

Chatting project

Using Firebase UI and Scaledrone Java API Client

14/11/2018

Inha University in Tashkent

Name:	Maftuna Sharabbaeva	ID:	U1510067
	Umidjon Zaribov		U1510046

Abstract

In this project, User firstly register and then login to messenger. User can his/her email and password and then use it forever to log in our chat program. User can chat anyone in channel. This is android application which is using Firebase UI for registering and logging in and Scaledrone Java API Client for messaging in channel.

Table of Contents

1. Introduction.....	
1.1 Prerequisites	
2. Functions and logics.....	
2.1 Functions and logics for Registering using Firebase.....	
2.2 Functions and logics for activity to make access for legible user to messenger.....	
2.3 Functions and logics for messaging using Scaledrone Java API Client...	
3. Conclusion.....	
4. Reference.....	

Chapter 1 Introduction

In this project, we built login and registration for android using Firebase Authentication. We have added Email & Password login. Firebase automatically stores user information in the database. Here, we are also going to be building a realtime group chat for Android using the Scaledrone Java API Client. It will work very similarly to apps such as WhatsApp, Facebook Messenger and LINE. In this project we built a fully functional group chat. We designed the UI elements such as chat bubbles and text inputs. We used Scaledrone as the realtime backend of your app. The project might seem daunting at first, but the messaging code outside of the layout files is pretty short.

1.2 Prerequisites

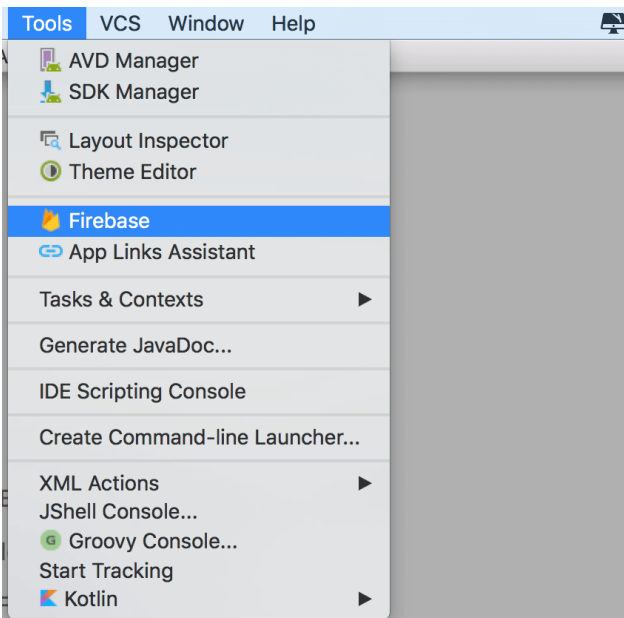
To be able to follow this step-by-step project, we needed the following:

- ✓ The latest version of Android Studio
- ✓ A Firebase account

Chapter 2 Functions and logics

2.1 Functions and logics for Registering using Firebase

Enabling Firebase Authority, we went to the android studio and click **Tools ⇒ Firebase**. Then click on Authentication tab.



We clicked on the **Connect to firebase** option.

Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

[Launch in browser](#)

1 Connect your app to Firebase

✓ Connected

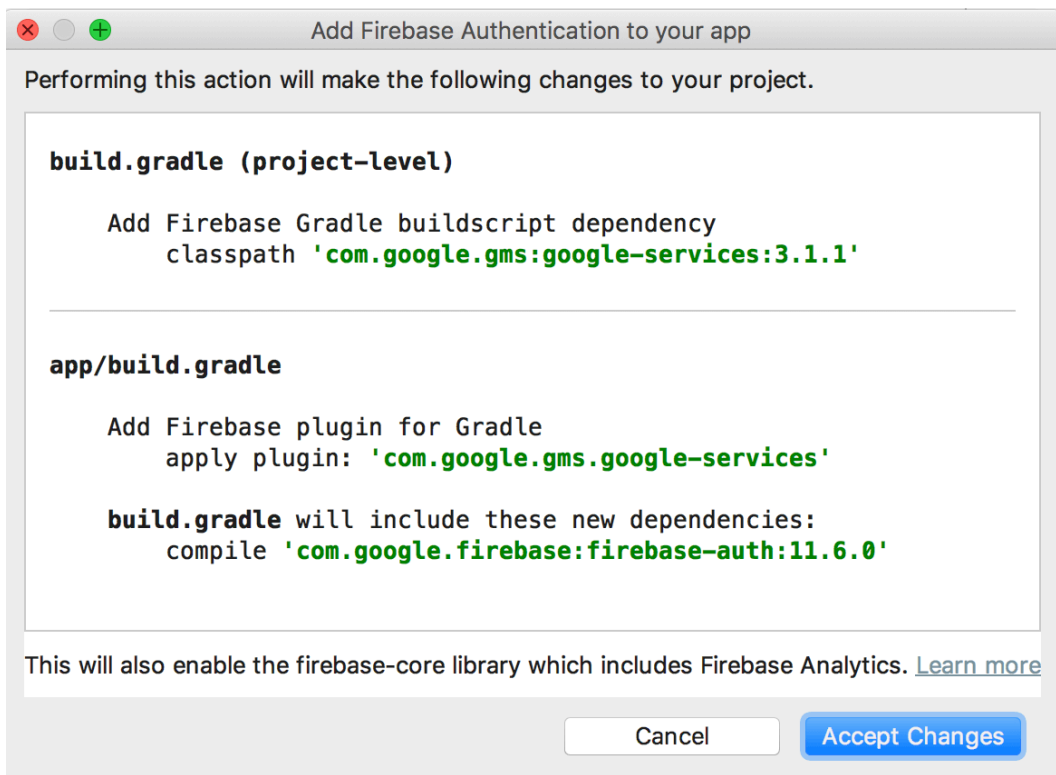
2 Add Firebase Authentication to your app

✓ Dependencies set up correctly

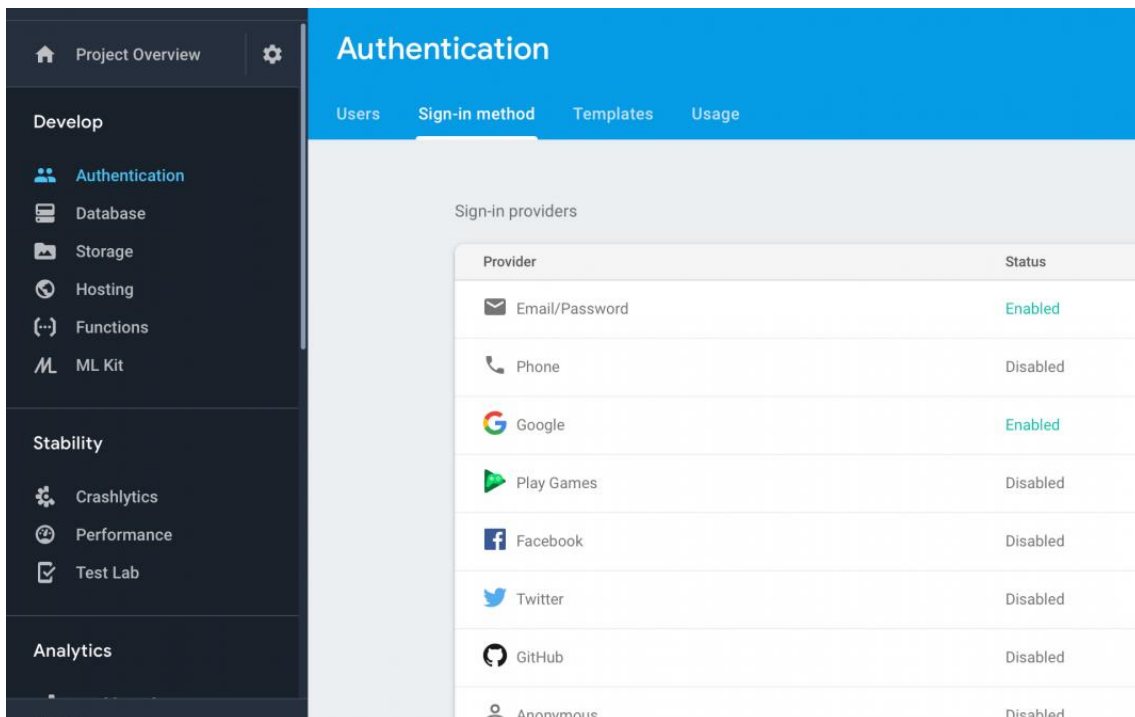
To use an authentication provider, you need to enable it in the [Firebase console](#). Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

When we were creating, we knew we needed to create a new project then we had to give new project name otherwise we could select the project you have made.

4. Then we selected the second option to add **firebase authentication** to our app. We clicked on apply changes. This added plugins and implementation in our Gradle file.



We enabled Authentication Feature by going to the site firebase.google.com. You have to select the authentication tab, then click on sign-in method. then we enabled the option **Email/password**



We created Android Project by opening Android Studio, then went to **File ⇒ New Project** and filled all the details. We opened AndroidManifest.xml file and Internet permission to our app.

```
//for Internet permission to our app
```

```
<uses-permission android:name="android.permission.INTERNET" />
```

Then gradle and plugin will add automatically when you do Step 1.

