safe Driving Mode*, the user can choose their own custom message by typing the message in the message text box as shown in Figure 5.31(a), which will be displayed to the person calling as in Figure 5.31(b).

![Figure: 5.31 Safe Driving Mode with custom message](image)
5.8 Settings

The *Settings* feature helps the user to choose the notification sound, enable or disable vibration, Voice Reminder and changing the Alert Radius as shown in Figure 5.32.

![Figure: 5.32 Settings activity](image-url)
6. CONCLUSION & FUTURE WORK

This project presents a new application for android based smart phones and tablets that allows the users to create reminders based on location, time etc. Locale-Reminder allows users to create location reminders and be alerted when they enter or exit the locality of a given location. Locale-Reminder has unique features such as creating voice based reminders, creating location reminders for unknown locations, repeating time based reminders, selecting a range of dates in which the user should be alerted when approaching a location, listing most frequent tasks, alerting the user with voice, saving the current location for creating reminders in future, navigating to saved locations, and finally sharing the current and saved locations of the user by messaging and email services. In addition, Locale-Reminder has Safe Driving Mode feature which helps the user to completely focus on driving, by automatically sending a text message to the caller that the user is driving.

Future Work

The app can be improved in the future by adding the following functionalities:

- Improving the battery life of the device by developing a new algorithm that filters the reminders to compare the location coordinates for generating the reminders based on the user direction.
- Changing phone Settings automatically based on location of the user.
- Automatically setting the device to Safe Driving Mode based on user’s rate of velocity.
- By improving the call blocking options in Safe Driving Mode.
• Ability to navigate to destination location by taking user voice commands.

• Ability to track user location automatically for a specific time interval, for example, if a parent and child are using the same app, the ability to track the location of the child helps the parent to know the location of the child every 5 minutes.

• User data can be stored in cloud.

• Ability to integrate with calendar events.

• Ability to share reminders with friends using the same application.

• Ability to display a list of suggested locations to the user based on users voice.
7. BIBLIOGRAPHY & REFERENCES


