SMART MESSENGER FOR ANDROID

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Atharva Puranik

FALL
2015
SMART MESSENGER FOR ANDROID

A Project

by

Atharva Puranik

Approved by:

__________________________________, Committee Chair
Dr. Jinsong Ouyang

__________________________________, Second Reader
Dr. Jun Dai

__________________________________
Date
Student: Atharva Puranik

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator
Dr. Jinsong Ouyang

Date

Department of Computer Science
Abstract

of

SMART MESSENGER FOR ANDROID

by

Atharva Puranik

Mobile devices are becoming part of everyday life. The growth in mobile devices is opens an option of application development. One of the most common operating system for mobile devices is Android. Android platform provides very rich APIs to work on. With rich application for android platform, most of those are dependent on internet to communicate, share, learn and even for entertainment. It is estimated that in the next 2 or 3 years more than 80% of the companies around the world will become internet dependent, which will cause a huge demand for App developer in this market. As the World Wide Web grows Chat apps needs to upgrade proportionally. Therefore, here is the upgraded, advanced chat messenger with amazing features and easy to use for the users. This is not only a one to one chat app, it allows users to create a group chat and have features like shared list, creation of poll and creation of event for the group members.

_______________________, Committee Chair
Dr. Jinsong Ouyang

_______________________
Date
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Jinsong Ouyang, for his time, suggestions and providing me an opportunity to work on challenging and interesting project, which significantly broadened my knowledge on Android Development. I thank him for continuously providing the feedback, help and support to complete the project successfully. In addition, I would like to thank Dr. Jun Dai for his willingness to serve on the committee. I would also like to thank the entire faculty and staff of the Department of Computer Science Engineering at California State University, Sacramento. Lastly, I would like to thank my parents for their love and support throughout my education.
# TABLE OF CONTENTS

| Acknowledgements | .......................................................... | v |
| List of Figures  | .................................................................. | ix |
| Chapter          | .................................................................. |   |
| 1. INTRODUCTION   | .................................................................. | 1 |
| 2. PROJECT REQUIREMENTS | ................................................................ | 3 |
| 2.1 User Options | ................................................................ | 3 |
| 2.2 Group Chat   | .................................................................. | 4 |
| 2.2.1 Shared List | ................................................................ | 5 |
| 2.2.2 Creation of Poll Option | ......................................................... | 6 |
| 2.2.3 Creation of Event Option | ......................................................... | 7 |
| 3. ANDROID DEVELOPMENT BASICS | .................................................. | 9 |
| 3.1 Various Android Application Components | .................................................. | 10 |
| 3.1.1 Activities | ................................................................ | 10 |
| 3.1.2 Intents    | .................................................................. | 14 |
| 3.1.3 Services  | .................................................................. | 14 |
| 3.1.4 Content Providers | ....................................................... | 16 |
| 3.1.5 Broadcast Receivers | ...................................................... | 18 |
| 3.2 Setup Android Environment | .................................................. | 18 |
| 3.3 Creating Sample Android Application | ........................................ | 19 |
3.4 Running the Application ................................................................. 23
  3.4.1 On Device .................................................................................. 23
  3.4.2 On Emulator ............................................................................. 23
4. FEATURES IMPLEMENTATION .............................................................. 26
  4.1 Application Front Page / Login Page ............................................. 26
  4.2 Registration Page .......................................................................... 30
  4.3 Contacts / Group Page .................................................................... 34
  4.4 Manage Group Page ....................................................................... 37
  4.5 Refresh / Logout Buttons ............................................................... 38
  4.6 Chat Page ...................................................................................... 39
  4.7 Group Chat Page ............................................................................ 44
  4.8 Create Shared List Page ................................................................. 46
    4.8.1 Add List Prompt ....................................................................... 55
    4.8.2 Task Completed Feature .......................................................... 59
  4.9 Create Poll Page ............................................................................ 65
    4.9.1 Response Prompt ..................................................................... 69
    4.9.2 Statistics Prompt ..................................................................... 73
  4.10 Create Event Page ......................................................................... 74
    4.10.1 RSVP Prompt ......................................................................... 80
    4.10.2 Show Location ....................................................................... 81
  4.11 Future Work .................................................................................. 84
5. CONCLUSION ......................................................................................... 85

References ............................................................................................. 86
<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Use case diagram of Users options</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Use case diagram of Shared List</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Use case diagram of Poll Option</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Use case diagram of Event Option</td>
<td>7</td>
</tr>
<tr>
<td>3.1 Life Cycle of Activity</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Life cycle of Service</td>
<td>16</td>
</tr>
<tr>
<td>3.3 Content Provider</td>
<td>17</td>
</tr>
<tr>
<td>3.4 Add Repository for Android Environment setup</td>
<td>19</td>
</tr>
<tr>
<td>3.5 New Android Application Window</td>
<td>20</td>
</tr>
<tr>
<td>3.6 Configure Project</td>
<td>21</td>
</tr>
<tr>
<td>3.7 Configure the attributes of Icon Set</td>
<td>22</td>
</tr>
<tr>
<td>3.8 AVD Manager</td>
<td>24</td>
</tr>
<tr>
<td>3.9 AVD Configuration Settings</td>
<td>25</td>
</tr>
<tr>
<td>4.1 Login Page</td>
<td>26</td>
</tr>
<tr>
<td>4.2 Registration Page</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Front Page - Contacts &amp; Groups</td>
<td>35</td>
</tr>
<tr>
<td>4.4 User Information</td>
<td>36</td>
</tr>
<tr>
<td>4.5 Menu Options on Front Page</td>
<td>37</td>
</tr>
<tr>
<td>4.6 Add - Delete Groups</td>
<td>38</td>
</tr>
</tbody>
</table>
4.25 Event RSVPs .................................................................81
4.26 Direction to event location ..................................................82
Chapter 1

INTRODUCTION

Smart messenger application is an instant messaging app, which focuses on personal as well as group interaction. Smart messenger can send multimedia files like images, videos quickly. It is one of the best way to send text, images with everyone.

Smart Messenger app has few outstanding features associated with the group messaging. This app overcomes very common issues that are with other messaging apps in the market. First of three such feature is to have a shared list within the group. The most common problem in-group messaging is the proper arrangement of to-do list. My smart messenger app provides users to create not only one but also multiple shared list within the group, which can also be modified and deleted. Once the task is done present in the shared list, user can go ahead and mark it as completed to avoid confusion between other users. Second of the three features is to have ability to create a poll system within the group. One of the common problem in-group chats is to have proper stats for the poll. My smart messenger app overcomes that issue, it will allow members in the group to create a poll with questions and few options associated with it, allow other members to response to that poll, and check the stats for it. Last of the three features associated with group chat is to create an events for the group members. Any member in the group can create events for the special occasion by mentioning the location, date and time and few other notes, if required and other
members in the group can RSVP for that particular event. This feature will help the host of the event to make proper arrangements for the members who have RSVP as YES.