Our main activity will be in following way, but note I just copied all code of main activity in this case:

MainActivity.java

```
package com.example.maftuna.messaging;
```

```
import android.app.ProgressDialog;
```

```
import android.content.Intent;
```

```
import android.support.annotation.NonNull;
```

```
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
import android.widget.EditText;
```

```
import android.widget.ListView;
```

```
import android.widget.Toast;
```

```
import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
```

```
import com.google.firebase.auth.AuthResult;
```

```
import com.google.firebase.auth.FirebaseAuth;
```

```
//main activity
```

```
public class MainActivity extends AppCompatActivity {
```

```
    Button explicit_btn; //for when login successful, then to go to  
//messenger
```

```
    private EditText email, pass;
```

```
    private FirebaseAuth firebaseAuth;
```

```
    private ListView messagesView;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    email = findViewById(R.id.email);
```

```

    pass = findViewById(R.id.passwprd);
    firebaseAuth = FirebaseAuth.getInstance();
}

public void btnLoginAction(View view) {

    //when network problem, it will say loading message
    final ProgressDialog pd = new ProgressDialog(MainActivity.this);
    pd.setMessage("loading...");
    pd.show();
    pd.show();
    //here when user try to log in
    firebaseAuth.signInWithEmailAndPassword(email.getText().toString(),
pass.getText().toString())
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                pd.dismiss();
                //if log in is successful then it will say successful
//message
                if (task.isSuccessful()) {
                    Toast.makeText(MainActivity.this, "Successful", Toast.LENGTH_SHORT);
                    Intent intent = new Intent(MainActivity.this, Profile.class);
//when it is successful then it permit to log in button to log in messenger
                    explicit_btn = (Button) findViewById(R.id.BtnLogin);
                    explicit_btn.setOnClickListener(new View.OnClickListener() {
                        @Override
                        public void onClick(View v) {

                            Intent intent = new Intent(getApplicationContext(), messagestored.class);
                            startActivity(intent);

                        }
                    });
                    //if there is problem with internet or other errors,
//it will print error message
                } else {
                    Toast.makeText(MainActivity.this, "Error",
Toast.LENGTH_SHORT).show();
                }

            }
        });
}
}

```



```

    }

    public void btnRegisterAction(View view) {

        Intent intent = new Intent(MainActivity.this, Register.class);
        startActivity(intent);

    }
}

```

Then we needed to create a new activity and named that activity **Register.java** and add the following code to that java file.

//register java code:

```

package com.example.maftuna.messaging;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class Register extends AppCompatActivity {
    EditText email, pass;
    private FirebaseAuth firebaseAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        email = findViewById(R.id.email);

```

```

    pass = findViewById(R.id.passwprd);
    firebaseAuth = FirebaseAuth.getInstance();

}

//when internet has some problem, it will show loading message
//during waiting button works

public void btnRegisterAction(View view) {
    final ProgressDialog pd = new ProgressDialog(Register.this);
    pd.setMessage("loading...");
    pd.show();
    pd.show();

    firebaseAuth.createUserWithEmailAndPassword(email.getText().toString(),
pass.getText().toString())
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                pd.dismiss();
                //if register is successful it will say successful
                if (task.isSuccessful()) {
                    Toast.makeText(Register.this, "Successful",
Toast.LENGTH_SHORT).show();
                //then it will start activity
                    Intent intent = new Intent(Register.this, MainActivity.class);
                    startActivity(intent);

                } else {
                    Toast.makeText(Register.this, "Error", Toast.LENGTH_SHORT).show();
                }

            }
        });
}
}

```

Here is Profile.java:

//here is profile java code

Profile.java

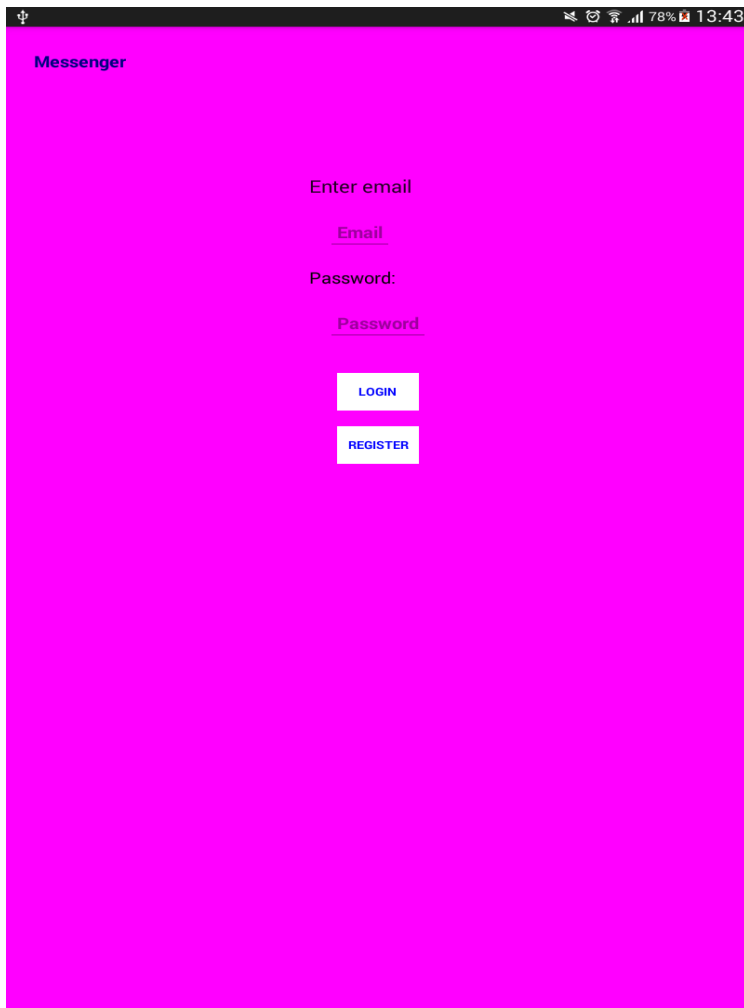
```
package com.example.maftuna.messaging;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

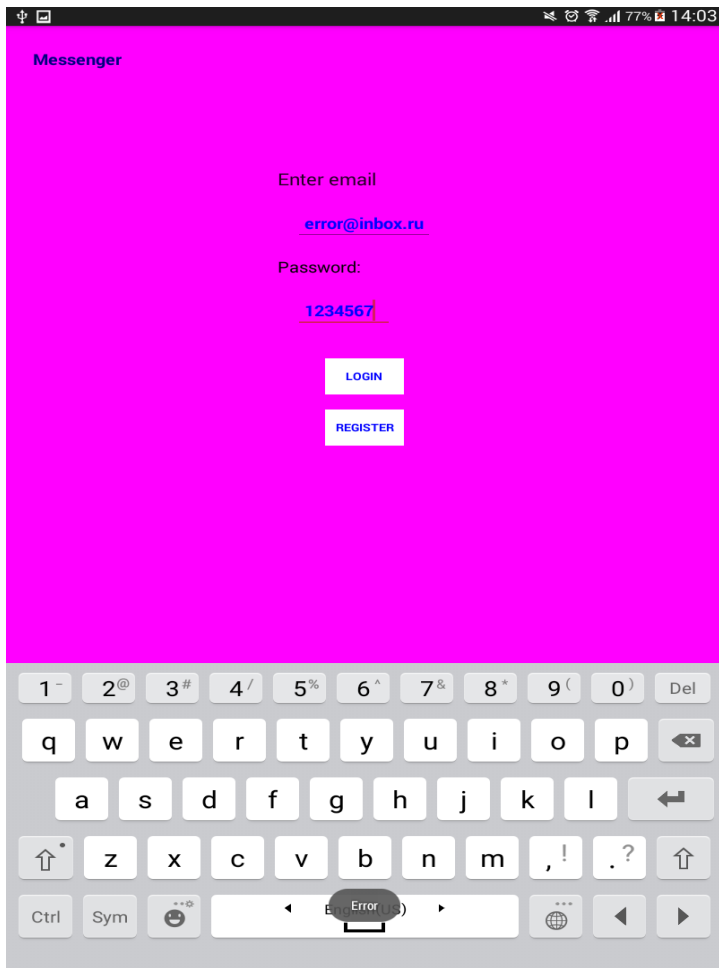
public class Profile extends AppCompatActivity {

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

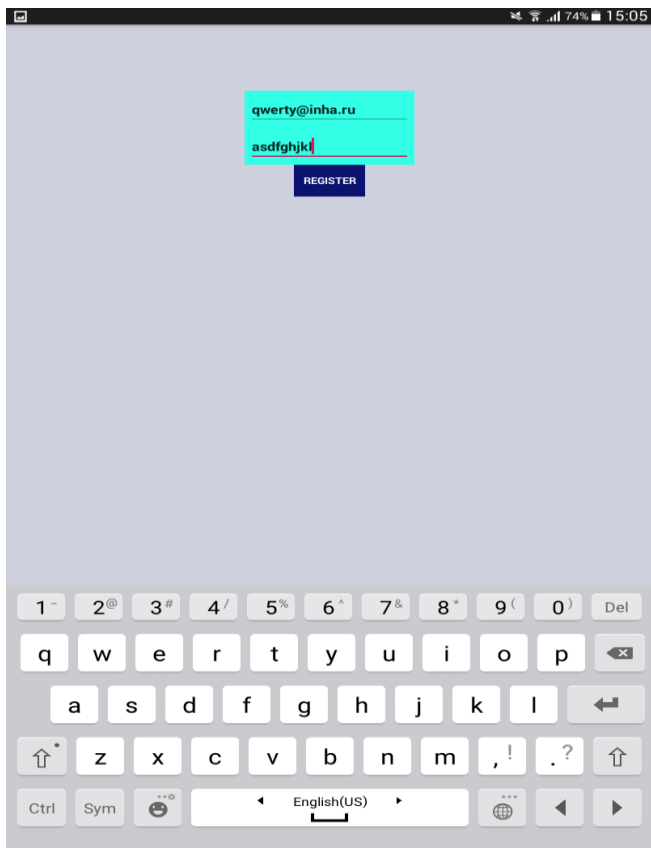
All of above codes will give in following result which are shown as pictures of application:
Note, here User should register firstly and then he should log in. Because without registering he/she cannot log in. after registering and logging in successfully, he can connect to channel.



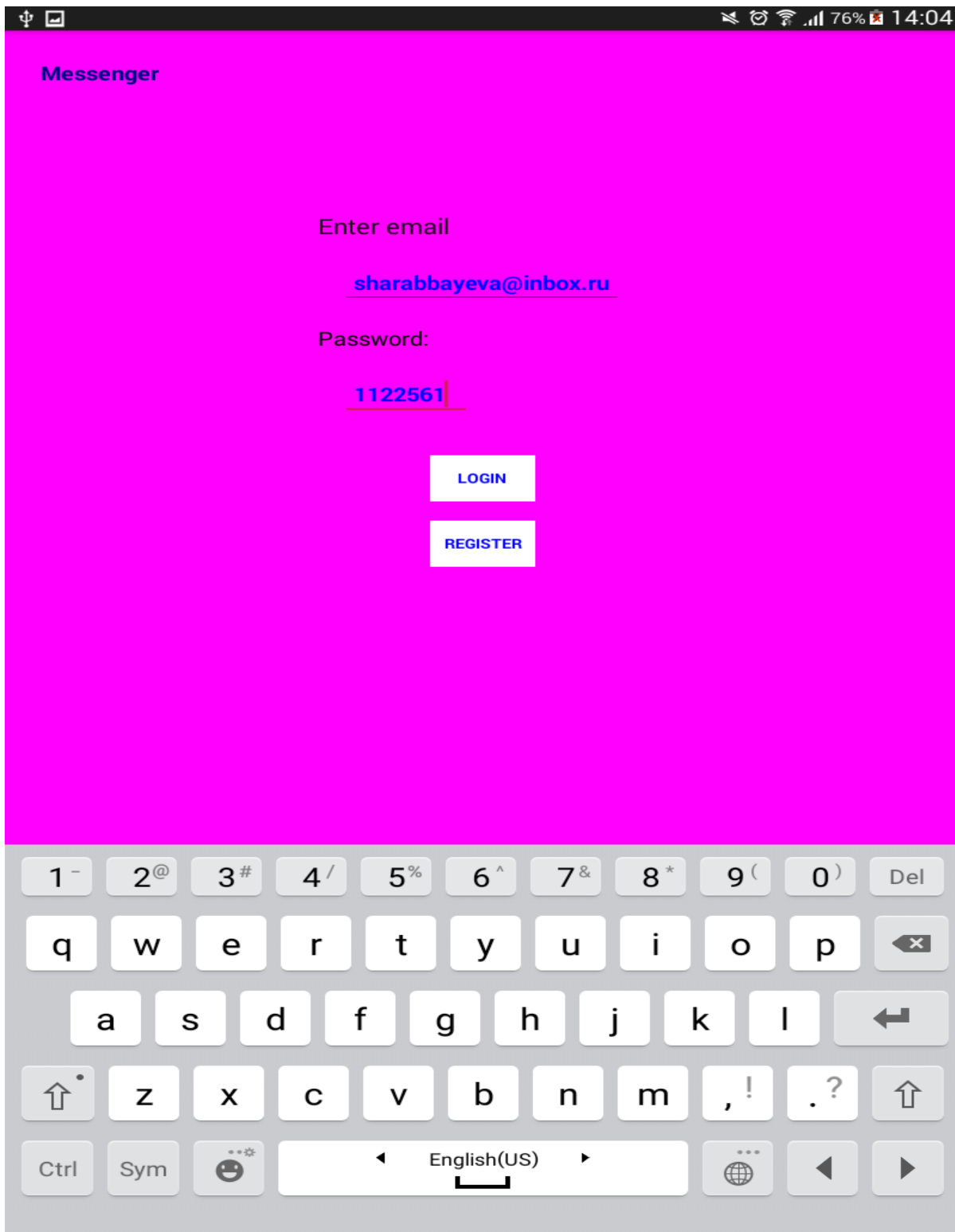
When without registering trying to log in:



When user click button it goes to register window

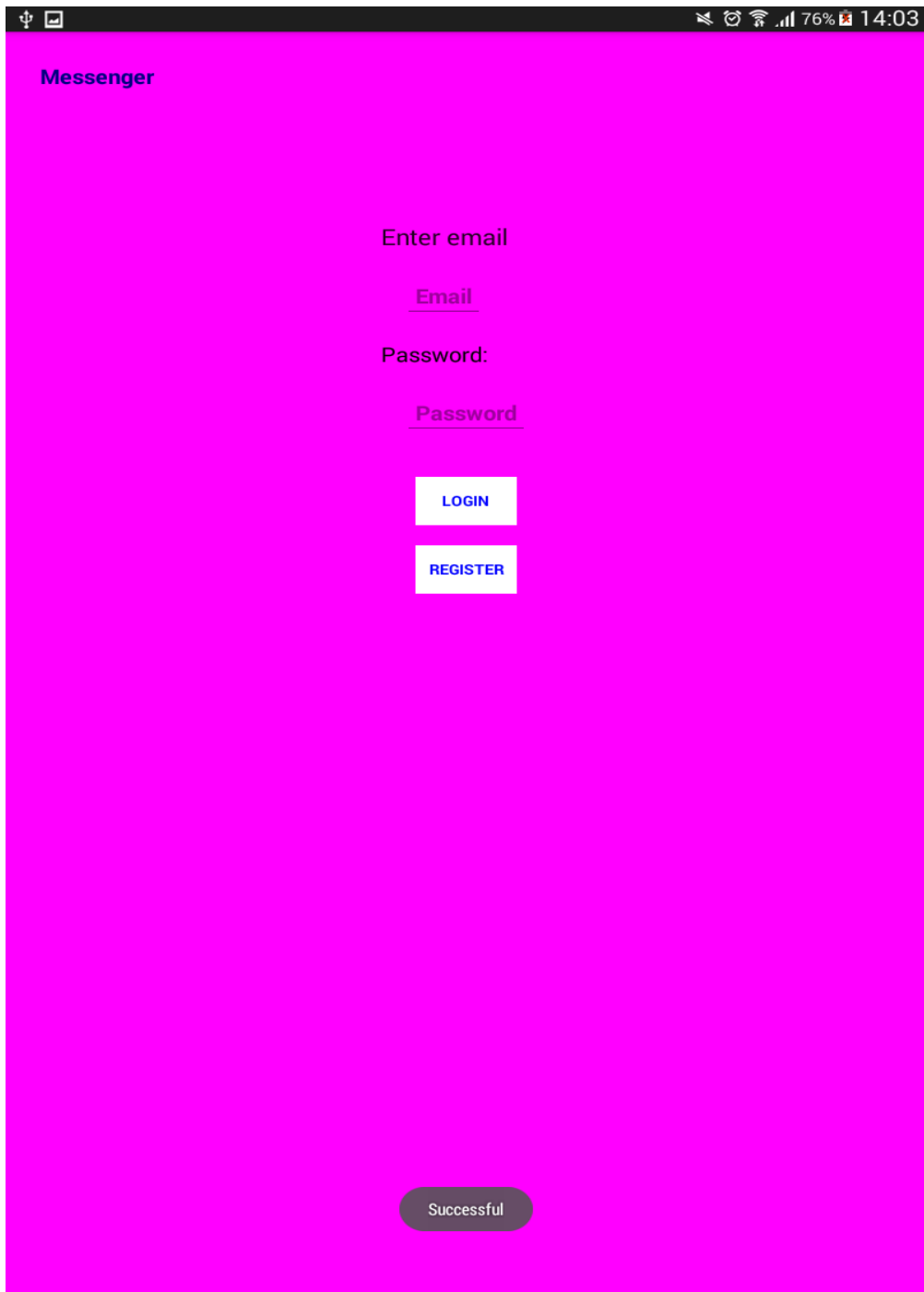


When you already registered just write that email and password to log in:



The screenshot shows a mobile application interface for a 'Messenger'. At the top, there is a status bar with various icons and a battery level of 76% at 14:04. Below the status bar, the word 'Messenger' is displayed in a blue font. The main area of the app is a solid blue color. In the center, there is a white text input field labeled 'Enter email' containing the text 'sharabbayeva@inbox.ru'. Below this, there is another white text input field labeled 'Password:' containing the text '1122561'. At the bottom of the input fields, there are two white buttons with blue text: 'LOGIN' and 'REGISTER'. A virtual keyboard is visible at the bottom of the screen, showing keys for numbers, letters, and symbols, with a language indicator set to 'English(US)'.

After registering successful, then he can log in using those registered email and password



Then it will enter to messenger

hi

this is my first chat project 😊

Write a message



😊

😄

😁

😂

😆

😅

😃

😏

😁

👑

😈

😜

😬

😏

😏

😏

😡

😏

😏

😏

😏

😏

😏

😏

ABC

Sym

🕒

😊

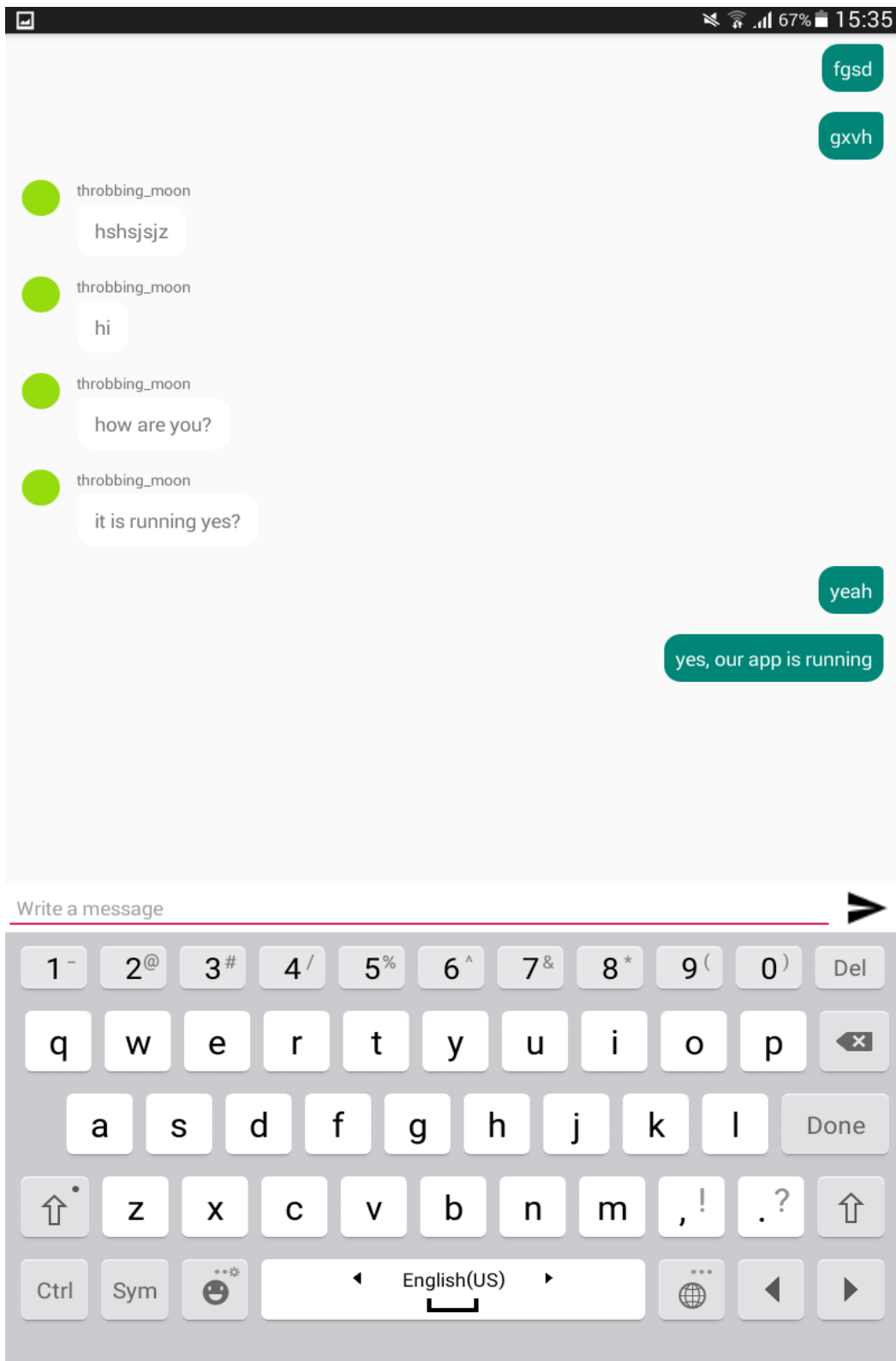
👑

👤

🏠

✳️

⬅️✕



2.2 Functions and logics for activity to make access for registered user to messenger

Android applies Intent for communicating between the components of an Application and also from one application to another application. Intent are the objects which is used in android for passing the information among Activities in an Application and from one app to another also. Intent are used for communicating between the Application components and it also provides the connectivity between two apps.

For example: Intent facilitate you to redirect your activity to another activity on occurrence of any event. By calling, startActivity() you can perform this task.

