iDoRemind: Location Based Reminder Application For Android

GRADUATE PROJECT

Submitted to the Faculty of
the Department of Computing Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

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

By

Srihari Reddy Pamulapati
Fall 2016

Committee Members

Dr. Long-Zhuang Li
Committee Chairperson

Dr. David R Thomas
Committee Member
ABSTRACT

In recent times, reminders play a vital role in human life. Reminder assists people in remembering things. Initially it used to be in the form of paper, but steadily it evolved to digital form. Location based reminders are the most widely used application at present. As name suggests location reminder helps people to remind about the location, still it requires more effort from users to utilize it. It does not have more features to make the life easier. In this application, the shortcomings of an existing location-based reminder are addressed, implementation and working of the features are explained in this document. It has been shown how the geofencing and data analysis along with trend analysis will make a reminder efficient.
# TABLE OF CONTENTS

ABSTRACT .......................................................................................................................... ii

TABLE OF CONTENTS ......................................................................................................... iii

LIST OF FIGURES ................................................................................................................... v

LIST OF TABLES ..................................................................................................................... vii

1. BACKGROUND AND RATIONALE .................................................................................. 1
   1.1 Android ..................................................................................................................... 1
   1.2 Android Studio ......................................................................................................... 2
   1.3 Geo-Fencing .............................................................................................................. 2
   1.4 Existing Application ................................................................................................. 2
   1.5 Common Problems in Other Existing Applications ................................................. 5
   1.6 Solution – iDoRemind App ....................................................................................... 6

2. NARRATIVE ....................................................................................................................... 7
   2.1 Problem Statement .................................................................................................... 7
   2.2 Motivation ................................................................................................................ 7
   2.3 Product Scope .......................................................................................................... 8
   2.4 Functionalities ......................................................................................................... 8

3. DEVELOPED SYSTEM DESIGN ..................................................................................... 10
   3.1 Use Case Diagram .................................................................................................... 11
   3.2 Class Diagram ......................................................................................................... 13
   3.3 User Interface ......................................................................................................... 14

4. IMPLEMENTATION OF APPLICATION MODULES ...................................................... 24
   4.1 Login ...................................................................................................................... 24
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Setting Permissions in AndroidManifest.xml file</td>
<td>25</td>
</tr>
<tr>
<td>4.3</td>
<td>Creating the Reminder</td>
<td>25</td>
</tr>
<tr>
<td>4.4</td>
<td>Connecting to Database</td>
<td>26</td>
</tr>
<tr>
<td>4.5</td>
<td>Adding Member to Group</td>
<td>28</td>
</tr>
<tr>
<td>4.6</td>
<td>Suggestions based on Preferences</td>
<td>30</td>
</tr>
<tr>
<td>4.7</td>
<td>Suggestions based on Reminders</td>
<td>30</td>
</tr>
<tr>
<td>4.8</td>
<td>Settings</td>
<td>31</td>
</tr>
<tr>
<td>5.1</td>
<td>Testing and Evaluation</td>
<td>33</td>
</tr>
<tr>
<td>5.2</td>
<td>Launching the Application</td>
<td>33</td>
</tr>
<tr>
<td>5.3</td>
<td>Creating the Reminder</td>
<td>34</td>
</tr>
<tr>
<td>5.4</td>
<td>Unique Code</td>
<td>36</td>
</tr>
<tr>
<td>5.5</td>
<td>Add Unique Code</td>
<td>37</td>
</tr>
<tr>
<td>5.6</td>
<td>Suggestions based on Preferences and Preferences</td>
<td>37</td>
</tr>
<tr>
<td>5.7</td>
<td>Near by places based on Reminders</td>
<td>38</td>
</tr>
<tr>
<td>5.8</td>
<td>Settings</td>
<td>39</td>
</tr>
<tr>
<td>5.9</td>
<td>Logout</td>
<td>40</td>
</tr>
<tr>
<td>5.10</td>
<td>Reminder</td>
<td>40</td>
</tr>
<tr>
<td>5.11</td>
<td>Reminder Status</td>
<td>42</td>
</tr>
<tr>
<td>5.12</td>
<td>Notification</td>
<td>43</td>
</tr>
<tr>
<td>6.1</td>
<td>Evaluation</td>
<td>45</td>
</tr>
</tbody>
</table>

6. CONCLUSION AND FUTURE WORK ...........................................48

BIBLIOGRAPHY and REFERENCES .............................................50

APPENDIX A ...........................................................................52
# LIST OF FIGURES

| Figure 3.1 | Architecture of iDoRemind Application | .................................................. | 10 |
| Figure 3.2 | Use Case Diagram of iDoRemind Application | .................................................. | 12 |
| Figure 3.3 | Class Diagram of iDoRemind Application | .................................................. | 13 |
| Figure 3.4 | Menu of iDoRemind Application | .................................................. | 14 |
| Figure 3.5 | Login Screen | .................................................. | 15 |
| Figure 3.6 | Create Reminder and Select Location, Date, Time, Repeat Screen | .................................................. | 16 |
| Figure 3.7 | Reminder Buttons: Delete, Assign Users, Share | .................................................. | 17 |
| Figure 3.8 | Enter Preferences and Your Preferences Screen | .................................................. | 18 |
| Figure 3.9 | Unique Token and Add Token Screen | .................................................. | 19 |
| Figure 3.10 | Group Members | .................................................. | 20 |
| Figure 3.11 | Suggestions Based on Preferences | .................................................. | 21 |
| Figure 3.12 | Near By Places Based on Reminders | .................................................. | 22 |
| Figure 3.13 | Settings | .................................................. | 23 |
| Figure 3.14 | Notification of the Reminder | .................................................. | 24 |
| Figure 4.1 | Adding Libraries to build.gradle File | .................................................. | 25 |
| Figure 4.2 | Gmail Authentication | .................................................. | 25 |
| Figure 4.3 | Setting Permissions in AndroidManifest.xml File | .................................................. | 26 |
| Figure 4.4 | Code to Pick the Location to Create Reminder | .................................................. | 27 |
| Figure 4.5 | Database Connection in Webserver | .................................................. | 28 |
| Figure 4.6 | Request and Response of Rest Service | .................................................. | 28 |
| Figure 4.7 | Reminder in List View | .................................................. | 29 |
| Figure 4.8 | Add New Member to the Group | .................................................. | 30 |
Figure 4.9 Sharing the Message to the Others in the Group .................................................. 30
Figure 4.10 Inserting Preferences into Array List ................................................................. 31
Figure 4.11 Populating Near By places in List View ............................................................. 32
Figure 4.12 Settings .................................................................................................................. 33
Figure 5.1 Login Screen and Dash Board of iDoRemind Application .......................................... 34
Figure 5.2 Authorization of Facebook and Gmail ........................................................................ 35
Figure 5.3 Validation of Creating Reminder (Title, Location, Date) ........................................... 36
Figure 5.4 Reminder with Restrictions and Without Restrictions .............................................. 37
Figure 5.5 Sharing the Token Using WhatsApp ....................................................................... 37
Figure 5.6 Add Token Validation ............................................................................................. 38
Figure 5.7 Suggestions Based on Preferences ......................................................................... 39
Figure 5.8 Near By Places Based on Reminders ..................................................................... 39
Figure 5.9 Notifications Based on Suggestions Status ............................................................ 40
Figure 5.10 Notifications When Only One Notification is On .................................................. 41
Figure 5.11 Assign Users .......................................................................................................... 42
Figure 5.12 Sharing the Reminder via Text Message ............................................................... 43
Figure 5.13 Notification Status ................................................................................................ 43
Figure 5.14 Getting the Directions to Destination by Clicking on Navigate ............................... 44
Figure 5.15 Sharing the Reminder When Ignored ................................................................... 45
Figure 5.16 Graphical Representation of Feedback .................................................................. 47
LIST OF TABLES