When it comes to normal chat apps, we might not think about all these advanced features. However, everyone faces these common issues so to overcome all common pitfalls “Smart Messenger” is created.
Chapter 2

PROJECT REQUIREMENTS

2.1 User Options

![Use case diagram of Users options](image)

Users of Smart Messenger app have following options as shown in figure 2.1

- Users have two options to do chat either user can involve in one on one chat or they can do chat within the group members.
- In personal chat i.e. one on one chat with other user, they can send text messages or they can share images or other types of documents like *.doc, *.pdf etc. Also in personal chat page, we can see the information of that user with whom they are chatting.
- In-group chat, users have various options other than what is present in one on one chat page. Those other options includes checking the group information like who
all members are there in the group and who the admin of this group is. Other than that user have an option to create a shared list for the group, also they can create a poll and event for the group.

Following section describes the detailed information and flow of important features available in Smart Messenger App.

2.2 Group Chat

On the front page of the app, there should be a list of groups in which the particular user is associated with. Once the user clicks on the group name it should go to another activity, which will be a chat page where all the members associated with the group, can have chat with each other. This page will also display all the member’s name in the group and it will specify the admin for that group with “A” as notation associated with the admin name. There will be a textbox and send button where members can type the message and on clicking send button, the message will go to all members of that group on their group chat page. There will be various features associated with the group chat like shared list, poll creation and event creation. These features are not currently available in traditional chat messenger. Following is the table which shows the comparison between the traditional messenger and my app i.e. Smart Messenger.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Smart Messenger</th>
<th>Traditional Messenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation of Shared list within the group chat</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Creation of Poll within the group chat</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Creation of events within the group chat</td>
<td>✔</td>
<td>✗</td>
</tr>
</tbody>
</table>

Features that are not available in traditional chat messenger but will be there in my app, we will see how those features will be implemented.

### 2.2.1 Shared List

![Figure 2.2: Use case diagram of Shared List](image)

Shared list is one of the feature associated with group chat. On the group chat page, there are various menu options, and one of the option is the shared list. When the user will click on manage shared list menu option, we will be redirected to another activity, which will manage the shared list feature. Manage shared list page will have list which are created by
members of the group. Moreover, this page will allow user to create a new list, which will be displayed as a list on the same page. List on this page will be the title of the list for example, grocery. Once we click, one the list title prompt will come out where user will be able to add the actual list within that title. There will be a list view, textbox and button on this prompt. After adding few items in the list, all the items will be associated with the string “yet to be completed”. For example, milk: yet to be completed. Therefore, every member on this group will be able to see this list and if some user decides to complete one of the task then he will option to mark that task as completed. Because of this feature, other members will come to know whether that task is completed or not. In addition, they will be able to see who completed that task.

### 2.2.2 Creation of Poll Option

Creating poll within the group will be another feature associated with the group chat. When we click on the mange poll option it will take user to another activity where we will be able to create new poll and will also able to see the list of other polls. When we will create a
new poll by specifying the question and the options associated with it, that poll will be listed on the same page. When we click on that poll, we will be able to see the prompt with the poll question and the options associated with it. User will have to select an appropriate option and in the backend, the result will be saved and will be displayed to the user.

2.3.3 Creation of Event Option

Another option associated with group chat will be creation of events for the group members. When user will click on manage events option, the user will be redirected to another activity where they will be able to create new event and will be able to see the list of other events. When user click on some particular event, they will see prompt saying, “Do you want to attend this event?” If the user selects “YES”, another prompt will be displayed which will have RSVP result. If the user selects “NO” then that user will stay on the same page. There will be another features associated with the event feature, i.e. when user will long press the event list, user will be redirected to another activity which will display maps with source and destination address. In the source address, users current address will be displayed and in the destination address the address associated with that

Figure 2.4: Use Case diagram of Event Option
event will get copied and when the user will click on direction button, that user will be directed to actual google maps which will give direction between current user location and the event location, this feature will help user to reach the event location easily.
Chapter 3
ANDROID DEVELOPMENT BASICS

Android is a Linux-based world’s most popular operating system for devices like mobile and tablets in much the same way that PCs run Microsoft Windows as their operating system. It is one of the mobile device operating system that is most widely used. Its market share is rapidly growing because dozens of mobile devices are being released or set to be released in less time. Android operating system allows developers from all over the world to write various applications using Java language with the help of Google Developed Java libraries. It has an open source software stack for mobile devices that includes an operating system and applications based on Linux and Java. There is an application store named “Google Play” which has various kind of applications developers have written. Google play has around 450,000 applications available that has been installed by users over 12 billion times. Those applications in the google play are arranged in various categories like game, finances, entertainment etc.

Smart messenger application is written in Java language with the help of Android SDK tool. Android SDK is integrated into eclipse IDE. The .apk file is the actual compiled file that is used to install the application on any android operating system based mobile devices.
3.1 Various Android Application Components

An android application has basic types of components: activities, intents, services, content providers and broadcast receivers. Following sections will discuss each of these components in detail.

3.1.1 Activities

Like most of the other languages main() function is where the program starts, likewise in android program it starts with Activity. It is the first android component we will encounter when we open an Android application. An Android app must have atleast one activity in it. Each screen in an android app represents an activity [1]. Every activity in an application must have user interface, it is impossible to have an activity in an app without having user interface associated with it. Every app can have one or more activities; one of the activity will act as an entry point to the app, which is also called as main activity. For example, in smart messenger application there are lot of activities like login page, registration page, contacts/groups page, chat page etc., but the entry point of the app rather the main activity of the app is the login page. Every activity in an android app must be defined in manifest.xml file to run the app without crashing and we have to specify main activity in this file. Intent filters can also be specified in the <activity> element to declare how other application components may activate it. The activity automatically includes an intent filter.
that declares the activity responds to the "main" action and should be placed in the "launcher" category. The intent filter looks like this:

```xml
<activity android:name=".ExampleActivity" android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
    </intent-filter>
</activity>
```

Furthermore, each activity in an android app has its own life cycle that are associated with various functions at different point of activities life cycle. Below we will discuss each of the functions in detail.