```
Intent intent = new Intent(getApplicationContext(), messagestored.class);
startActivity(intent);
```

In the above example, foreground activity is getting redirected to another activity i.e. messagestored.java. getApplicationContext() returns the context for your foreground activity.

Let's design the UI of activity_main.xml:

First designed the text view displaying basic details of the App. Second we designed the one button of explicit Intent

```
//here main activity xml file for design
//it will be in linear layout
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false"
    android:focusableInTouchMode="true"
    android:orientation="vertical"
    android:background="@color/fuchsiaColor"
    tools:context=".MainActivity">

    <!-- first TextView -->
    //it will give text about what user should enter in edittext
    //below it
    <TextView
        android:id="@+id/firstTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_margin="20dp"
```

```

    android:padding="10dp"
    android:text="Messenger"
    android:textColor="@color/navy"
    android:textSize="20sp"
    android:textStyle="bold" />
<!--inner layout-->
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginTop="100dp"
    android:orientation="vertical">

    <!-- second TextView -->

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Enter email"
        android:id="@+id/textView3"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <!--for edittext for email to take input for email part-->
    <EditText
        android:id="@+id/email"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_margin="20dp"
        android:padding="10dp"
        android:textColor="#0000FF"
        android:hint="Email"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_toRightOf="@+id/textView3"
        />

    <!--it will give text about what user should enter in edittext
    below it-->
    <TextView
        android:id="@+id/password"
        android:layout_width="wrap_content"

```

```
android:layout_height="wrap_content"
android:layout_below="@+id/textView3"
android:text="Password:"
android:textColor="#000000"
android:textSize="20sp" />
```

<!--for edittext for email to take input for password part-->

```
<EditText
    android:id="@+id/passwprd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/textView3"
    android:layout_centerHorizontal="true"
    android:layout_margin="20dp"
    android:padding="10dp"
    android:hint="Password"
    android:textColor="#0000FF"
    android:textSize="20sp"
    android:textStyle="bold"/>
```

```
</LinearLayout>
```

<!--this is button for log in action-->

```
<Button
    android:id="@+id/BtnLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/password"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:layout_marginTop="20dp"
    android:background="#FFFFFF"
    android:onClick="btnLoginAction"
    android:text="Login "
    android:textColor="#0000FF"
    android:textStyle="bold"/>
```

<!--This is button for register action-->

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:layout_gravity="center"
    android:onClick="btnRegisterAction"
```

```
android:text="Register"
android:layout_centerHorizontal="true"
android:layout_marginTop="20dp"
android:background="#FFFFFF"
android:textColor="#0000FF"
android:textStyle="bold"/>
```

</LinearLayout>

We designed the UI of messengerstored activity messagestored.xml

Now let's design UI of another activity where user will navigate after he click on Explicit button. We went to layout folder, created a new activity and named it activity_messengerstored.xml. In this activity we used TextView to tell user he/she is now on second activity.

Below is the complete code of activity_messagestored.xml

```
<!--it is for message design we used linear layout-->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clipToPadding="false"
    android:focusableInTouchMode="true"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <!-- here is list view-->
    <ListView
        android:id="@+id/messages_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:divider="#fff" />

    <!--inner layout-->
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#fff"
        android:orientation="horizontal">
```

```

    <!--here is place where user can write message to send-->
    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:ems="10"
        android:hint="Write a message"
        android:inputType="text"
        android:paddingHorizontal="10dp"
        android:text="" />

    <!--it is for sending message to channel-->
    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginHorizontal="10dp"
        android:background="@drawable/ic_send_black_24dp"
        android:onClick="sendMessage"
        android:padding="20dp"
        android:scaleType="fitCenter" />
    </LinearLayout>
</LinearLayout>

```

We implemented onClick event for Explicit Button inside MainActivity.java

Now we used `setOnClickListener()` method to implement `onClick` event on button. Explicit button will move to `messagestored.java`.

Below is the complete code of MainActivity.java

// here is main activity in java

```

Button explicit_btn;
package com.example.maftuna.messaging;

import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;

```

```

import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {
    Button explicit_btn;
    private EditText email, pass;
    private FirebaseAuth firebaseAuth;
    private ListView messagesView;

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

        email = findViewById(R.id.email);
        pass = findViewById(R.id.passwprd);
        firebaseAuth = FirebaseAuth.getInstance();
    }

    //when network problem, it will say loading message
    public void btnLoginAction(View view) {

        final ProgressDialog pd = new ProgressDialog(MainActivity.this);
        pd.setMessage("loading...");
        pd.show();
        pd.show();

        //here when user try to log in
        firebaseAuth.signInWithEmailAndPassword(email.getText().toString(),
pass.getText().toString())
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {

                    pd.dismiss();
                    //if log in is successful then it will say successful
                    //message
                    if (task.isSuccessful()) {
                        Toast.makeText(MainActivity.this, "Successful", Toast.LENGTH_SHORT);

```

```
//when log in button is clicked with registered
//account it will go to messenger
explicit_btn = (Button) findViewById(R.id.BtnLogin);
explicit_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent intent = new Intent(getApplicationContext(), messagestored.class);
        startActivity(intent);

    }
});
//if there is problem with internet or other errors,
//it will print error message
} else {
    Toast.makeText(MainActivity.this, "Error",
    Toast.LENGTH_SHORT).show();
}

});

}

public void btnRegisterAction(View view) {

    Intent intent = new Intent(MainActivity.this, Register.class);
    startActivity(intent);

}

}
```

We needed to create another `messengerstored.java` which opens the layout of `activity_messagestored.xml`. Also we will use `Toast` to display message that he is on messenger activity.