Table 2.1 Comparison of iDoRemind with Other Application..................................................9

Table 5.1 List of Questions in Feedback and Their Average Rating........................................46
1. BACKGROUND AND RATIONALE

A location based reminder is more useful than that of a traditional time based reminders [1]. The current location of users can be detected with the embedded sensors in the mobile devices [2]. But these location based reminders may not be useful in many scenarios. It will not be effective if it reminds the user when he reaches a particular location after the working hours. This disadvantage can be eliminated by combining time parameters with location [2].

The shortcomings of these location based reminders can be improved by implementing an advanced geofencing concept, which provides more specific information than the normal one. This can help in generating more information about the user and his preferences [3]. This location based reminder along with the information about the places will help to provide more effective reminders [4]. By continuously evaluating the user interests will give a picture about interests and reminder patterns. This can be done by analyzing the data of the user. It requires an efficient database which can store all the data without affecting the performance of devices [5]. Trend analysis concept can be included in this reminder which helps to understand the interest of all other people and how it can be adapted in improving the intelligence of a location based reminder. Sharing the reminder with your family members or roommates can be very useful. The approach in this project is basically a mixture of all these features.

1.1 Android

Android is an operating system based on Linux kernel developed by Google for smartphones and tablets. It is also used for watches, cars, game consoles, cameras [6]. It is a free and open source which encourages the large community of developers to use the
open source code as a base for many community-driven projects which provide updates to older devices and add new features for advanced users. According to Statista (an online data analysis website), the share of the android operating system is growing every year and is 86.2% at present, nearly 1.5 million new devices have been activating around the world [7]. The first android based phone was released in 2008 and so many updated versions have been released from then and each version named after a dessert in alphabetical order starting from cupcake and the latest version is android 6.0 named as “Marshmallow”.

1.2 Android Studio

Android studio is the official integrated development platform for Android platform. It was designed based on IntelliJ IDEA. The first stable build of android studio was released in December 2014 and from then it replaced Eclipse Android Development Tools (ADT) as Google’s primary IDE for native android application development. Features like Graphical Layout editor, Multiple APK generation, Gradle-based build support, Code Refactoring makes it easy to work on projects and manage resources [8]. We can also test the applications developed on a virtual device provided by Android Studio.

1.3 Geo-Fencing

Geo-fence is nothing but a virtual barrier. Geo-fencing is creating virtual boundaries around a point using Global Positioning System(GPS) [9]. When a device enters these boundaries it will trigger the reminder.

1.4 Existing Applications

There are so many reminder applications available for android but very few of
them can work based on the location. Even the location based applications have so many limitations. Some of the location based reminder apps and their shortcomings are mentioned below.

1.4.1 Location Reminders

*Location Reminders* [10] is an android application that reminds user about reminder based on location.

**Advantages**

i. Can create reminder whether to alert at the time of entry or exit or both.

ii. Time based restriction of morning/afternoon.

**Disadvantages**

i. Cannot enter the date and time.

ii. Not possible to share the reminder.

iii. Cannot enter the radius value.

iv. Once added the place cannot be deleted from the list.

1.4.2 NTRT- Location reminders

*NTRT (Next Time Remember To)* [11]- Location reminders is another android application which works based on location.

**Advantages**

i. Live status of distance to location.

ii. Easy to search and select location.

**Disadvantages**

i. No time restriction

ii. No option to enter reminder at the current location.
iii. Cannot share a reminder.

iv. Cannot select location on the map, should enter the location name.

1.4.3 **Google Keep**

*Google Keep* [12] is a note taking and reminder application by google. The reminders can be based on either time or location but not both.

**Advantages**

i. Reminders will be synced with all the devices registered under the users google account.

ii. Can share the reminder with others.

**Disadvantages**

i. No time restriction for location based reminders.

ii. Unorganized user interface makes it difficult to use.

iii. No option to customize radius.

1.4.4 **Some papers based on location based reminder applications**

There are some papers which discuss about the importance of location based reminders and further improvement needed in this area. This section contains details about 2 papers about location based reminders.

*Place-Its: A study of Location-Based Reminders on Mobile Phones* [13] is a paper which gives the overview about features of *Place-Its* application. The advantages of this application is user can create the reminder at the current location (can pick the location on the map). But on the flip side no option to enter date or time and also no option to share the reminder. The user can the reminder to trigger when he enters the area or when he exits the area. The feedback in the paper indicates that users found the application
useful than regular to-dos and calendar events.

*Remind me- An Enhanced Mobile Location-Based Reminder application* [14] is another such paper. The advantages of remind me application is the option to add time restriction to the location. It also has option of deleting the reminders. But no option provided to enter date and time. The feedback from users shows that they found the feature of adding time restriction more useful.

### 1.5 Common Problems in Other Existing Applications

There are some more applications in which reminders can be set based on date and time as well as location. Most commonly faced challenges of such reminders are explained below.

#### 1.5.1 Person not able to travel

It is very common for everyone to not travel to a particular location where already planned to visit within a period of time. For example, in a scenario which user has planned to visit a pharmacy to buy medicines, but he may not be able to travel to or near to the pharmacy. In this case, the reminder would be ineffective. Even if the reminder alerts the user when the time elapsed, the user will either ignore the alert or modify the alert for future. The solution for this problem in application is “Near by places”.

#### 1.5.2 Alert remains unattended

There are situations where users miss to notice the alert in their phone. For example, in a situation where the user sets a reminder to alert him about a bill payment and if he fails to attend the reminder he may miss the deadline. The user can use the option “ignore button” in the *iDoRemind* application.

#### 1.5.3 No alert preferences
There are hardly any preferences for an existing reminder to know the right time to alert the user. For example, in a scenario where the user needs to visit the post office, the location-based reminders would not fulfill the needs because the reminder may alert when the user is close to the post office at late night. It will not satisfy the need of the reminder [2]. This problem overcame in the application by providing the feature of entering “preferences”.

1.5.4 No autonomous capabilities

An existing reminder does not have an intelligence to come up with a suggestion to set a reminder. Even if the user sets redundant reminders, an existing application will not suggest the user to set the alert. All the suggestions, near by places, repeat in the application can be the solution for this problem.