**onCreate**

onCreate() function is called when the activity is getting created for the first time. This function is called only once in the entire activity lifecycle. It can also be used to initialize the variables. Whenever you create an activity in an android app, the minimum function we have to override is onCreate().

```java
class MainActivity extends Activity
@override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

**onStart**
onStart() function is called just before the Activity becomes visible to the user. This function is called after onCreate function and onRestart function and is always followed by onResume or onStop.

Figure 3.1: Life Cycle of Activity [2]
• **onResume()** function is called when the Activity becomes visible to the user. At this point of time, the current activity will be on the top of the activity stack and the user can start accessing or interacting with the activity also, this function will help to refresh the user interface with the new change that might have occurred when the current activity was in the background.

• **onPause()** function is called when another activity comes on the top of the current activity. This function always make sure that the current activity will become invisible also release the resources for that particular time.

• **onStop()** function is called when the Activity is no longer visible to the user, it is similar to onPause but here user will not see your android activity entirely. We can use this method as well to store the state of your application and shut down time intensive or CPU intensive operations. This method is guaranteed to be called as of API level 11.

• **onRestart()** function is called after the activity has been stopped, just before it being started again. This is always followed by onStart().

• **onDestroy()** function is called before the activity is destroyed. This is followed by nothing.
3.1.2 Intents

Intents are used to move from one activity to another on a button click. Using intents, we can pass data and open new application. Apart from launching an activity, intent can also launch a service.

3.1.3 Services

Services is the android way of keeping the operation going on in the background without the user interaction and it also works even if application is destroyed. When you need to have long running tasks like playing music, loading an image or a file also uploading photos; it is achieved through Service [2]. It has two states started and bound. Following is the detail description of started and bound states.

1. Started – The service is started when the activity starts it by calling startService() function and it will run the service in the background for indefinite amount of time.

2. Bound – The service is bound when an application component binds to it by calling bindService() function. A bound service provides a client-server interface that allows components to interact with the service, send requests and get results. A bound service runs only as long as another application component is bound to it. Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.
The following figure 3.2 shows the life cycle of services in android, the left part shows the life cycle when the service is created with startService() and on the right part it shows the life cycle when the service is created with bindService(). The callback methods are:

- **onStartCommand()** function is called when an activity requests for a service to be started by calling startService().

- **onBind()** function is called when an activity wants to bind with the service by calling bindService().

- **onUnbind()** function is called when all clients have disconnected from a particular interface published by the service.

- **onRebind()** function is called when new clients have connected to the service.

- **onCreate()** function is called when the service is first created to perform setup of the service before onStartCommand or onBind is called.

- **onDestroy()** function is called when service is no longer required and is to be destroyed.
3.1.4 Content Providers

A content provider component supplies data from one application to others on request. The data is usually stored in file system or SQLite database and the data can be accessed by the application through content provider, other applications can query or modify its content depending on the settings of the content provider [2]. A content provider is implemented
as a subclass of ContentProvider and an application accesses the data from a content provider with a ContentResolver client object. Figure 3.3 shows the methods that need to override in Content Provider class to have your Content Provider working.

Figure 3.3: Content Provider

- onCreate() method is called when the provider is started.
- Insert() method inserts a new record into the content provider.
- Update() method updates an existing record from the content provider.
- Delete() method deletes the existing record from content provider.
- Query() method receives request from a client and the result is returned as a cursor object.
3.1.5 Broadcast Receivers

A broadcast receiver allows registering for system or application events. An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file. When any registered event occurs, receivers for an event will be notified by the Android runtime. Some of the system broadcast include the battery is low/charged, or a picture was captured and some of the applications broadcast would include letting other applications know that some data has been downloaded to the device and is available for them to use. A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as a Intent object parameter.

3.2 Setup Android Environment

For creating, any android app is to setup a development environment. The first step would be to download the Android Development Tools (ADT) plugin to use it with Eclipse IDE. By default, Eclipse is not good suited for Android application development. ADT is a plugin that configures the environment to use Android SDK and enables convenient development. Launch Eclipse IDE and go to Help -> Install New Software, install the ADT plugin from here by entering “https://dl-ssl.google.com/android/eclipse/” address.
After this select the developer’s tools option and click on finish button. Once installing ADT plugin finishes, launch the SDK manager and select the appropriate android API, android SDK tools option and google services options for installation. Once all these installations are done, we are ready to create an android app.

3.3 Creating Sample Android Application

Following is the way we can create any kind of android application in Eclipse IDE with Android SDK. Eclipse IDE models each application as a project, with the coding and other resources related to the app stored and managed from within a directory for the project in your workspace [4].

Creating a new Android project involves a few steps; first, we have to click on “new” in the toolbar then in the wizard, expand the "Android" folder and select "Android Application Project". After that, “New Android Application” screen will appear where we will see various fields like Application Name, Project Name, Package Name, Minimum Required SDK, Target SDK etc. Entering the application name will automatically populate
project name and package name fields, this can be altered easily. The lowest version that this app supports is the minimum required SDK. If we want to support multiple devices, this needs to be set to lowest version available such that the application can perform its basic operations. The target SDK is the highest version of android for this application. The platform version with which we compile this sample app is specified in the Compile With. The theme specifies the Android UI style to apply for your app. Figure 3.5 shows the first screen comes up when we select to create a new android application.

![New Android Application Window](image)

**Figure 3.5: New Android Application Window**
On clicking next button, the configure project screen will appear which will have various options for configuration of the project like selecting workspace for the project, creating the project as a library and so on. The screen appears as figure 3.6.

![Configure Project Screen](image)

**Figure 3.6: Configure Project**

On clicking next, the next screen will create a launcher icon for this app as shown below.
Figure 3.7: Configure the attributes of Icon Set

On clicking next, there are several of options to select the activities, for this sample project we have selected Blank Activity and clicked on next. The other details are left at default and click on finish. All the above steps will help us to create a new android application.
3.4 Running the Application

The android application can be run either on the actual device or on the emulator [5]. We need to setup both device and emulator to run android application on it. The details about how we can setup and run any android application on them will be described below.

3.4.1 On Device

To run any android application on device we have to go through few steps. First, we have to connect out device to the laptop via USB cable. On connecting the device, windows will automatically install the drivers for USB connection. Once the device is ready to use with the laptop, we have to go to setting of device and go to the developer’s options where we have to enable the option of USB debugging. If this option is not enabled on the device, we cannot run and debug our application on phone. After that from Eclipse IDE click on the “run” button then need to select “Android Application” and click ok. After that select any one of the devices that are connected to the laptop and this will cause Eclipse to install the app on the selected/connected device and start it.

3.4.2 On Emulator

When the android application needs to be run on emulator, then first step is to set the Android Virtual Device (AVD). The AVD is a device configuration for the Android emulator that allows modeling different devices with different target API, CPU/ABI,
memory option like RAM, VM Heap and few other options like internal storage and SD card options.

In order to create AVD, we have to launch AVD manager, which looks like figure 3.8.

![AVD Manager](image)

**Figure 3.8: AVD Manager**

Once we open the AVD manager, the first step is to click on create button that will open an AVD settings window that will have various options to fill out for the AVD. We can have AVD according to our requirement. For example, the name of the AVD can be “Smart Messenger AVD” with device configuration similar to Nexus 4 that has 4.7” of screen size. Our targeted device is with API Support of level 21. We have RAM capacity of 1024 MB and an internal SD card of 200MB. Setting internal card is optional, but it is need to install
Google Play Service, which is essential for this Application. Figure 3.9 shows AVD configuration settings window.

![AVD Configuration Settings](image)

**Figure 3.9: AVD Configuration Settings**

Click OK. The new AVD will appear in the Android Virtual Device Manager screen. Select the AVD and click on Start. Wait for the emulator to boot up and then unlock the emulator screen. In order to run the app from Eclipse, click Run from the toolbar. Then select Android Application and click OK. Eclipse will install the app on the AVD and once we unlock the emulator screen app will start.
Chapter 4
FEATURES IMPLEMENTATION

4.1 Application Front Page / Login Page

Almost every android application will have login and registration process in order to authenticate a user. The same case is with Smart Messenger application. It has Login page where user will have to type their appropriate username and password to enter into the system. Figure 4.1 shows the login page of the app, which is also a front page.

Figure 4.1: Login Page
This is the main activity in the app, so whenever you will open this app you will get to see the above screen as a front page. For each screen on Smart Messenger app there is an associated activity. Each activity is associated with XML documents, which specifies the overall screen layout i.e. how the screen is going to look like. In this app, main activity is called as login page, which is associated with activity_main.xml file. This .xml file defines various categories like having an Editbox, Textbox, ImageView and buttons. Let us look at the code related to this page and little explanation over it. The following code shows that it is creating an activity using onCreate() method and we are setting the contentView layout as activity_main.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

Same user should not be able to use multiple username from the same device, to implement IMEI number is used which is unique for every device. Therefore, what happens is if you have registered yourself using particular device it will save the IMEI number of that device which will not allow the same user to register again from the same device. Following code snippet shows how that has been done.