Below is the complete code of messagestored.java:


```

package com.example.maftuna.messaging;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.scaledrone.lib.Listener;
import com.scaledrone.lib.Member;
import com.scaledrone.lib.Room;
import com.scaledrone.lib.RoomListener;
import com.scaledrone.lib.Scaledrone;

import java.util.Random;

public class messagestored extends AppCompatActivity implements RoomListener {
    private String channelID = "dk4lYB1b0rZBPOqB"; // this is channel id to
    //communicate
    private String roomName = "observable-room"; // this is observable room
    //for all users
    private EditText editText;
    private Scaledrone scaledrone; // we are using scaledrone to chatting
    private MessageAdapter messageAdapter;
    private ListView messagesView;

    //when it is in messenger it will say it is in messenger
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_messagestored);
        Toast.makeText(getApplicationContext(), "We are in messenger",
        Toast.LENGTH_LONG).show();

        // This is where we write the message
        editText = (EditText) findViewById(R.id.editText);

```

```
messageAdapter = new MessageAdapter(this);  
messagesView = (ListView) findViewById(R.id.messages_view);  
messagesView.setAdapter(messageAdapter);
```

```
MemberData data = new MemberData(getRandomName(), getRandomColor());
```

```
scaledrone = new Scaledrone(channelID, data);  
scaledrone.connect(new Listener() {
```

```
// Successfully connected to Scaledrone room
```

```
@Override
```

```
public void onOpen() {  
    Log.d("Scaledrone", "Scaledrone connection open");  
    scaledrone.subscribe(roomName, messagestored.this);  
}
```

```
// Connecting to Scaledrone room failed
```

```
@Override
```

```
public void onOpenFailure(Exception ex) {  
    System.err.println(ex);  
}
```

```
@Override
```

```
public void onFailure(Exception ex) {  
    System.err.println(ex);  
}
```

```
@Override
```

```
public void onClosed(String reason) {  
    System.err.println(reason);  
}
```

```
});
```

```
}
```

```
public void sendMessage(View view) {  
    String message = editText.getText().toString();  
    if (message.length() > 0) {  
        scaledrone.publish(roomName, message);  
  
        editText.getText().clear();  
    }  
}
```

@Override

```
public void onOpen(Room room) {  
    System.out.println("Conneted to room");  
}
```

@Override

```
public void onOpenFailure(Room room, Exception ex) {  
    System.err.println(ex);  
}
```

// Received a message from Scaledrone room

@Override

```
public void onMessage(Room room, final JsonNode json, final Member member) {  
  
    // To transform the raw JsonNode into a POJO we can use an  
// ObjectMapper  
    final ObjectMapper mapper = new ObjectMapper();  
    try {  
  
        // member.clientData is a MemberData object, let's parse it  
// as such  
        final MemberData data = mapper.treeToValue(member.getClientData(),  
MemberData.class);  
  
        // if the clientID of the message sender is the same as  
// our's it was sent by us  
        boolean belongsToCurrentUser = member.getId().equals(scaledrone.getClientID());  
  
        // since the message body is a simple string in our case we can use json.asText() to parse  
// it as such  
        // if it was instead an object we could use a similar pattern to data parsing  
        final Message message = new Message(json.asText(), data, belongsToCurrentUser);  
        runOnUiThread(new Runnable() {  
            @Override  
            public void run() {  
                messageAdapter.add(message);  
  
                // scroll the ListView to the last added element  
                messagesView.setSelection(messagesView.getCount() - 1);  
            }  
        });  
    } catch (JsonProcessingException e) {  
        e.printStackTrace();  
    }  
}
```

```
}
```

```
//To create a random name, we pre-define two lists of random  
//adjectives and nouns, //then combine them randomly.
```

```
private String getRandomName() {  
    String[] adjs = { "autumn", "hidden", "bitter", "misty", "silent", "empty", "dry",  
"dark", "summer", "icy", "delicate", "quiet", "white", "cool", "spring", "winter",  
"patient", "twilight", "dawn", "crimson", "wispy", "weathered", "blue", "billowing",  
"broken", "cold", "damp", "falling", "frosty", "green", "long", "late", "lingering",  
"bold", "little", "morning", "muddy", "old", "red", "rough", "still", "small",  
"sparkling", "throbbing", "shy", "wandering", "withered", "wild", "black", "young",  
"holy", "solitary", "fragrant", "aged", "snowy", "proud", "floral", "restless", "divine",  
"polished", "ancient", "purple", "lively", "nameless"};  
    String[] nouns = { "waterfall", "river", "breeze", "moon", "rain", "wind", "sea",  
"morning", "snow", "lake", "sunset", "pine", "shadow", "leaf", "dawn", "glitter",  
"forest", "hill", "cloud", "meadow", "sun", "glade", "bird", "brook", "butterfly",  
"bush", "dew", "dust", "field", "fire", "flower", "firefly", "feather", "grass", "haze",  
"mountain", "night", "pond", "darkness", "snowflake", "silence", "sound", "sky",  
"shape", "surf", "thunder", "violet", "water", "wildflower", "wave", "water",  
"resonance", "sun", "wood", "dream", "cherry", "tree", "fog", "frost", "voice",  
"paper", "frog", "smoke", "star"};  
    return (  
        adjs[(int) Math.floor(Math.random() * adjs.length)] +  
            " " +  
        nouns[(int) Math.floor(Math.random() * nouns.length)]  
    );  
}
```

```
private String getRandomColor() {  
    Random r = new Random();  
    StringBuffer sb = new StringBuffer("#");  
    while (sb.length() < 7) {  
        sb.append(Integer.toHexString(r.nextInt()));  
    }  
    return sb.toString().substring(0, 7);  
}  
}
```

```
class MemberData {  
    private String name;  
    private String color;
```

```

public MemberData(String name, String color) {
    this.name = name;
    this.color = color;
}

public MemberData() {
}

public String getName() {
    return name;
}

public String getColor() {
    return color;
}

@Override
public String toString() {
    return "MemberData{" +
        "name='" + name + '\'' +
        ", color='" + color + '\'' +
        '}';
}
}

```

Manifest file: We must be sure Manifest file has both the MainActivity and messagestored listed it. Also here MainActivity is our main activity which will be launched first. So we must make sure intent-filter is correctly added just below MainActivity.

Below is the code of Manifest file:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.maftuna.messaging">

    <uses-permission android:name="android.permission.INTERNET" />

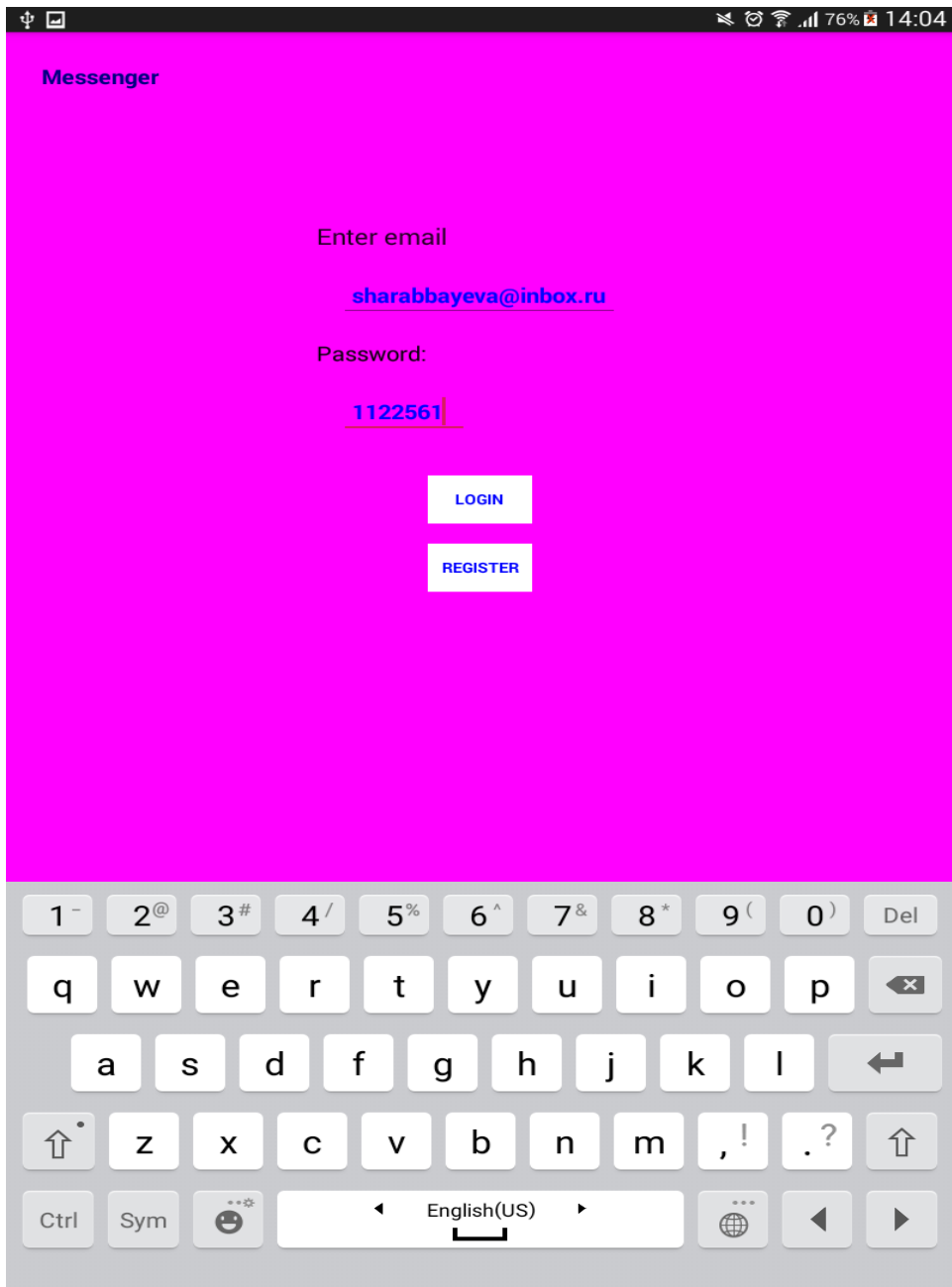
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"

```