1.6 Solution – iDoRemind App

It was mentioned in section (1.5) that there are some reminder applications that will work based on location and time but none of them address all the problems listed. The main purpose of this project is to develop an android project which will address all the previously mentioned problems.
2. NARRATIVE

2.1 Problem Statement

In the present world, people are busy with number of tasks that need to be accomplished. These tasks vary from meeting at workplace to buying groceries. Since it is hard to remember all the works, use of reminders came into place. Various types of reminder tools like post-its, daily planners, personal task management software, etc. are in use [15]. Smart phones are widely used as tools to assist people to remember and plan their tasks. Alerts based on time are traditional methods. It will help to alert users on particular date and time. This type of reminder may not be useful if the users expect to be alerted when they approach to a particular location. For example, in a scenario where the user needs to visit a supermarket, traditional reminders would not be very useful, because the user may not be close to the supermarket at that point of time. Due to its limitation, it has been improvised with the help of location [1]. If the user does not travel to that particular location, location reminders without time restrictions will not be useful. Sometimes user may not be able to attend the reminder, in such cases there is no application which will share user reminder with others.

2.2 Motivation

A location-based reminder would alert the user whenever he approaches to particular location. Current location of users can be detected with the help of embedded sensors in the mobile phones [1]. It will make use of the geofencing technology to alert the user when he reaches a particular location [3]. In some scenarios it may not fulfill the need of users. It may not alert the user when he does not travel to that particular location at all. For example, in a scenario where the user needs the reminder to alert him when he
reaches post office, it would not be useful if the user does not travel near to the post office. So to improve the use of reminders in these scenarios, combination of both time and location have been taken into account. There are many location based reminder applications available in the market. In which only very few applications make use of time along with the location to remind users [2]. If the user misses an alert, there is no substitute in existing applications. The shortcomings and inadequate features of location based reminders motivated me to develop an effective reminder application.

2.3 Project Scope

Application is developed for android based smart phones. To avail the features of this application user must switch on the GPS of his/her smartphone and should connect to the internet. According to Statista (an online data analysis website), by May 2016 97.7% of android users are using Android 4.0 (Ice Cream Sandwich) or later versions. This reminder application is compatible from Android 4.0 to latest android version Android 6.0 (Marshmallow).

2.4 Functionalities

iDoRemind application provides the following functionalities.

i. Allows users to create location reminders and alert when they approach the location.

ii. Provides feature to make the reminder date and time restricted.

iii. Able to suggest the user based on preferences. Sorting out the results by the preference ratio value.

iv. A feature to enter user preferences.

v. Provide each user a unique code, which the others can use to add him in to their group.
vi. Automatically alert the members in group, if any person in his/her group travels near the location in the reminder (providing the user assigns the reminder to the group). (User can select members whom he/she wants to share) [15].

vii. Share the reminder with other people in group when reminder remains unattended. (Ignore button)

viii. Alert the user when he/she travels near similar kind of stores to that of the one in reminder like if user create a reminder to remind at Starbucks *iDoRemind* application must remind him if he travels through some other coffee shop such as coffee waves. If there are more than one of similar kind results should be sorted in order of preference value.

Table 2.1 Contains the comparison of functionalities of *iDoRemind* application with other applications or the papers in the area of reminders based on location.

<table>
<thead>
<tr>
<th>Features</th>
<th>Location Reminder s</th>
<th>NT</th>
<th>Google Keep</th>
<th>Place -Its</th>
<th>Remind me</th>
<th>iDoRemind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reminder based on location</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Time Restriction</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Date Restriction</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Repeat</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Groups</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Preferences</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Near by places suggestions</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>Sharing via social media</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Delete Reminder</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Picking location on map</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

*Table 2.1 Comparison of iDoRemind with other applications.*
3. DEVELOPED SYSTEM DESIGN

The architecture of the developed reminder application is shown in Figure 3.1. A *iDoRemind* consists of Presentation layer, business logic, rest access layer, server, Shared preferences, Location Manager. Server contains webserver used to establish connection with centralized database and centralized database itself.

![Figure 3.1 Architecture of iDoRemind Application](image)

In this architecture, presentation layer contains the UI. Rest access layer is used to establish communication between web server and application client. To access the centralized database in Amazon Webserver (AWS) rest API’s had been used. When user made the request to create reminder, the request will be processed by web server in Amazon Webserver and create a table in centralized database with the information like
reminder title, location information (latitude, longitude values), any restrictions, and the user or users of the group reminder assigned to. Locations and location related information are obtained via the location manager, Google Places API and Google Maps API. Location manager is used to obtain user’s current location. Google Places API is used to get nearby places, average rating, and cost range. Shared preferences store the user details like name, user ID, email. Based on the details obtained from the shared preferences AWS process the request. Apart from performing the whole tasks, business logic will process the data from API’s.

3.1 Use Case Diagram

The use case diagram for iDoRemind is shown in Figure 3.2. The purpose of use case is to present graphical overview of the functionalities provided by a system. The oval shapes represent the functionality of the user. As name represents login with Facebook / google sign in will helps the user to login in to the application. The reason behind implementing Facebook / Google login is to avoid the users from adding one more user name and password in already long list of their usernames and passwords.

The create reminder helps user to create location based reminder with or without restrictions and with or without repetition. Enter preferences provides a list of option to select user preferences. Unique code displays a unique code for each user to share with other users and form as groups. The Add unique code functionality will help the user to enter unique code of other users and add them to his/her group. Your preferences will display the preferences you selected. Suggestions will give you the places you might interested based on your preferences. Near by places will display the alternatives to the places in reminders. Suggestions and Near by places can be turned on/off in suggestions.
Figure 3.2 Use Case Diagram of iDoRemind Application
3.2 Class Diagram

Class Diagram is a type of static diagram. It represents classes in the system their attributes, operations and relations among the objects. The class diagram for the iDoRemind is shown in the Figure 3.3. The class diagram shows the major classes (functionalities) in the project. For each class, class diagram shows the information about its methods.
3.3 User Interface

This section gives the overview of user interface of the application. User interface is where user interacts with the application. Figure 3.4 displays the menu or dashboard of the iDoRemind.

![Menu of iDoRemind Application](image)

**Figure 3.4 Menu of iDoRemind Application**

3.3.1 Login

To prevent the unauthorized use, the system resources can be utilized by the users who logged in to the application. In today’s world with increasing use of mobile applications it will be difficult for the users to remember username and password for every account. So to make it easy for the user the user can login into the iDoRemind through his/her Facebook or Google account. The login screen of the iDoRemind is shown in Figure 3.5.
3.3.2 Create Reminder

On selecting the create reminder the user will redirected to the reminder creation page where he can enter the details of the reminder. The user needs to enter the title, select the location, and can restrict it based on time or date. The user can also set the frequency of the reminder. The create reminder screen is shown in Figure 3.6(a). Clicking on pick button the application will be directed to google maps. Here the user will select the location. Select location screen is shown in Figure 3.6(b)
Figure 3.6 Create Reminder and Select Location, Date, Time, Repeat Screens

Once the reminder created user can share reminder with his family/friends. The user can assign the reminder to the other people in his group. The reminders can be deleted anytime. Figure 3.7(a) shows the response when user clicks on delete button. If the user wants to assign reminder to some people in the group, clicking on assign users will give the response as displayed in figure 3.7(b). If the user clicks on share, all the available options to share will be displayed on screen, the same demonstrated in the Figure 3.7(c).
3.3.3 Preferences