```java
// Get the IMEI Number
MY_IMEI = ((TelephonyManager)
getSystemService(Context.TELEPHONY_SERVICE)).getDeviceId();
```
Then on this front login page, we are using two EditBox & two Buttons, which are mentioned in the code in the following way.

```java
// Get the controls
final EditText txtUserName = (EditText) findViewById(R.id.txtLoginUserName);
txtPassword = (EditText) findViewById(R.id.txtLoginPassword);
final Button btnLogin = (Button) findViewById(R.id.btnLogin);
final Button btnSignup = (Button) findViewById(R.id.btnSignup);
```

When you click on “Login” button, it takes username and password from the user and checks if the same pair is present in the database, if it present in the database it will allow the user to go through the system otherwise it will prompt user that the login failed, try again. In the background, systems verifies various things, which include current user’s location i.e. latitude and longitude and IMEI number.

```java
btnLogin.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        AsyncTask<Context, String, Void> atask = new AsyncTask<Context, String, Void>() {
            @Override
            protected Void doInBackground(Context... params) {
                ArrayList<BasicNameValuePair> lstNVP = new ArrayList<BasicNameValuePair>();
                BasicNameValuePair nvp = new BasicNameValuePair("username", txtUserName.getText().toString());
                lstNVP.add(nvp);
                nvp = new BasicNameValuePair("password", txtPassword.getText().toString());
                lstNVP.add(nvp);
                // Rest of the code...
            }
        }
    }

    @Override
    protected void doInBackground(Context... params) {
        // Rest of the code...
    }
});
```
For the better-looking UI of the screen all the changes are made in .XML code. For the front page all, the changes are made in activity_main.xml file. Few of the code snippet and its explanation are as follows.

In the Login page, we have used Relative layout, which is designed to display child View controls in relation to each other. For instance, we can set a control to be positioned “above” or “below” or “to the left of” or “to the right of” another control, referred to by its unique identifier. In addition, we have used ScrollView that surrounds the entire layout so that the entire screen is scrollable. Moreover, there is EditBox where user are supposed to write their username and password and there are buttons, which on click will switch the user from one activity to the other. In addition, there is an ImageView right above the username Editbox. Following is the code snippet of activity_main.xml.
4.2 Registration Page

Smart Messenger has another important page, which should be present in every android application, without registration page you cannot get register in the system that eventually means user cannot login into the system. Therefore, to get register in the system application
asks various basic information to store it in its database. The information required to get register into the system are unique username, which mean that there must not be any other user who has same username, password, first name of the user, last name of the user and phone number. Figure 4.2 shows the registration page screen.

![Registration Page](image.png)

**Figure 4.2: Registration Page**

As we can see in figure 4.2., various fields needs to be filled by the user to be registered in the system.