```
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".Register" />
    <activity android:name=".Profile" />
    <activity android:name=".messagestored"></activity>
</application>

</manifest>
```



We are in messenger

Write a message



2.4 Functions and logics for messaging using Scaledrone Java API Client

How we built a fully functional group chat. We designed the UI elements such as chat bubbles and text inputs. We used Scaledrone as the realtime backend of our app. The project might seem daunting at first, but the messaging code outside of the layout files is pretty short. We start Import the Scaledrone module. To add the Scaledrone dependency to our app, we need to add it to our build.gradle file.


```
dependencies {  
    implementation 'com.scaledrone:scaledrone-java:0.3.0'  
}
```

We already allowed internet connection. So next, we need to define the UI layout where we need to build empty state. It consists of an empty ListView into where the messages will go, an EditText where the user can type their message and finally, an ImageButton as a button to send the message



The base layout is defined in /res/layout/activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:clipToPadding="false"
android:focusableInTouchMode="true"
android:orientation="vertical"
android:background="@color/fuchsiaColor"
tools:context=".MainActivity">
```

<TextView

```
android:id="@+id/firstTextView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_margin="20dp"
android:padding="10dp"
android:text="Messenger"
android:textColor="@color/navy"
android:textSize="20sp"
android:textStyle="bold" />
```

<!-- second TextView -->

<LinearLayout

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginTop="100dp"
android:orientation="vertical">
```

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textAppearance="?android:attr/textAppearanceLarge"
android:text="Enter email"
android:id="@+id/textView3"
android:layout_alignParentTop="true"
android:layout_centerHorizontal="true" />
```

<EditText

```
android:id="@+id/email"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerHorizontal="true"
android:layout_margin="20dp"
android:padding="10dp"
```

```
android:textColor="#0000FF"
android:hint="Email"
android:textSize="20sp"
android:textStyle="bold"
android:layout_toRightOf="@+id/textView3"
/>
```

```
<TextView
    android:id="@+id/password"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textView3"
    android:text="Password:"
    android:textColor="#000000"
    android:textSize="20sp" />
```

```
<EditText
    android:id="@+id/passwprd"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toRightOf="@+id/textView3"
    android:layout_centerHorizontal="true"
    android:layout_margin="20dp"
    android:padding="10dp"
    android:hint="Password"
    android:textColor="#0000FF"
    android:textSize="20sp"
    android:textStyle="bold"/>
```

```
</LinearLayout>
```

```
<Button
    android:id="@+id/BtnLogin"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/password"
    android:layout_centerHorizontal="true"
    android:layout_gravity="center"
    android:layout_marginTop="20dp"
    android:background="#FFFFFF"
    android:onClick="btnLoginAction"
    android:text="Login "
    android:textColor="#0000FF"
    android:textStyle="bold"/>
```

```

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:layout_gravity="center"
    android:onClick="btnRegisterAction"
    android:text="Register"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:background="#FFFFFF"
    android:textColor="#0000FF"
    android:textStyle="bold"/>

```

</LinearLayout>

String constants are defined in /res/values/strings.xml:

```

<resources>
    <string name="app_name">Messenger</string>
    <string name="input_placeholder">Write a message</string>
    <string name="count_initial_value">0</string>
    <string name="button_label">Send</string>
    <string name="toast_message">Hurra!</string>

</resources>

```

The icon for the send button is defined in /res/drawable/ic_send_black_24dp.xml:

```

<vector xmlns:android="http://schemas.android.com/apk/res/android"
    android:width="24dp"
    android:height="24dp"
    android:viewportWidth="24.0"
    android:viewportHeight="24.0">
    <path
        android:fillColor="#FF000000"
        android:pathData="M2.01,21L23,12 2.01,3 2,10l15,2 -15,2z"/>
</vector>

```

Next up, chat bubbles! Our chat app is going to have two type of chat bubbles: a bubble for messages sent by us and bubbles for messages sent by others. Chat bubble sent by us is the messages sent by us will look dark and be aligned to the right. We're using a drawable to get the border radius effect.

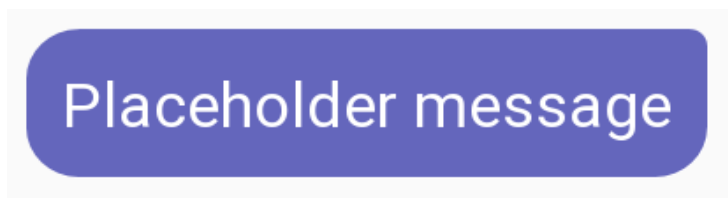
/res/drawable/my_message.xml:

```

<shape
xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/colorPrimary" />
    <corners android:topRightRadius="5dp" android:radius="15dp"
/>
</shape>

```

The message itself is just a simple TextView aligned to the right.



/res/layout/my_message.xml:

```

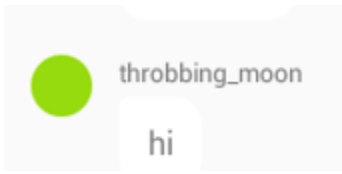
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingVertical="10dp"
    android:paddingRight="15dp"
    android:paddingLeft="60dp"
    android:clipToPadding="false">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/message_body"
        android:background="@drawable/my_message"
        android:textColor="#fff"
        android:padding="10dp"
        android:elevation="2dp"
        android:textSize="18dp"
        android:layout_alignParentRight="true"
        android:text="Placeholder message"
    />

</RelativeLayout>

```

In chat bubble sent by others The chat bubble sent by others within the group chat will be light and aligned to the left. In addition to the bubble itself, we will show an avatar (as a simple full-color circle) and the name of the user.



For the avatar let's define a circle shape under `/res/drawable/circle.xml`:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="oval">
    <solid android:color="#48b3ff"/>
</shape>
```

And for the bubble let's create a shape with curved corners and the sharp corner on the left. This goes in `/res/drawable/their_message.xml`:

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="#fff" />
    <corners android:topLeftRadius="5dp" android:radius="15dp" />
</shape>
```

Putting it together their message bubble layout under `/res/layout/their_message.xml` will look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clipToPadding="false"
    android:paddingLeft="15dp"
    android:paddingRight="60dp"
    android:paddingVertical="10dp">

    <View
        android:id="@+id/avatar"
        android:layout_width="34dp"
        android:layout_height="34dp"
        android:layout_alignParentLeft="true"
        android:background="@drawable/circle"
        android:scaleType="centerInside" />

    <TextView
        android:id="@+id/name"
```

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignTop="@+id/avatar"
android:layout_marginLeft="15dp"
android:layout_toRightOf="@+id/avatar"
android:paddingBottom="4dp"
android:text="Rose" />
```

<TextView

```
    android:id="@+id/message_body"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/name"
    android:layout_below="@+id/name"
    android:background="@drawable/their_message"
    android:elevation="2dp"
    android:paddingHorizontal="16dp"
    android:paddingVertical="12dp"
    android:text="How are you doing? This is a long message that should probably
wrap."
    android:textSize="18dp" />
```

</RelativeLayout>

Let's hook up the realtime messaging logic. We find the EditText view from our layout and extend Scaledrone's RoomListener so we could receive messages. Most of the methods will have minimal code in them, and we'll fill them up as the tutorial goes along.

```
package com.example.maftuna.messaging;
```

```
import android.app.ProgressDialog;
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
```

```

import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {
    Button explicit_btn;
    private EditText email, pass;
    private FirebaseAuth firebaseAuth;
    private ListView messagesView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        email = findViewById(R.id.email);
        pass = findViewById(R.id.password);
        firebaseAuth = FirebaseAuth.getInstance();
    }

    public void btnLoginAction(View view) {

        final ProgressDialog pd = new ProgressDialog(MainActivity.this);
        pd.setMessage("loading...");
        pd.show();
        pd.show();

        firebaseAuth.signInWithEmailAndPassword(email.getText().toString(),
pass.getText().toString())
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {

                    pd.dismiss();

                    if (task.isSuccessful()) {
                        Toast.makeText(MainActivity.this, "Successful", Toast.LENGTH_SHORT);
                        Intent intent = new Intent(MainActivity.this, Profile.class);

                        explicit_btn = (Button) findViewById(R.id.BtnLogin);
                        explicit_btn.setOnClickListener(new View.OnClickListener() {

```


@Override

public void onClick(View v) {

Intent intent = **new** Intent(getBaseContext(), messagestored.**class**);
startActivity(intent);

}
});

} **else** {

Toast.makeText(MainActivity.**this**, "**Error**",
Toast.**LENGTH_SHORT**).show();
}

}
});

}

public void btnRegisterAction(View view) {

Intent intent = **new** Intent(MainActivity.**this**, Register.**class**);
startActivity(intent);

}
}

Let's connect to Scaledrone. We did not have Scaledrone account yet, so we opened up [Scaledrone.com](https://scaledrone.com) and created a new free account. To successfully connect to Scaledrone we needed to get our own channel ID from the Scaledrone's dashboard. To do that we went to the dashboard and clicked the big green +Create Channel button to get started. We could choose Never require authentication for now. We copied the channel ID from the just created channel and replaced CHANNEL_ID_FROM_YOUR_SCALEDRONE_DASHBOARD with it.