The user can select his preferences by checking the boxes among the options provided in the *Enter Preferences*. These are the places user likely to visit frequently. The user can check his preferences by selecting the option *Your Preferences*. Selecting the preferences and the results in your preferences is shown in Figure 3.8.
3.3.4 Unique Code

*Unique Code* displays users exclusive token which he can share with his family or friends to form as groups. The groups work in one-way fashion for example user x adds user y, user y will be in group of user x but not vice versa. For user x to be in group of user y user y should add the token of user x using the option *Add Unique Code*. As mentioned *Add Unique Code* helps the user to add other people to his group. While assigning a reminder to other users in group the user can select the users to share the reminder. Unique token and adding the token of other user is shown in Figure 3.9.
3.3.5 Group Members

By selecting the Group Members option user can see all the members in his group. The user can delete any other user(s) from his group by clicking on delete button beside user name. Figure 3.10 shows a sample screen of group members. If the user added any new token the user to whom that token belongs to will be added to the user groups. Based on the token the user details will be fetched from centralized database and displays the user name in group members.
3.3.6 Suggestions

Based on the preferences selected by the user *iDoRemind* autonomously suggest some places the user might want to visit. For example, if the user selects café in preferences. When the user travels to some new place he can check in suggestions for the cafes in that area, all the cafes within 400meters will be displayed. Sample screen of suggestions is shown in Figure 3.11. The results are sorted in the order of preference ratio, calculated using average rating and cost range. The formula used to calculate preference ratio is \( \text{Preference ratio} = \frac{\text{average rating}}{\text{cost range}} \). Average rating is obtained from the Google Places API. The cost range obtained from Google Places API is converted into integers and used in this formula. If the cost range is $ it will be converted to 1, $$ will be 2 and $$$ will be 3. The user can navigate by clicking on navigate button in suggestions.
3.3.7 Near by locations

Near by locations is similar to suggestions. This option also has some suggestions but not based on preferences. Here the suggestions will be displayed based on the reminder. For example, if the user wants to go to some grocery store but for some reason he is not able to travel to that location he can check the grocery stores within 400meters in user’s current location using this option. Figure 3.12 represents some of the nearby places suggestions.
3.3.8 Settings

Some users might not like the notifications about suggestions from application. In such cases user can turn off the suggestions and near by places notifications in the settings. Settings screen is shown in figure 3.13.
3.3.9 Logout

If the user wants to logout of the application, he can click on the logout option which will take him to the login screen. But to avail the services of *iDoRemind* user needs to be logged in and his mobile’s GPS enabled.

3.3.10 Notifications

Whenever user reaches the location in the reminder or if there are some suggestion or near by places they should be reminding as notification. *iDoRemind* works with 400meters range and 0.5meters accuracy. User can see navigate and ignore option in the notification. If the user selects navigate the app should redirect to google map.
showing the directions to the destination from the current location. If the user selects ignore the reminder should be shared to the other users. Figure 3.14 shows a sample notification with 2 buttons navigate and ignore.

Figure 3.14 Notification of the Reminder
4. IMPLEMENTATION OF APPLICATION MODULES

4.1 Login

To use the Facebook and Gmail login, the libraries shown in Figure 4.1 should be added in the `build.gradle` file. For the first time, the user needs to authenticate the application.

```java
compile 'com.google.android.gms:play-services:8.1.0'
compile 'com.facebook.android:facebook-android-sdk:4.7.0'
```

**Figure 4.1 Adding Libraries to build.gradle File**

For the authentication, AsyncTask is created, which runs in background. If the authorization is successful, AsyncTask will return a JSON array with all the details of the user. Gmail Authentication is shown in Figure 4.2. Facebook authentication is also similar.

```java
private class EmailLoginTaskRunner extends AsyncTask<Void, Void, Void> {
    private String email;
    private String name;
    JSONArray jsonArray;
    ProgressDialog progressDialog;
    Activity activity;

    public EmailLoginTaskRunner(String email, String name, Activity activity) {
        this.email = email;
        this.name = name;
        this.activity = activity;
    }

    @Override
    protected Void doInBackground(Void... params) {
        UserLoginService login = new UserLoginService();
        jsonArray = login.loginUserGmail(email, name);
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        progressDialog.dismiss();
        if (jsonArray != null) {
            postverifiedLogindetails(jsonArray);
        }
    }
}
```

**Figure 4.2 Gmail Authentication**
4.2 Setting permission in AndroidManifest.xml file

AndroidManifest.xml is a must file for every android application. All the permissions using in the application should mention in this file. AndroidManifest.xml contains all the essential information about application. Figure 4.3 displays the permissions of *iDoRemind*. Internet permission is required to make a request to Amazon Webserver. Access_Fine_Location permission allows the application to get information about latitude, longitude values.

```xml
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.READ_PROFILE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Figure 4.3 Setting Permissions in AndroidManifest.xml file

4.3 Creating the Reminder

This module deals with creating the reminder. The user needs to enter the title, pick the location using location picker, applying date or time restrictions if he need any, entering the number of days if the user wants the reminder to repeat. When users select the location `PlacePicker.getPlace()` method will get the details about the location selected. Using the reverse geocoding the values will be converted into human readable text format and displayed in text view. The same thing (getting details about location and
displaying in text format) is shown in the Figure 4.4.

```java
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
if (requestCode == RESULT_OK) {
    Place place = PlacePicker.getData(this);
    Latitude = place.getLatitude();
    reminderModel.setLatitude("" + latitude.latitude);
    reminderModel.setLongitude("" + latitude.longitude);

    TextView locationInfo = (TextView) findViewById(R.id.locationInfo);
    locationInfo.setVisibility(View.VISIBLE);
    Geocoder gc = new Geocoder(this, Locale.getDefault());
    try {
        List<Address> addresses = gc.getFromLocation(latitude.latitude, latitude.longitude, 1);
        if (addresses.size() > 0) {
            locationInfo.setText(addresses.get(0).getAddressLine(0) + " + addresses.get(0).getAddressLine(1) + " +
            reminderModel.setReminderInfo(locationInfo.getText().toString());
        } else {
            locationInfo.setText(latitude.latitude + " + latitude.longitude);
            reminderModel.setReminderInfo(locationInfo.getText().toString());
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

public void pickLocation(View view) {
    PlacePicker.IntentBuilder builder = new PlacePicker.IntentBuilder();
    Context context = getApplicationContext();
    try {
        startActivityForResult(builder.build(context), PLACE_PICKED);
    } catch (GooglePlayServicesRepairableException e) {
        e.printStackTrace();
    } catch (GooglePlayServicesNotAvailableException e) {

    }
}
```

**Figure 4.4 Code to Pick the Location to Create Reminder**

### 4.4 Connecting to Database

The rest service will send the reminder data to Amazon Webserver (AWS). To store the data in database, database connection need to be established in the webserver. The connection to database can be established by using `mysql_connect()` method. The code snippet to connect database is shown in the Figure 4.5. Once the database established the rest service will create the http request using the method `new DefaultHttpClient()`. The response will be sent using the method `httpClient.execute(httpPost)`. The response rest service received will be in JSON format.

The request and response of the rest service is shown in Figure 4.6. The JSON data needs to be parsed to human readable format. Here the JSON data processed and displays in the form of list view. The JSON parsing is done by using `JsonObject.getString(object)`.
Figure 4.7 shows the how reminder data is inserted in to list view.