```java
btnSignup.setOnClickListener(new OnClickListener(){
    @Override
```
public void onClick(View v)
{
AsyncTask<Context, String, Void> atask = new AsyncTask<Context, String, Void>()
{
@Override
protected Void doInBackground(Context... params)
{
    ArrayList<BasicNameValuePair> lstNVP = new ArrayList<BasicNameValuePair>();
    BasicNameValuePair nvp = new BasicNameValuePair("username", txtUserName.getText().toString());
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("password", txtPassword.getText().toString());
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("latitude", my_latitude + "");
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("longitude", my_longitude + "");
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("imei", MY_IMEI);
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("first_name", txtFirstName.getText().toString());
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("last_name", txtLastName.getText().toString());
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("contact", txtContact.getText().toString());
    lstNVP.add(nvp);
    nvp = new BasicNameValuePair("operation", "signup");
    lstNVP.add(nvp);

    String resp = WebFunctions.postHttpData(
        WebFunctions.URL_SERVER_OPERATIONS, lstNVP);

    publishProgress(resp);
    return null;
}
@Override
protected void onProgressUpdate(String... values)
{
    super.onProgressUpdate(values);
    try
    {
        int resp = Integer.parseInt(values[0]);
        if (resp >= 1)
In the above Java code snippet for registration page, we are taking input from the user and comparing those string values in the web function. The field username checks in the database for its uniqueness if the same username is not present, it will allow user to register in the system and will show toast as signup successful. If some exception occurs during the registration, it will show toast as unable to sign up.
4.3 Contacts / Group Page

In Smart Messenger, application there is a page, which shows all the users that are connected with the current logged in user and the associated groups. In addition, there is an option to check the information about yourself like what is the first and last name of the current user, and then when that user got register into this messenger system, what is the location of the user in terms of latitude and longitude with its IMEI number and phone number. All this information was written by the user himself while registration. When the user information prompt comes up with all the above information, it also ask if the user wants to hide his location or show his location to the other users. Two figure 4.3 & 4.4 shows the contact / group page as well as the user information prompt.
In the above screenshot, we can see on the left side of the screen it shows all the users connected with the current user and on the right side, it shows all the groups in which the current user is involved. It also has the information button that will give user a prompt for user information.
Figure 4.4 shows the user information. On contacts / group page on the right – top corner there is a menu button on clicking that various options will come out like manage groups, refresh and logout options that will be discussed in next few sections.
4.4 Manage Group Page

In manage group’s page of smart messenger application we can add or delete groups. Firstly, to create a new group we have to write the group name in the EditBox and select the contacts from the contacts list, which the current user wants into that particular group, and click on, add button, which will create a new group as well as the current user who has created this group, will be the group admin. Secondly, to delete the group which you have admin rights can be deleted here. Just write the name of the group you want to delete it and
click in delete group button that will delete that particular group for you. Figure 4.6 shows where we can add or delete the groups.

![Figure 4.6: Add - Delete Groups](image)

**4.5 Refresh & Logout Buttons**

Refresh button is mostly used to refresh the app page when some sort of data-retrieval from a server is required. When you click on this option, it will update the data from the server. Whenever user wants to logout from the app, they can click on this option.
4.6 Chat Page

Chat page is nothing but the traditional instant messaging system, which allows users to communicate with each other. Text or any shared file is instantly displayed in the chat page after the user hits send button. Other user included in the chat session will be able to see the message. This page is just for private chat between two users. In this page user will be able to see all the details of the user communicating with like first name, last name, registered since, location, device id and phone number. As specified in the contacts / group page if the users specifies not to allow the user to show current location then they can hide it and will not displayed to the other user. Here in the figure 4.7 we can see the tradition chat page where two people are communicating with each other and the information of the user with whom the current user is communicating with. In addition, we can see that the value of the location here in the figure is 0, 0, that means the user has decided not to show his current location.
Also, as we can see in the above figure there is an EditBox as well as send button where user writes the message and send using send button. Whenever the user will click on the EditBox on this page, a keyboard will come out. Figure 4.8 shows the same.
When the user type and click on send button the prompt will come out which will ask user if they wish to attach any file from the system. If the user selects “No” then the prompt will go away and only the text in the Edit Box will be sent to the other user but if the user select “Yes” option on the prompt then the user will be redirected to the other activity which will be the file system of your device where you can select any kind of file and attach it with the text message. Now in this case both, text and the file user has selected will be sent to the other user. Two figures 4.9 & 4.10 will show prompt to attach the file with the message and the file system it will go through if selected “Yes” in the prompt.
Figure 4.9: Prompt to attach files
Once the file is selected from the file system screen like figure 4.11 will show up the progress bar that shows it is sending message.
4.7 Group Chat Page

There are many features in the group page, which are similar to the chat page. However, there are many things, which are not in the chat page. The first thing which is very obvious as the name suggests it’s not just a one on one chat page it’s a group chat page which means here all the members in a particular group can chat with every other member in the group. Other thing which is different here from the chat page is, in the chat page we can see the information of the user with whom we are chatting with, but here in the group chat page at the same location we can see the members of the group also the admin of the group is specified as (A) with their name. That means the user name in the members list with (A)
associated with it is the one who has created this group. The group name can be seen on the action bar. Following is the figure 4.12, which shows the group chat page screen.

As we can see in figure 4.12, the name of the group is “Friends” and “atharva” is the admin of this group. In addition to that, we can see all three members of the group are chatting with each other. Also we can see the three dots on the top - right corner of the page if we
select that option, it will shows us various options we can do within the group. Figure 4.13 will show what all options the group chat provides.

![Figure 4.13: Menu options for group chat](image)

As the figure 4.13 shows, we can have shared list, polls and events within the group. All these features for the group will be explained in following sections.

### 4.8 Create Shared List Page

Organize and share your to-do, work, grocery, movies and household lists. No matter what you are planning, how big or small the task may be. Creating a shared list is the option
given in-group chat only. Within the group chat page, we can have shared list where anyone within group can create a list or delete a list that will be sync with other members in the group. Once the task is finished, the user who finished that task can check it as task completed. Having shared list within the group has many advantages like it help every member in the group to remember what task they have to do so that they do not miss anything. It will help to avoid repetition of labor. For example, if there is a shared list of grocery and someone in the group is already doing shopping for that then that person has to mask that task as completed so that other members in the group will come to know that this task is already completed and they will not do the same task. It helps enables you to mark off the tasks you have completed. At the end of the day, when you look at the list, it will give you a sense of accomplishment and satisfaction. It might also have the effect of waking you up if nothing has been marked completed. Figure 4.14, shows the name of shared list.
As we can see in figure 4.14 various list names are listed and on which date those lists are created is mentioned. In addition to that, we can rename and delete the list. When we click on the list it will show us the options that is shown in figure 4.14 (a), rename it or delete it. If user selects an delete option the list will get deleted from the database and if we select the rename it option then it will pop up prompt as figure 4.14 (b).
The above java code snippet shows how the shared list is added. Here, when we click on the add button after entering the list name it will store the user id of the user who created
that list and group id, to specify the group for which the list is created. All the information will be stored in the database and then using web function the list will be fetched.

Figure 4.14 (a): Rename and delete option for the list
Figure 4.14 (b) shows the option after selecting the rename option. User can rewrite the name of the list and click on the change name button, which will eventually change the name of the list. Cancel button will go back to the previous screen.