Connecting to Scaledrone happens within the onCreate() method, right after we have set up the UI. Scaledrone gives us the ability to attach arbitrary data to a user (users are called members in Scaledrone lingo), we're going to be adding a random name and color.

MemberData data = **new** MemberData(getRandomName(), getRandomColor());

scaledrone = **new** Scaledrone(**channelID**, data);

```

scaledrone.connect(new Listener() {
    @Override
    public void onOpen() {
        Log.d("Scaledrone", "Scaledrone connection open");
        // Since the messagestroed itself already implement RoomListener
        //we can pass it as a target
        scaledrone.subscribe(roomName, messagestored.this);
    }

    @Override
    public void onOpenFailure(Exception ex) {
        System.err.println(ex);
    }

    @Override
    public void onFailure(Exception ex) {
        System.err.println(ex);
    }

    @Override
    public void onClosed(String reason) {
        System.err.println(reason);
    }
});

```

You might have noticed that we named our name Scaledrone room observable-room. You can name the room anything you want, a single user can actually connect to an infinite amount of rooms to provider for all sorts of application scenarios. To create the MemberData we implemented the getRandomName() and getRandomColor() functions as well as the MemberData class itself. For the sake of keeping project simple, we'll define a random username on the client side of the application. To create a random name, we pre-define two lists of random adjectives and nouns, then combine them randomly. The random color function will be generating a random seven-character color hex such as #FF0000.

```

public class messagestored extends AppCompatActivity implements RoomListener {
    * ...
    more code
    ... */

    //To create a random name, we pre-define two lists of random
    //adjectives and nouns, then combine them randomly.
    private String getRandomName() {
        String[] adjs = { "autumn", "hidden", "bitter", "misty", "silent", "empty", "dry",
        "dark", "summer", "icy", "delicate", "quiet", "white", "cool", "spring", "winter",

```

```
"patient", "twilight", "dawn", "crimson", "wispy", "weathered", "blue", "billowing",
"broken", "cold", "damp", "falling", "frosty", "green", "long", "late", "lingering",
"bold", "little", "morning", "muddy", "old", "red", "rough", "still", "small",
"sparkling", "throbbing", "shy", "wandering", "withered", "wild", "black", "young",
"holy", "solitary", "fragrant", "aged", "snowy", "proud", "floral", "restless", "divine",
"polished", "ancient", "purple", "lively", "nameless"};
```

```
String[] nouns = {"waterfall", "river", "breeze", "moon", "rain", "wind", "sea",
"morning", "snow", "lake", "sunset", "pine", "shadow", "leaf", "dawn", "glitter",
"forest", "hill", "cloud", "meadow", "sun", "glade", "bird", "brook", "butterfly",
"bush", "dew", "dust", "field", "fire", "flower", "firefly", "feather", "grass", "haze",
"mountain", "night", "pond", "darkness", "snowflake", "silence", "sound", "sky",
"shape", "surf", "thunder", "violet", "water", "wildflower", "wave", "water",
"resonance", "sun", "wood", "dream", "cherry", "tree", "fog", "frost", "voice",
"paper", "frog", "smoke", "star"};
```

```
return (
    ads[(int) Math.floor(Math.random() * ads.length)] +
    "_" +
    nouns[(int) Math.floor(Math.random() * nouns.length)]
);
}

private String getRandomColor() {
    Random r = new Random();
    StringBuffer sb = new StringBuffer("#");
    while (sb.length() < 7) {
        sb.append(Integer.toHexString(r.nextInt()));
    }
    return sb.toString().substring(0, 7);
}
}
```

The MemberData class is super minimal and will later be serialized into JSON and sent to users by Scaledrone.

```
class MemberData {
    private String name;
    private String color;

    // Add an empty constructor so we can later parse JSON into MemberData using Jackson
    public MemberData(String name, String color) {
        this.name = name;
        this.color = color;
    }

    public MemberData() {
    }
}
```

```

public String getName() {
    return name;
}

public String getColor() {
    return color;
}

@Override
public String toString() {
    return "MemberData{" +
        "name='" + name + '\'' +
        ", color='" + color + '\'' +
        '}';
}
}

```

To send (or publish) the message to the Scaledrone room we need to add a `onClick()` handler to the `ImageButton` in the `activity_messagestored.xml` file.

<ImageButton

```

android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center"
android:layout_marginHorizontal="10dp"
android:background="@drawable/ic_send_black_24dp"
android:onClick="sendMessage"
android:padding="20dp"
android:scaleType="fitCenter" /> <!-- ⌘ This is the onClick that we
just added -->

```

Then we added the `sendMessage()` function to the `MainActivity`. If the user has input something we send the message to the same observable-room as we subscribed to above. After the message has been sent we can clear the `EditText` view for convenience. Scaledrone will take care of the message and deliver it to everyone that has subscribed to the observable-room room in your channel.

```

public void sendMessage(View view) {
    String message = editText.getText().toString();
    if (message.length() > 0) {
        scaledrone.publish(roomName, message);

        editText.getText().clear();
    }
}

```

```
}  
}
```

To display messages, as seen in the layout file the messages are going to be displayed via ListView. To use a ListView we need to create a class that extends android.widget.BaseAdapter. This class is then used as the state of the ListView. Let's define our MessageAdapter as well as the Message class itself. The Message class will hold all the needed info to render a single message.

```
// Message.java  
package com.example.maftuna.messaging;  
  
public class Message {  
    private String text; // message body  
    private MemberData data; // data of the user that sent this message  
    private boolean belongsToCurrentUser; // is this message sent by us?  
  
    public Message(String text, MemberData data, boolean belongsToCurrentUser) {  
        this.text = text;  
        this.data = data;  
        this.belongsToCurrentUser = belongsToCurrentUser;  
    }  
  
    public String getText() {  
        return text;  
    }  
  
    public MemberData getData() {  
        return data;  
    }  
  
    public boolean isBelongsToCurrentUser() {  
        return belongsToCurrentUser;  
    }  
}
```

The MessageAdapter defines how we render our rows within the ListView.

```
// MessageAdapter.java  
package com.example.maftuna.messaging;  
  
import android.app.Activity;  
import android.content.Context;  
import android.graphics.Color;
```

```
import android.graphics.drawable.GradientDrawable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;
```

```
import java.util.ArrayList;
import java.util.List;
```

```
public class MessageAdapter extends BaseAdapter {
```

```
    List<Message> messages = new ArrayList<Message>();
    Context context;
```

```
    public MessageAdapter(Context context) {
        this.context = context;
    }
```

```
    public void add(Message message) {
        this.messages.add(message);
        notifyDataSetChanged();    // to render the list we need to notify
    }
```

```
    @Override
    public int getCount() {
        return messages.size();
    }
```

```
    @Override
    public Object getItem(int i) {
        return messages.get(i);
    }
```

```
    @Override
    public long getItemId(int i) {
        return i;
    }
```

```
// This is the backbone of the class, it handles the creation of
//single ListView row (chat bubble)
```

@Override

```
public View getView(int i, View convertView, ViewGroup viewGroup) {
    MessageViewHolder holder = new MessageViewHolder();
    LayoutInflater messageInflater = (LayoutInflater)
context.getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
    Message message = messages.get(i);

    if (message.isBelongsToCurrentUser()) { // this message was sent by us
//so let's create a basic chat bubble on the right
        convertView = messageInflater.inflate(R.layout.my_message, null);
        holder.messageBody = (TextView) convertView.findViewById(R.id.message_body);
        convertView.setTag(holder);
        holder.messageBody.setText(message.getText());
    } else { // this message was sent by someone else so let's
create an advanced chat bubble on the left
        convertView = messageInflater.inflate(R.layout.their_message, null);
        holder.avatar = (View) convertView.findViewById(R.id.avatar);
        holder.name = (TextView) convertView.findViewById(R.id.name);
        holder.messageBody = (TextView) convertView.findViewById(R.id.message_body);
        convertView.setTag(holder);

        holder.name.setText(message.getData().getName());
        holder.messageBody.setText(message.getText());
        GradientDrawable drawable = (GradientDrawable) holder.avatar.getBackground();
        drawable.setColor(Color.parseColor(message.getData().getColor()));
    }

    return convertView;
}

class MessageViewHolder {
    public View avatar;
    public TextView name;
    public TextView messageBody;
}
```

To Receive messages, Now that we can display and render our chat bubbles we need to hook up the incoming messages with the MessageAdapter that we just created. We can do that by going back to the MainActivity class and finishing the onMessage() method. Scaledrone uses the popular Jackson JSON library for serializing and parsing the messages, and it comes bundled with the Scaledrone API client.