```php
<?php

class DB_CONNECT {

    function __construct() {
        $this->connect();
    }

    function __destruct() {
        $this->close();
    }

    function connect() {
        include_once __DIR__ . '/db_config.php';
        $conf = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or die(mysql_error());
        $db = mysql_select_db(DB_DATABASE) or die(mysql_error()) or die(mysql_error());
        return $conn;
    }

    function close() {
        mysql_close();
    }
}

?>
```

**Figure 4.5 Database Connection in Webserver**

```java
try {
    BufferedReader reader = new BufferedReader(new InputStreamReader(
        is, "iso-8859-1"), 3);
    StringBuilder sb = new StringBuilder();
    String line = null;
    while ((line = reader.readLine()) != null) {
```

**Figure 4.6 Request and Response of Rest Service**
4.5 Adding Member to Group

The Users can form into groups to share reminders with each other. Every user is presented with a unique code by adding that code in add token field anyone with iDoRemind can add that user to their group. On clicking the done after adding the token, upon validation addGroupMember() method will be called. This method will pass the user id of the user and the person he wanted to add to the web server, where the request will be processed by adding the received IDs to the group ID. Figure 4.8 shows the code representation of adding a member to the group.
When the user assigns his reminder to the other users in the group if any of that member(s) travel to the location of the reminder it will alert that member. Also when the user ignored the reminder the application will share the reminder with the user’s group members. Clicking on ignore button `onReceive()` method in MyBroadcastReceiver class will be called. It will send the reminder ID, User ID (from shared preferences) to `assignReminders()` method in reminderService class. A request will be created to AWS using reminder ID and user ID by Rest API. AWS will broadcast the reminder to all the members in the user’s group. The whole process is shown in form of code in Figure 4.9.

```java
public class MyBroadcastReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        System.out.println("Testing");
        int id = intent.getIntExtra("notification_id", 1);
        final int reminderID = intent.getIntExtra("reminder_id", 1);
        Toast.makeText(context, "Reminders sharing with others in your group", Toast.LENGTH_SHORT).show();
        final SharedPreferences sp = context.getSharedPreferences(Preferences, Context.MODE_PRIVATE);
        NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancel(id);
        task = new AsyncTask<Void, Void, Void>() {
            @Override
            protected Void doInBackground(Void... params) {
                synchronized (myService) {
                    ReminderService reminderService = new ReminderService();
                    reminderService.assignReminders(reminderId, sp.getString("userid", ""));
                }
                return null;
            }

            @Override
            protected void onPostExecute(Void result) {
                // TODO Auto-generated method stub
                super.onPostExecute(result);
            }
        };
    }
}
```

Figure 4.9 Sharing the Message to the Others in the Group
4.6 Suggestions based on Preferences

The user can select his preferences in the preferences menu. Once selected the preferences will be updated in AWS. If the settings for the suggestions is on, PreferenceService() will create request with user ID, latitude and longitude values. The AWS will return the preference array list in JSON format. PreferenceService will process the data and return the places that matches the categories of all the user preferences. the iDoRemind will give suggestions on the places to visit in the current location based on his preferences. The results will be displayed in the increasing order of the value of preference ratio which will be calculated using the formulae (average rating/cost range).

The default value for average rating is 2.5 and cost range is 1.5. Figure 4.10 shows how preferences are stored into array list (background).

```java
public void loadSuggestion(String uid, String lat, String lng) {
    task = new AsyncTask<Void, Void, Void>() {
        @Override
        protected Void doInBackground(Void... params) {

            PreferenceService preferenceService = new PreferenceService();
            ArrayList<Preference> preferenceArrayList = preferenceService.getPreferences(uid, "getPreferences");

            preferenceReminders.clear();
            if (preferenceArrayList.size() > 0) {
                loadPreferencesService = new loadPreferencesService();
                preferenceReminders.addAll(loadPreferencesService.makeHttpRequest(preferenceArrayList, lat, lng));
            }
            return null;
        }
    }
}
```

Figure 4.10 Inserting Preferences into Array List

4.7 Suggestions based on Reminders

Based on the user reminders iDoRemind will suggest the user about the places the user might be interested. As soon as the reminder created if there is any keyword it will be saved into a new array created by PreferenceReminder class. Whenever a reminder is created, the keyword will be appended into array. By matching the keyword in the current
latitude and longitude values a set of results will be collected from google maps API. This list will be displayed in near by places tab. The results will be sorted by the preference ratio value. Populating the near by places in list view is shown in Figure 4.11.

```java
public class NearbyplacesActivity extends BaseActivity {
    ListView listview;
    ArrayList<PreferenceReminder> preferenceReminders = new ArrayList<>();
    private NearbyPlacesAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nearby_places);
        listview = (ListView) findViewById(R.id.reminder_list);

        GPSLocator locator = new GPSLocator(this);
        locator.getMyLocation();
        double latitude = locator.getLatitude();
        double longitude = locator.getLongitude();

        LatLng currentLocation = new LatLng(latitude, longitude);

        adapter = new NearbyPlacesAdapter(this, 0, preferenceReminders, currentLocation);
        listview.setAdapter(adapter);