```java
lstSharedLists.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    }

    public boolean onItemLongClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        // Long click listener logic
    }
```
list_id = lstListIds.get(arg2);
AlertDialog.Builder abuild =
    new AlertDialog.Builder(
        ManageLists.this);
abuild.setTitle("Select an option");
abuild.setPositiveButton("Delete it!", new
    DialogInterface.OnClickListener()
    {
        @Override
        public void onClick(DialogInterface dialog,
            int which)
        {
            // TODO Auto-generated method stub
        }
    });
AsyncTask<String, String, Void> ataskDelete = new
AsyncTask<String, String, Void>()
    {
        @Override
        protected Void doInBackground(String... params)
        {
            List<BasicNameValuePair> lstNvp = new
                ArrayList<BasicNameValuePair>();
lstNvp.add(new BasicNameValuePair("list_id",
                list_id + "");
lstNvp.add(new BasicNameValuePair("operation",
                "delete_list");
WebFunctions.postHttpData(WebFunctions.
    URL_SERVER_OPERATIONS, lstNvp);
publishProgress("");
        return null;
    }
    @Override
    protected void onProgressUpdate(
        String... values)
    {

// TODO Auto-generated method stub
super.onProgressUpdate(values);
finish();
};
ataskDelete.execute("");
);
abuild.setNegativeButton("Rename it", new
DialogInterface.OnClickListener()
{
@Override
public void onClick(DialogInterface dialog,
int which)
{
final Dialog dlg = new Dialog(ManageLists.this);
dlg.setContentView(R.layout.change_name);
dlg.setTitle("Enter new name for list");
final EditText txtNewName = (EditText)dlg
.findViewById(R.id.txtNewName);
final Button btnChangeName = (Button)dlg
.findViewById(R.id.btnChangeName);
final Button btnCancel = (Button)dlg
.findViewById(R.id.btnCancel);
btnChangeName.setOnClickListener(new OnClickListener()
{
@Override
public void onClick(View v)
{
AsyncTask<String, String, Void> atask = new
AsyncTask<String, String, Void>()
{
@Override
protected Void doInBackground(String... params)
{
List<BasicNameValuePair> lstNvp = new
ArrayList<BasicNameValuePair>();
lstNvp.add(new BasicNameValuePair("list_id",
list_id + "");
The above code snippet shows how user can delete and rename the list title. Whenever user will long press the list title the options will come up like shown in figure 4.14 (a). User will have option either to delete the list or to rename the list. If the user selects to delete the list, it will select the particular list id from the database and will be deleted from there. If the user selects to rename the list, it will pop up the prompt just like figure 4.14 (b) which will ask either to rewrite the name if the list and click on change name button or the user can cancel the current task. All these functionality is implemented in the above code.
4.8.1 Add List Prompt

As we have seen in the earlier section within the group shared list page there will be a list of names of the task for example, shopping, grocery, work in university etc. All these names are the generalized one, which will give user an idea of what things should be inside that particular list name. Add list prompt will appear when we click on one of the list name and it will allow you to add a data/list within that list name. For example, grocery is the name of the list and when we click on that list name, it will pop up the Add List Prompt and will allow user to add data inside that list like milk, veggies, cookies etc. Figure 4.15 shows how the Add List Prompt look like.
When the data is added in the list, it will start appearing in that prompt in the form of list and it will specify that this task is yet to be completed. Figure 4.16 shows how it looks like.
Figure 4.16: Sub-List with details of completion

```java
lstSharedLists.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
    {
        list_id = lstListIds.get(arg2);
        // Show the tasks in the list, and give an option to add more tasks
        Dialog dlg = new Dialog(ManageLists.this);
        dlg.setContentView(R.layout.list_data);
    }

    public void onItem颇为(AdapteView arg0, View arg1, int arg2, long arg3)
    {
    list_id = lstListIds.get(arg2);
        // Show the tasks in the list, and give an option to add more tasks
        Dialog dlg = new Dialog(ManageLists.this);
        dlg.setContentView(R.layout.list_data);
```
final EditText txtListDataName = (EditText)dlg.findViewById(R.id.txtListDataName);
final Button btnAddData = (Button) dlg.findViewById(R.id.btnAddListData);
lstData = (ListView) dlg.findViewById(R.id.lstSharedListData);
refreshListData(list_id, lstData);
btnAddData.setOnClickListener(new OnClickListener(){
    @Override
    public void onClick(View v)
    {
        final String msg = txtListDataName.getText().toString();
        txtListDataName.setText("");
        if (msg.trim().equals(""))
            return;
        AsyncTask<String, String, Void> atask =
            new AsyncTask<String, String, Void>(){
                @Override
                protected Void doInBackground(String... params)
                {
                    ArrayList<BasicNameValuePair> lstNVP = new ArrayList<BasicNameValuePair>();
                    lstNVP.add(new BasicNameValuePair("list_id", list_id + ""));
                    lstNVP.add(new BasicNameValuePair("user_id", user_id + ""));
                    lstNVP.add(new BasicNameValuePair("msg", msg));
                    lstNVP.add(new BasicNameValuePair("operation", "add_list_data"));
                    String resp = WebFunctions.postHttpData(WebFunctions.URL_SERVER_OPERATIONS, lstNVP);
                    publishProgress(resp);
                    return null;
                }
                @Override
                protected void onPostExecute(String... values)
                {
                    // TODO Auto-generated method stub
                    super.onPostExecute(values);
                }
            };
    }
};
The above java code shows when we click on the list title it will pop up another prompt just like figure 4.15 and figure 4.16 where user will be able to add the sub list within the list title. On “Add Shared List Data” button click, the list data will be stored in the database according to the list id, user id. On progress, the list will get refreshed according to list id and list data.

### 4.8.2 Task Completed Feature

Task completed Feature is nothing but it allows user to check or mark the task as completed. When user creates any list and within that list users add data about the task, in front of the task it will be written that the task is yet to be completed until one of the member does that task and mask it as a completed. So, when the user clicks on any list data it will pop up the prompt that will say do you want to mark this as completed? If user click on NO the prompt will go away and the list will still show that, it is yet to be completed. However, if user click on YES then user will have option to upload the image of the task completed with the list item and once image is uploaded, it will go back to previous screen,
which will show that the task is completed. In addition, it will show who completed that
task and on which date as can be seen in figure 4.17.

Figure 4.17: Prompt to mark the completion of the task

As we can see in figure 4.17, it gives option to click an image of completion of task. Figure
4.18 shows that who completed the task and on which date. As shown below it shows that
the task cake is completed by Atharva, task Board Game is completed by Aditya and so
on.
As we can see in figure 4.18 (a), the person who completes the task is given an option to click an image of that particular thing and once any user selects the completed task they will be able to see the image of that particular item. Below figure shows the image of completed task of buying balloons by Atharva on 12/11/15.
lstData.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3)
    {
        String task_info[] = lstListDataIds.get(arg2).split(",");
        list_data_id = Integer.parseInt(task_info[0]);
        final int done = Integer.parseInt(task_info[1]);
        if (done == 1)
// Fetch the image of the item
AsyncTask<String, String, Void> atask = new
AsyncTask<String, String, Void>()
{
protected void onPreExecute()
{
super.onPreExecute();
Toast.makeText(ManageLists.this, "Fetching completion image",
Toast.LENGTH_LONG).show();
};
@Override
protected Void doInBackground(String... params)
{
List<BasicNameValuePair> lstNvp = new
ArrayList<BasicNameValuePair>();
lstNvp.add(new BasicNameValuePair("list_data_id",
list_data_id + "");
lstNvp.add(new BasicNameValuePair("operation",
"fetch_list_image"));
String resp = WebFunctions.postHttpData(WebFunctions.
URL_SERVER_OPERATIONS,
lstNvp);
try
{
if (resp.trim().equals("") == false)
{
// Download the file
String fName = Environment.getExternalStorageDirectory()
.getAbsolutePath() + "/image.jpg";
WebFunctions.downloadFile(WebFunctions.URL_BASE+
resp, fName);
publishProgress(fName);
}
}
catch (Exception ex)
{
}
return null;
}
@Override
protected void onProgressUpdate(String... values)
{
// TODO Auto-generated method stub
super.onProgressUpdate(values);
try{
    Dialog dlg = new Dialog(ManageLists.this);
dlg.setContentView(R.layout.view_image);
final ImageView img = (ImageView)dlg.findViewById(R.id.imgTaskCompleted);
    img.setImageBitmap(BitmapFactory.decodeFile(values[0]));
    dlg.show();
} catch (Exception ex) {
    
} 
return;
}
atask.execute("");
AlertDialog.Builder abuild = new AlertDialog.Builder(ManageLists.this);
    abuild.setTitle("Do you want to mark this completed?");
    abuild.setMessage("Mark this task completed?");
    abuild.setPositiveButton("Yes (Upload Completion Image)",
    new DialogInterface.OnClickListener()
{

    @Override
public void onClick(DialogInterface dialog, int which)
{
    Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(intent,CAMERA_CODE);
The above snippet shows, when we click on the list data, it will ask user if the user wants to mark this task as completed, if the user clicks on “No” button the dialog will get cancelled and if the user selects “Yes” it will ask the user to click the picture using camera and the picture will be saved in the local folder. Once the task is completed and if the other users wants to see if the user who has completed this task has bought the correct thing or not, they will have to click on the listed task and the image will be fetched. All these operations implementation is shown in above code snippet.

4.9 Create Poll Page

Creating poll features allows any user in the group to create a poll by giving the name of the poll and the options to select. This feature is useful for the group members in such a way that if an user wants to ask other members in the group if they are interested in going for movie and few members will answer this question as yes few will say no and there is a
possibility that remaining members in the group might not have answered that question. In such a case the user who have asked the question will not have a proper statistics for that poll and the user will have to go through all the messages and count who have given what answer. Figure 4.19 shows the page for creating a poll where there is a edit box in which user can add the question and various options he wants to add. There is an Add poll button on clicking there the poll will be added into the list of various poll for that group with the date it is created on.

Figure 4.19: Creation a poll for group
Figure 4.20: List of polls