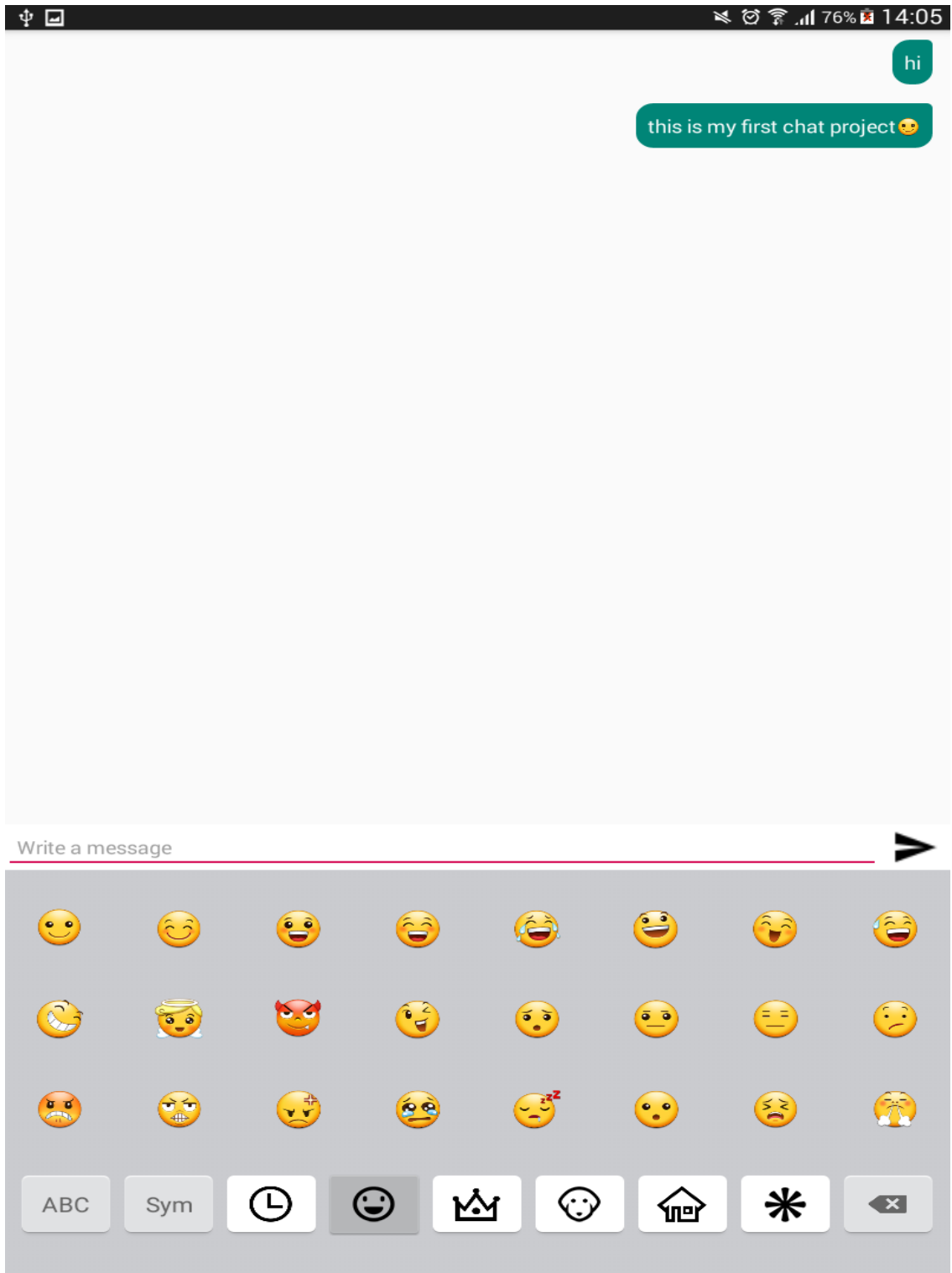
@Override

```
public void onMessage(Room room, final JsonNode json, final Member member) {  
    final ObjectMapper mapper = new ObjectMapper();  
    try {  
        final MemberData data = mapper.treeToValue(member.getClientData(),  
MemberData.class);  
        boolean belongsToCurrentUser = member.getId().equals(scaledrone.getClientID());  
        final Message message = new Message(json.asText(), data, belongsToCurrentUser);  
        runOnUiThread(new Runnable() {  
            @Override  
            public void run() {  
                messageAdapter.add(message);  
                messagesView.setSelection(messagesView.getCount() - 1);  
            }  
        });  
    } catch (JsonProcessingException e) {  
        e.printStackTrace();  
    }  
}
```

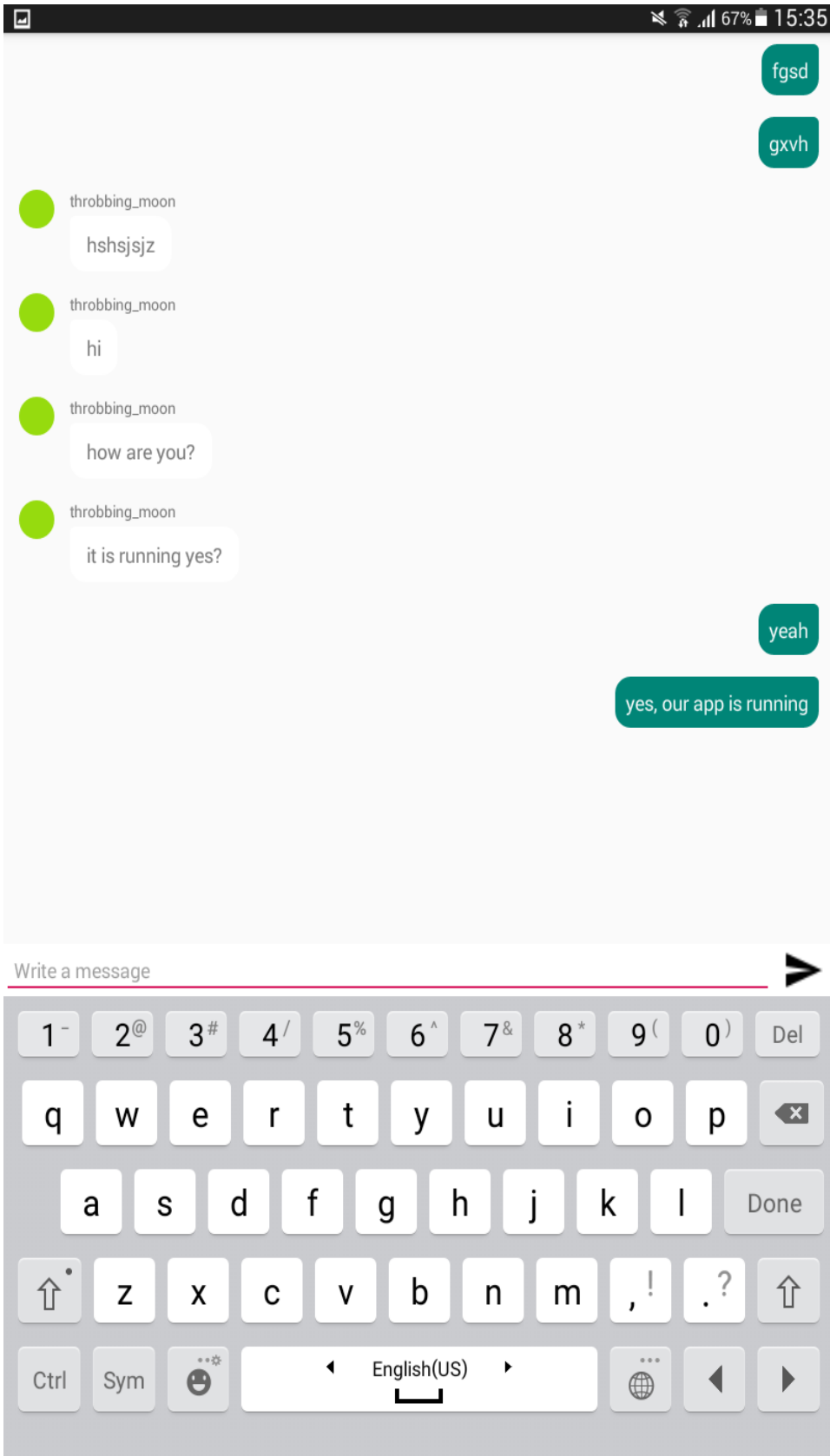
When we register and logged in, first time of messenger condition will be:



Then when we start texting:



When we others start texting:



Chapter 3 Conclusion

Overall project gave new experience of learning how to program in android. We learnt how to connect to database and make nice chatting application with friendly user interface. It took time to make program properly but in the end we can program these kind of project easily and with minimum effort.

Chapter 4 Reference

<https://abhiandroid.com>

<https://www.scaledrone.com/>

<https://firebase.google.com/>