        loadSuggestions(sp.getString("userid", ""), "" + latitude, "" + longitude);
        setTitle("Near by places");
    }
}
```

**Figure 4.11 Populating Near By Places in List View**

### 4.8 Settings

If the user doesn’t want the suggestions from the application. So *iDoRemind* has the option to turn off the notifications about suggestions based on preferences(suggestions) and notifications about suggestions based on reminders (near by places). Figure 4.12 shows the code for settings.
public void onchange(View view) {
    boolean on = ((Switch) view).isChecked();
    SharedPreferences.Editor editor = sp.edit();
    if (on) {
        editor.putBoolean("mmotis", true);
    } else {
        editor.putBoolean("mmotis", false);
    }
    editor.commit();
}

public void onchange(View view) {
    boolean on = ((Switch) view).isChecked();
    SharedPreferences.Editor editor = sp.edit();
    if (on) {
        editor.putBoolean("mmotis", true);
    } else {
        editor.putBoolean("mmotis", false);
    }
    editor.commit();
}

Figure 4.12 Settings
5. TESTING AND EVALUATION

Testing and evaluation includes testing all the functionalities of the application. This application is tested using Nexus 4 mobile phone with android version 5.1.1. This application is compatible for android version 4.0 and latest. Each module of the application is tested with all the possible test cases.

5.1 Launching the application

When the user launches the application for the first time he will see the login screen. Figure 5.1 shows the login screen and dashboard of iDoRemind.

![Login Screen and Dashboard of iDoRemind Application](image)

Figure 5.1 Login Screen and Dashboard of iDoRemind Application

When the user logs in for the first time he need to authorize the application. Figure 5.2 shows the authorization of application. The home screen contains the reminders of the user.
5.2 Creating the Reminder

The user can create reminder by entering the title and picking the location. If the user needs/wanted to restrict he/she can restrict it based on time or date or both. If the user wants a particular reminder to be reminded periodically he can set the period using the option repeat. For example, the user needs to collect the pay for every two weeks, he can set the reminder to repeat for every 14 days. Input validation has been done by leaving the title blank, not picking up the location, entering the to date earlier than from date. The validation output has been shown in Figure 5.3.
Figure 5.3 Validation of Creating Reminder (Title, Location, Date)

The reminder with and without restriction are shown in the Figure 5.4. The restrictions can be of date restriction, time restriction, repetition period. The 2\textsuperscript{nd} reminder in the Figure is restricted between dates 11/29/2016 to 1/02/2017 between 7:05 am to 7:06 pm. It is also set to repeat every 14 days where as first reminder in the Figure doesn’t contain any restrictions. Only one of the restrictions can also be applied. All the restrictions are mutual exclusive and not depend on each other.
5.3 Unique Code

Unique code displays the exclusive token given to the user. The user can share the token to family friends and users using share option. Sharing of token using different applications has been tested. Sharing via WhatsApp is shown in Figure 5.5.
5.4 Add Unique Code

Unique code has been tested by adding codes of different users. If we enter invalid code, it will show the error message. On adding valid code token added successfully message will be displayed. Figure 5.6 shows successful and unsuccessful validation of add token.

![Add Token Validation](image)

**Figure 5.6 Add Token Validation**

5.5 Suggestion based on preferences and Preferences

This feature has been evaluated by selecting different preferences at different locations. Possible test cases have been tested. One such test case is displayed in Figure 5.7, where the preferences have been selected as library and café. When I arrived in the campus, the results shown as in Figure 5.7.
5.6 Near by places based on reminders

This option is tested by creating different kinds of reminders. When travelling to different places various nearby places had been displayed. Figure 5.8 shows the output when reminder was created on food (restaurant) and gym and the user is in TAMUCC campus.
5.7 Settings

The settings option was tested by turning off suggestions, and near by places. The combinations of “turn off both”, “turn off suggestions notifications, turn on near by notifications”, “turn on suggestions notifications, turn off near by notifications”, “turn on both” have been tested and the results for the notification are as expected. Figure 5.9(a) shows the notifications when both the suggestions and near by places turned off and Figure 5.9(b) shows the notifications when both the suggestions and near by places are turned on in settings.

Figure 5.9 Notitions Based on Suggestions Status
Figure 5.10(a) shows the notifications when only suggestions notifications turned on. Figure 5.10(b) shows the notifications when only near by notifications is on.

![Figure 5.10 Notifications When Only One Notification is On](image)

5.8 Logout

Clicking on logout should take the user to the login screen. The user should not be able to use any functionality of the application (except login) until he login again.

5.9 Reminder

After the reminder created the user should be able to see the reminder on the home screen with 3 buttons on it. Delete button should delete the reminder. The delete
button was tested by deleting the existing individual reminders and group reminders. For the group reminders if the user who creates the reminder deletes it, it will be deleted for every user. If some user in group for whom the reminder is assigned, deletes the reminder it will delete only for that user but not for whole group. Assign Users functionality is to assign the reminder to the selected users in the group. The same had been tested by assign a reminder to multiple users. One of the test cases is represented in Figure 5.11. In Figure 5.11 the reminder selected to share with user named “Vinay Datta” and successfully assigning the reminder to the selected user can be seen in the Figure 5.11.

![Figure 5.11 Assign Users](image)

By using the share option user should be able to share the reminder through all possible ways i.e. text messaging, WhatsApp, Gmail, messenger etc. I tried sharing a reminder through different options and able to do it. Among them sharing via text messaging is captured and displayed in Figure 5.12.
5.10 Reminder Status

The reminder will be shown the status of pending when created. Until the user notified about reminder it will be in pending status. Once the reminder triggered the status will change to reminded.

Figure 5.12 Sharing the Reminder via Text Message

Figure 5.13 Reminder Status
5.11 Notification

The reminder will be shown as notification with two options “navigate” and “ignore”. Clicking on navigate will redirect you to the map showing the directions to the location in reminder from your current location. Even in near by places and suggestions navigate button will work the same way, it will redirect to maps and show directions.

![Figure 5.14 Getting the Directions to Destination by Clicking on Navigate](image)

Clicking on ignore will share the reminder with your group members. Ignore is created for the case when the user was unable to attain the reminder he can share with others.
5.12 Evaluation

In order to evaluate iDoRemind feedback collected from 15 users who were using the application from one week to 15 days. The feedback consists of eight questions; the users have option to rate each question between [0,10], where 0 indicates disagree and 10 indicates strongly agree. Table 5.1 shows the list of questions in the feedback and average rating for each question. Figure 5.16 shows the graphical representation of the feedback.
<table>
<thead>
<tr>
<th>Question</th>
<th>Average rating</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>iDoRemind</em> gives notifications as expected</td>
<td>9.73</td>
</tr>
<tr>
<td>Applying restriction to reminder is useful</td>
<td>9.07</td>
</tr>
<tr>
<td>Groups option is useful</td>
<td>9.8</td>
</tr>
<tr>
<td>Entering preferences and suggestions based on it are useful</td>
<td>8.6</td>
</tr>
<tr>
<td>Near by places suggestions from application is useful</td>
<td>9.07</td>
</tr>
<tr>
<td>Sharing The reminder via social media is useful</td>
<td>8.4</td>
</tr>
<tr>
<td>Sharing the reminder with others when I can't attain (clicking on Ignore button) is useful</td>
<td>9.2</td>
</tr>
<tr>
<td>The user interface is easy to understand.</td>
<td>9.5</td>
</tr>
<tr>
<td>Reminding alerts are clear.</td>
<td>9.27</td>
</tr>
<tr>
<td>Navigating the application is easy.</td>
<td>9.5</td>
</tr>
<tr>
<td>Setting up the tasks is easy.</td>
<td>9.43</td>
</tr>
<tr>
<td><em>iDoRemind</em> application is satisfying.</td>
<td>9.5</td>
</tr>
<tr>
<td>I would recommend <em>iDoRemind</em> to other people.</td>
<td>9.73</td>
</tr>
</tbody>
</table>

Table 5.1 List of Questions in Feedback and Their Average Rating
Figure 5.16 Graphical Representation of Feedback

Looking at the evaluations the results indicate that among all the options/questions sharing via social media got the least rating (8.4). The option “Sharing via social apps” is useful when the some of the family members or friends have iPhone or some other OS phones and some have android phones. That is not the case with the people participated in evaluation, so it got the lowest rating. Preferences offers very little set of options so it got the second least rating (8.6). If we look at other side of the table i.e. highest rating “Groups option” got the highest rating (9.8). No other application is providing the group reminder option (at least to participants and developers knowledge). So the participants found the option very interesting and useful.
6. CONCLUSION AND FUTURE WORK

$iDoRemind$ is a new application for smart phones and tablets with android operating system. This application allows the user to login using their Facebook or Google account. $iDoRemind$ application helps user to create location based reminder. These reminders can be restricted with date, time and user can set it to repeat for particular period of time. Based on these conditions the $iDoRemind$ should remind the user when he reaches near the location in the reminder. This application has some advanced features such as allowing the user to create groups and assign his reminder to selected members in group. $iDoRemind$ alerts if any member of the group travels near the location in the reminder. Advanced features include giving suggestions based on user preferences and reminders. This application let the user share the reminder to the other people (people who are not the users of $iDoRemind$) via external applications like WhatsApp, text messaging, messenger, Gmail etc. $iDoRemind$ also provides an option to turn off the suggestions from the application.

Future Work

This application can be improved by adding following functionalities.

- Categorizing the reminders based on location which helps in reducing the battery consumption. Now the application will keep an eye on every reminder created by user, based on the location if the reminders with far away location kept idle, it will reduce battery consumption.

- Option to change the radius according to the users need.

- Providing the personalized preferences using data mining on user activity.

- Ability to create groups automatically by analyzing the Wi-Fi patterns of
the user.

- Option to create multiple groups like office, family, friends etc.
- Making the application more autonomous by combining data analysis with trend analysis such as storing the data of the user reminders and observing the patterns of the user activity and giving the suggestions based on that.
- Providing the option to comment on reminders (to-dos), this will be useful especially in case of group reminders.
BIBLIOGRAPHY and REFERENCES


APPENDIX A: CODE SNIPPETS

This section contains code snippets of the main tasks of the project.

UserLoginService.java