```java
btnAddPoll.setOnClickListener(new OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        final String name = txtPollName.getText().toString();
        txtPollName.setText(""");
    }
});
```
final String poll_options =
txtPollOptions.getText().toString();
txtPollOptions.setText("");  
if (name.trim().equals("") ||
poll_options.contains(",")) == false)
return;

AsyncTask<String, String, Void> atask = new
AsyncTask<String, String, Void>()
{
@Override
protected Void doInBackground(String... params)
{
ArrayList<BasicNameValuePair> lstNVP = new
ArrayList<BasicNameValuePair>();
lstNVP.add(new BasicNameValuePair("user_id", user_id + ""));
lstNVP.add(new BasicNameValuePair("group_id",
group_id + ");
lstNVP.add(new BasicNameValuePair("name", name));
lstNVP.add(new BasicNameValuePair("poll_options", poll_options));
lstNVP.add(new BasicNameValuePair("operation","create_poll"));
String resp =
WebFunctions.postHttpData(WebFunctions.URL_SERVER_OPERATIONS, lstNVP);
publishProgress(resp);
return null;
}
@Override
protected void onProgressUpdate(String... values)
{
// TODO Auto-generated method stub
super.onProgressUpdate(values);
refreshPolls();
}
};
atask.execute("");
The above java code snippet shows about creation of new poll for the group. Here group members will be able to create a poll question using the edit box and will be able to add poll options. When user will click on add poll button, it will save the question and the options in the database according to the user id, group id.

4.9.1 Response Prompt

One the poll is created, any member in the group can click on that poll from their own devices, which will give a response prompt popup with the question and the options it has. The user will have to select one of the option, which he feels appropriate and it will pop up the statistics prompt. Figure 4.21 shows the response prompt.
Figure 4.21: Poll Response Prompt

```java
public void refreshPollAnswers(int PollId, String[] PollOptions)
{
    final int poll_id = PollId;
    final String[] poll_options = PollOptions;

    AsyncTask<String, String, Void> atask = new 
    AsyncTask<String, String, Void>()
    {
```

@Override
protected Void doInBackground(String... params)
{
    ArrayList<BasicNameValuePair> lstNVP = new ArrayList<BasicNameValuePair>();
    lstNVP.add(new BasicNameValuePair("poll_id", poll_id + ""));
    lstNVP.add(new BasicNameValuePair("operation", "fetch_poll_results"));
    String resp = WebFunctions.postHttpData(WebFunctions.URL_SERVER_OPERATIONS, lstNVP);
    if (resp.contains(";;;"))
    {
        String[] results = resp.split(";;;");
        String str = ""
        for (int count = 0; count < results.length; count++)
        {
            String[] counts = results[count].split(",,");
            int option = Integer.parseInt(counts[0]);
            int option_count = Integer.parseInt(counts[1]);
            str = str + poll_options[option] + ":" + option_count + " votes\n"
        }
        publishProgress(str);
    }
    else
    {
        publishProgress(resp);
    }

    return null;
}
The above code is the refreshPollAnswer function that will take poll id and its options and using web function it will fetch the poll result and displays it to the members of the group. The poll result will be displayed with the poll option and the number of votes each options got. The prompt will look like figure 4.22 which will have title as “Poll Results”.

4.9.2 Statistics Prompt

After the user selects an appropriate options the statistics prompt will come out which will show how many members have selected which options. Figure 4.22 shows the statistics prompt.

Figure 4.22: Stats for the poll
In figure 4.22, we can see the statistics of the poll created by some user. We can easily see which option gets how many votes for that particular poll. Here we can see Sachin got 2 votes and Sehwag got 1 vote.

4.10 Create Event Page

Smart messenger app gives user one more amazing feature i.e. Creation of Events. The members of any group can create an event by giving few details like Event name, location, date and additional information about the event. When other members in the group check the event page, they will see the invite for the event and they will have to RSVP for that event. The user who created that event will be able to see the stats for that event which will help them to arrange according to the head count. Figure 4.23 shows the event page where members can create an event and give responses for the events.
Figure 4.23: Event Creation