```java
package com.location.reminder.com.location.reminder.restcalls;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import java.util.ArrayList;

public class UserLoginService {

    public JSONArray longWithFacebook(String facebookid, String email, String name) {
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("facebookid", facebookid));
        postdata.add(new BasicNameValuePair("name", name));
        postdata.add(new BasicNameValuePair("email", email));
        RestService getjson = new RestService();
        JSONArray jsonArray = getjson.makeHttpRequest(postdata, "facebook_registration.php");
        return jsonArray;
    }

    public JSONArray loginUserGmail(String email, String name) {
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("name", name));
        postdata.add(new BasicNameValuePair("email", email));
        RestService getjson = new RestService();
        JSONArray jsonArray = getjson.makeHttpRequest(postdata, "gmail_registration.php");
        return jsonArray;
    }
}
```

Database Creation and Connection

db_config.php
<?php
define('DB_USER', "root");
define('DB_PASSWORD', "");
define('DB_DATABASE', "reminderapp");
define('DB_SERVER', "localhost");
?>

db_connect.php

<?php
    class DB_CONNECT {
        function __construct() {
            $this->connect();
        }
        function __destruct() {
            $this->close();
        }
        function connect() {
            include_once __DIR__ . '/db_config.php';
            $con = mysql_connect(DB_SERVER, DB_USER, DB_PASSWORD) or die(mysql_error());
            $db = mysql_select_db(DB_DATABASE) or die(mysql_error());
            return $con;
        }
        function close() {
            mysql_close();
        }
    }
?>
HomeActivity.java

```java
public class HomeActivity extends BaseActivity
  implements NavigationView.OnNavigationItemSelectedListener {
  SharedPreferences sharedpreferences;
  private ListView mList;
  private ReminderAdapter mAdapter;
  private MultiSelector mMultiSelector = new MultiSelector();
  ArrayList<ReminderModel> mItems = new ArrayList<ReminderModel>();
  private boolean isMyServiceRunning(Class<?> serviceClass) {
    ActivityManager manager = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
    for (ActivityManager.RunningServiceInfo service : manager.getRunningServices(Integer.MAX_VALUE)) {
      if (serviceClass.getName().equals(service.service.getClassName())) {
        return true;
      }
    }
    return false;
  }
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    needhomebutton = false;
    setTitle("Location based reminders");
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_home);
    mList = (ListView) findViewById(R.id.reminder_list);
    sharedpreferences = getSharedPreferences(Constants.sharedPreferences,
      Context.MODE_PRIVATE);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    FloatingActionButton fab = (FloatingActionButton)
      findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        onResume();
      }
    });
    DrawerLayout drawer = (DrawerLayout)
      findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
      this, drawer, toolbar, R.string.navigation_drawer_open,
      R.string.navigation_drawer_close);
    drawer.setDrawerListener(toggle);
    toggle.syncState();
    NavigationView navigationView = (NavigationView)
      findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
  }
```
View headerLayout = 
navigationView.getHeaderView(0); 
TextView user_fullname = (TextView) headerLayout.findViewById(R.id.username); 
user_fullname.setText(sharedpreferences.getString("name", "")); 
TextView email = (TextView) headerLayout.findViewById(R.id.email); 
email.setText(sharedpreferences.getString("email", "")); 
mAdapter = new ReminderAdapter(this, 0, mItems, 
sharedpreferences.getString("userid", "")); 
if (!isMyServiceRunning(AppLocationManager.class)) {
    startService(new Intent(this, AppLocationManager.class));
}
public void create_new() {
    Intent intent = new Intent(this, CreateNewReminder.class);
    startActivity(intent);
}
@Override
public void onBackPressed() {
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.home, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();
}
if (id == R.id.nav_logout) {
    logout_user();
    return true;
} else if (id == R.id.nav_reminder) {
    create_new();
} else if (id == R.id.nav_generatecode) {
    Intent intent = new Intent(this, UniqueTokenActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_addcode) {
    Intent intent = new Intent(this, AddTokenActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_Groupmembers) {
    Intent intent = new Intent(this, ShowGroupsActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_groupmembers) {
    Intent intent = new Intent(this, ShowGroupsActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_preferences) {
    Intent intent = new Intent(this, EnterPreferencesActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_showpreferences) {
    Intent intent = new Intent(this, NearbyPlacesActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_yourpreferences) {
    Intent intent = new Intent(this, YourPreferencesActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_suggestions) {
    Intent intent = new Intent(this, SuggestionsActivity.class);
    startActivity(intent);
    return true;
} else if (id == R.id.nav_settings) {
    Intent intent = new Intent(this, SettingsActivity.class);
    startActivity(intent);
    return true;
}
DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
drawer.closeDrawer(GravityCompat.START);
return true;

@Override
public void onResume() {
    super.onResume();
    System.out.println("EXECUTE THIS");
    new LoadRemindersTask(this).execute(null, null, null);
}

private class LoadRemindersTask extends AsyncTask<Void, Void, Void> {
    Activity activity;
    ProgressDialog progressdialog;

    public LoadRemindersTask(Activity activity) {
        this.activity = activity;
    }

    @Override
    protected void onPreExecute() {
        progressdialog = ProgressDialog.show(activity, ",", "Please wait...", true);
        mItems.clear();
        mAdapter.notifyDataSetChanged();
    }

    @Override
    protected Void doInBackground(Void... params) {
        ReminderService reminderService = new ReminderService();
        ArrayList<ReminderModel> newitems = reminderService.getReminders(sharedpreferences.getString("userid", ",");
        System.out.println("Number of reminders" + newitems.size());
        if (newitems != null) {
            mItems.addAll(newitems);
        }
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        progressdialog.dismiss();
        mAdapter.notifyDataSetChanged();
        mList.setAdapter(mAdapter);
    }
}

LoadPreferencesService.java

```java
public class LoadPreferencesService {
    public ArrayList<PreferenceReminder> makeHttpRequest(ArrayList<Preference> preferences, String clat, String clng) {
        if (preferences.size() == 0) {
            return new ArrayList<PreferenceReminder>();
        }
        InputStream is = null;
        JSONObject jsonObject = null;
        String json = "";
        try {
            String types = "";
            for (Preference preference : preferences) {
                types += preference.getName().toLowerCase() + "|";
            }
            if (types.endsWith("|")) {
                types = types.substring(0, types.length() - 1);
            }
            clat = URLEncoder.encode(clat, "UTF-8");
            clng = URLEncoder.encode(clng, "UTF-8");
            types = URLEncoder.encode(types, "UTF-8");
            String key = URLEncoder.encode("AIzaSyC0HVwfm7Z_GC8-fAD2BrEw6woyN1zClg", "UTF-8");
            String url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=" + clat + "," + clng + "&radius=400&types=" + types + "&key=" + key + "&";
            System.out.println("URL" + url);
            DefaultHttpClient httpClient = new DefaultHttpClient();
            HttpGet httpGet = new HttpGet(url);
            HttpResponse httpResponse = httpClient.execute(httpGet);
            HttpEntity httpEntity = httpResponse.getEntity();
            is = httpEntity.getContent();
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (ClientProtocolException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        try {
            BufferedReader reader = new BufferedReader(new InputStreamReader(is, "iso-8859-1"), 8);
            StringBuilder sb = new StringBuilder();
            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            is.close();
            json = sb.toString();
        } catch (Exception e) {
            e.printStackTrace();
        }
        ArrayList<PreferenceReminder> preferenceReminders = new ArrayList<>();
        try {
            System.out.println(json);
            JSONObject jsonObject = new JSONObject(json);
            JSONArray jsonArray = new JSONArray(jsonObject.getString("results"));
            for (int i = 0; i < jsonArray.length(); i++) {
                String lat = jsonArray.getJSONObject(i).getJSONObject("geometry").getString("lat");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```
String lng = 
JSONArray.getJSONObject(i).getJSONObject("geometry").getJSONObject("location").get 
String("lng"); 
String name = JSONArray.getJSONObject(i).getString("name"); 
String types = JSONArray.getJSONObject(i).getString("types"); 
String icon = JSONArray.getJSONObject(i).getString("icon"); 

double rating = 2.5; 
double price_level = 1.5; 
double preferencerating = 0; 

if (JSONArray.getJSONObject(i).has("rating")) { 
    rating = JSONArray.getJSONObject(i).getDouble("rating"); 
} 
if (JSONArray.getJSONObject(i).has("price_level")) { 
    price_level = JSONArray.getJSONObject(i).getDouble("price_level"); 
    preferencerating = rating / price_level; 

    PreferenceReminder preferenceReminder = new PreferenceReminder(); 
    preferenceReminder.setPreferencetype(types); 
    preferenceReminder.setLatitude(lat); 
    preferenceReminder.setLongitude(lng); 
    preferenceReminder.setTitle(name); 
    preferenceReminder.setImage(icon); 
    preferenceReminder.setRating(rating); 
    preferenceReminder.setPrice_level(price_level); 
    preferenceReminder.setPreferencerating(preferencerating); 
    preferenceReminders.add(preferenceReminder); 
} 

class StudentDateComparator implements Comparator<PreferenceReminder> { 
    public int compare(PreferenceReminder s1, PreferenceReminder s2) { 
        return Double.compare(s2.getPreferencerating(), 
        s1.getPreferencerating()); 
    } 
}

ArrayList<PreferenceReminder> infos = new ArrayList<PreferenceReminder>(); 
Collections.sort(infos, new StudentDateComparator()); 

return preferenceReminders; 
}
public class GroupsService {
    public void updateGroups(String reminderid, ArrayList<Integer> userids) {
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("reminderid", reminderid));
        for (int i : userids) {
            postdata.add(new BasicNameValuePair("userids[]", "" + i));
        }
        RestService getjson = new RestService();
        getjson.makeHttpRequest(postdata, "addusertoreminder.php" );
    }
    public JSONArray addGroupMember(String userid, String otheruserid) {
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("userid", userid));
        postdata.add(new BasicNameValuePair("otheruserid", otheruserid));
        RestService getjson = new RestService();
        return getjson.makeHttpRequest(postdata, "addGroup.php");
    }
    public void deleteGroupMember(String userid, String otheruserid) {
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("userid", userid));
        postdata.add(new BasicNameValuePair("otheruserid", otheruserid));
        RestService getjson = new RestService();
        getjson.makeHttpRequest(postdata, "deleteuserfromGroup.php");
    }
    public ArrayList<Groups> getGroupMember(String userid) {
        ArrayList<Groups> groupmembers = new ArrayList<>();
        final ArrayList<NameValuePair> postdata = new ArrayList<NameValuePair>();
        postdata.add(new BasicNameValuePair("userid", userid));
        RestService getjson = new RestService();
        JSONArray jsonarray = getjson.makeHttpRequest(postdata, "getGroups.php");
        for (int i = 0; i < jsonarray.length(); i++) {
            try {
                JSONObject jsonObject = jsonarray.getJSONObject(i);
                Groups group = new Groups();
                group.setId(Integer.parseInt(jsonObject.getString("id")));
                group.setName(jsonObject.getString("name"));
                groupmembers.add(group);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        return groupmembers;
    }
}
Creating Notification Manager

AppLocationManager.java

```java
public void createNotification(ReminderModel reminder, String clat, String clong, 
String dlat, String dlong) {
    Intent intent = new Intent(android.content.Intent.ACTION_VIEW,
        Uri.parse("http://maps.google.com/maps?saddr=" + clat + "," + clong + 
        "&daddr=" + dlat + "," + dlong + ";"));
    PendingIntent pIntent = PendingIntent.getActivity(this, (int)
System.currentTimeMillis(), intent, 0);
    int notificationId = (int) System.currentTimeMillis();
    Intent cancelIntent = new Intent(this, MyBroadcastReceiver.class);
    Bundle extras = new Bundle();
    extras.putInt("notification_id", notificationId);
    extras.putInt("reminder_id", reminder.getId());
    cancelIntent.putExtras(extras);
    PendingIntent resultPendingIntent = 
        PendingIntent.getBroadcast(this, notificationId, cancelIntent,
PendingIntent.FLAG_CANCE\nCANCEL_CURRENT);
    Notification noti = new Notification.Builder(this)
        .setContentTitle("Reminder at this location "+
reminder.getReminderinfo())
        .setContentText("Subject")
        .setSmallIcon(R.mipmap.ic_launcher)
        .setContentIntent(pIntent)
        .addAction(android.R.drawable.ic_menu_my_calendar, "Navigate",
        pIntent)
        .addAction(android.R.drawable.ic_menu_delete, "Ignore",
resultPendingIntent)
        .build();
    NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
    noti.flags |= Notification.FLAG_AUTO_CANCEL;
    Uri alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
    if (alarmSound == null) {
        alarmSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
        if (alarmSound == null) {
            alarmSound = 
Rington\n    eManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
    }
    noti.sound = alarmSound;
    noti.defaults |= Notification.DEFAULT_VIBRATE;
    notificationManager.notify(notificationId, noti);
}
```