```java
btnAddEvent.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        final String name = txtEventName.getText().toString();
        txtEventName.setText("");
        final String event_location = regLoc;
        txtEventLocation.setText("");
        final String event_date = txtEventDate.getText().toString();
    }
});
```
final String event_info =
txtEventInfo.getText().toString();
txtEventDate.setText("");
txtEventInfo.setText("");

if (name.trim().equals("") ||
    event_location.trim().equals("")
    || event_date.trim().equals("")
    || event_info.trim().equals("")) return;

AsyncTask<String, String, Void> atask = new
AsyncTask<String, String, Void>()
{
    @Override
    protected Void doInBackground(String... params)
    {
        ArrayList<BasicNameValuePair> lstNVP = new
        ArrayList<BasicNameValuePair>();
        lstNVP.add(new BasicNameValuePair("user_id", user_id+
""));
        lstNVP.add(new BasicNameValuePair("group_id", group_id+
""));
        lstNVP.add(new BasicNameValuePair("name", name));
        lstNVP.add(new
        BasicNameValuePair("event_location",event_location));
        lstNVP.add(new BasicNameValuePair("event_on",event_date));
        lstNVP.add(new
        BasicNameValuePair("other_info",event_info));
        lstNVP.add(new
        BasicNameValuePair("operation","create_event"));
        String resp = WebFunctions.postHttpData(WebFunctions.

        URL_SERVER_OPERATIONS, lstNVP);
        publishProgress(resp);

        return null;
    }

    @Override
    protected void onProgressUpdate(String... values)
    {

    }
}
The above java code snippet shows the implementation of Add Event button, where user will write all the information about the event and when clicked on add event button, all the data in the edit box will be stored in the background database, which then can be fetched using web function. On progress, it will call refreshEvents function that will make sure to refresh the database; fetch the latest list and display the latest list on the screen.

One the user have created an event, all the members in the group can see the event list. Members can click on that particular event which will pop up a prompt saying, “Do you want to attend this event? If the member responded YES that means he is going to attend that event and the host of the event i.e. the person who have created this event will be able to see the RSPV of that particular person. If the member select the option NO the prompt will go away. Figure 4.24 shows the prompt for RSVP.
Figure 4.24: Response Prompt for Event

```java
lstEvents.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1,
        int arg2,
        long arg3)
    {
        String[] val = lstEventIds.get(arg2).split(";;");
        final int event_id = Integer.parseInt(val[0]);

        AlertDialog.Builder abuild = new AlertDialog.Builder(
            ManageEvents.this);
```
setTitle("Do you want to attend this event?");

new DialogInterface.OnClickListener()
{
    @Override
    public void onClick(DialogInterface dialog, int which)
    {
        final DialogInterface dlg = dialog;

        AsyncTask<String, String, Void> atask = new
        AsyncTask<String, String, Void>()
        {
            @Override
            protected Void doInBackground(String... params)
            {
                ArrayList<BasicNameValuePair> lstNVP = new
                ArrayList<BasicNameValuePair>();
                lstNVP.add(new BasicNameValuePair("event_id",
                    event_id + ")
                lstNVP.add(new BasicNameValuePair("user_id",
                    user_id + "")
                lstNVP.add(new BasicNameValuePair("operation",
                    "add_event_rsvp"));
                String resp = WebFunctions.postHttpData
                    (WebFunctions.URL_SERVER_OPERATIONS, lstNVP);
                publishProgress(resp);
                return null;
            }

            @Override
            protected void onProgressUpdate(String... values)
            {
                // TODO Auto-generated method
                // stub
                super.onProgressUpdate(values);
                refreshEventRSVPs(event_id);
                dlg.cancel();
            }
        };
    }
The above java code snippet will list all the events within that group and once we click on one of the events listed in; there it will show us a prompt as shown in figure 4.24. If the user responds to the event as “Yes” then, it will fetch all list of users who are going to attend that event and will be displayed as list of attendees as shown in figure 4.25 and if the user says “No” then onClickListener function will get invoked and the dialog will get cancelled.

4.10.1 RSPV Prompt

Once the user did RSVP for the event, figure 4.25 will be displayed which shows all the list of attendees. As we can see for some event in the group user “Atharva”, “Purva” and “Aditya” are going to attend that event.
4.10.2 Show Location

Show location of the event is also one of the important feature associated with the creation of event feature. Here, when we long click on any particular event within that group, it will go to another activity, which will look like figure 4.26, where there will be map, two textbox and a button. The first text box will automatically take current location of the user.
and another textbox will take the location of the event. All this will be automated; user will not have to explicitly write anything in the textbox and when user click on direction button app will redirect to GoogleMaps that eventually will show the direction and time between current location of the user to the event’s location.

![Map showing event location](image)

**Figure 4.26: Direction to event location**

```java
LocationListener locListen = new LocationListener()
{
  @Override
  public void onStatusChanged(String provider, int status, Bundle extras)
  {
  }
```
The above java code snippet will get the current location of the user using Location Manager. It will fetch the user location’s latitude and longitude by listening to GPS_PROVIDER and NETWORK_PROVIDER. These location coordinates will be used to see the distance between the current location of the user and the location of the event.
4.11 Future Work

There is always a room for improvements in any application, it does not matter how good and efficient it may be done. Currently, this app has support for text communication only. In near future, this app can have various functionality like video and voice chat. This app can have emoji’s added into it. In addition, this app can support voice notes.
Chapter 5

CONCLUSION

The main objective of “Smart Messenger” is to develop a high-end chat application with various advanced features for the group chat. I have taken a wide range of literature review in order to see if the features I thought of are existing in the market or not. I made a detailed research in the path to cover the loopholes that existing systems are facing and to remove it in this application. I came up with various features like shared list, creation of polls and events in the group chat. I had implemented that functionality in my android application. All these functionalities of the application has been achieved by using Android Development Kit (ADK), WAMP server, MySQL database. I implemented these functionalities using JAVA language. In order to test this app I used emulator to run the app as well as mobile device with android operating system.
REFERENCES


