Mobile Application Development Laboratory

Mini Project

Project Title : Chat Application using Firebase

Team:

Ashwin.P 211614104026

Chahat Agarwal 211614104029

CHAT APPLICATION USING FIREBASE

Abstract:

Since the Android platform was first open sourced by Google in November 2007, it has attracted more than 180,000 developers and the deployment of 50,000 mobile applications in the Android Market. Today more than 60 smart phones from major manufactures run the Android platform. All these numbers show that the Android projects has gained momentum and has moved forward. In addition to its openness, all the tools in the Android development are free and no special hardware is required.

People mainly use a mobile phone for ease of communication. Wide range of android applications were developed to satisfy the communication needs of the people and among them the one which is widely used by the current population is WhatsApp. Hence this concept of communications motivated us to create a simple application that can be used by the people to communicate each other.

We have developed a Chat Application whose platform of working is Android operating system, using Android Studios. The programming language is Java and XML. Firebase is used as the database in the application.

Scope and Important Features

The main scope of the project is to create a Chat Application which can be used for communicating among people.

The Chat Application uses Firebase as its database. **Firebase** is a powerful platform for building iOS, **Android**, and web-based apps, offering real-time data storage and synchronization, user authentication, and more features. All you need is a Google Account in order to access a firebase account.

It is an ideal feature for Android project that needs intensive backend database scripting. Firebase has a broad spectrum of functionalities but

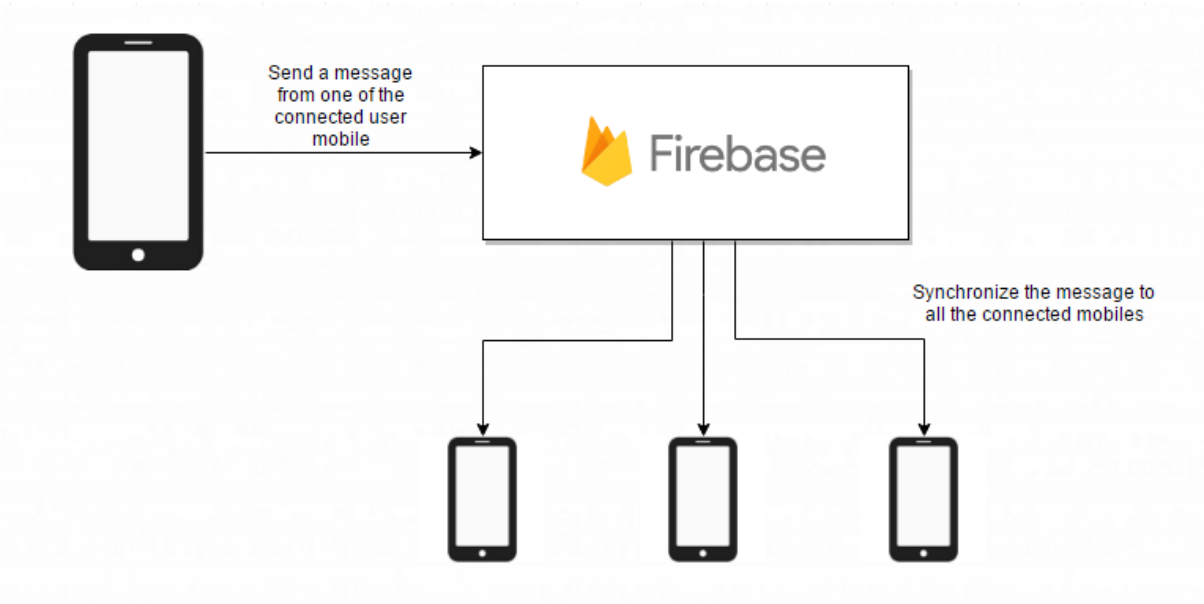
at the same time it is convenient to use. To brief the common advantages of Firebase are as follows:

- Simple App hosting
- Ability to set User Authentication
- Manipulating. Reordering, and eliminating data
- Availability of data analytics
- Flexible front end

Working of the Application

The Chat Application works in such a way that initially the users who access the application are asked to register themselves with any of their Email Accounts.
Once the user is registered to it he/she can view the messages of all the people who are connected to the same firebase database and also send messages which is directed to all the users who are connected to the particular database.
The messages when sent from the user are pushed to the database. Messages are stored in the database as a JSON (Java Script Object Notation) format every time a child node is created for each and every message.
Then further by synchronising these messages, they are broadcasted to all the users who are connected to the user database.

Application Architecture Diagram:



Explanation:

The above figure represents the working of the Chat Application. Initially all the users are connected to the firebase account of the application owner. Once the message is sent by one of the connected user, the message get pushed to the firebase database. Then further these messages are sent to all the users who are synchronised/connected to the particular firebase database.

Source Code:

XML Source Code:

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.android.friendlychat.MainActivity">

    <ListView
        android:id="@+id/messageListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"
        android:divider="@android:color/transparent"
        android:transcriptMode="alwaysScroll"
        tools:listitem="@layout/item_message"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:orientation="horizontal">

        <ImageButton
            android:id="@+id/photoPickerButton"
            android:layout_width="36dp"
            android:layout_height="36dp"
            android:background="@android:drawable/ic_menu_gallery" />

        <EditText
            android:id="@+id/messageEditText"
            android:layout_width="0dp"
```

```
            android:layout_height="wrap_content"
            android:layout_gravity="center_vertical"
            android:layout_weight="1" />

        <Button
            android:id="@+id/sendButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:enabled="false"
            android:text="@string/send_button_label"/>

    </LinearLayout>

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"/>
</RelativeLayout>
```

Item_message.xml

```
<?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="wrap_content"
              android:layout_marginLeft="@dimen/activity_horizontal_margin"
              android:layout_marginStart="@dimen/activity_horizontal_margin"
              android:orientation="vertical">

    <ImageView
        android:id="@+id/photoImageView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:adjustViewBounds="true" />

    <TextView
        android:id="@+id/messageTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:textAppearance="?android:attr/textAppearanceLarge"
        tools:text="Message" />
```

```
    <TextView
        android:id="@+id/nameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="0"
        android:textAppearance="?android:attr/textAppearanceSmall"
        tools:text="Name" />

</LinearLayout>
```

Colours.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#FF9800</color>
    <color name="colorPrimaryDark">#E65100</color>
    <color name="colorAccent">#2E7D32</color>
    <color name="colorTitle">#ffffff</color>
</resources>
```

Strings.xml

```
<resources>
    <string name="app_name">Friendly Chat</string>
    <string name="sign_out">Sign Out</string>
    <string name="send_button_label">Send</string>
</resources>
```

Java Source Code:

MainActivity.java

```
package com.google.firebase.udacity.friendlychat;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.text.Editable;
import android.text.InputFilter;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.firebase.ui.auth.*;
import com.firebase.ui.auth.BuildConfig;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.ActionCodeResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.ChildEventListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.remoteconfig.FirebaseRemoteConfig;
import com.google.firebase.remoteconfig.FirebaseRemoteConfig;
import
com.google.firebase.remoteconfig.FirebaseRemoteConfigSettings;
```

```
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.net.URI;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static android.R.attr.data;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "MainActivity";

    public static final String ANONYMOUS = "anonymous";
    public static final int DEFAULT_MSG_LENGTH_LIMIT = 1000;
    public static final String FRIENDLY_MESSAGE_LENGTH_KEY =
"friendly_message_length";
    public static final int RC_SIGN_IN = 1;
    public static final int RC_PHOTO_PICKER = 2;

    private ListView mMessageListView;
    private MessageAdapter mMessageAdapter;
    private ProgressBar mProgressBar;
    private ImageButton mPhotoPickerButton;
    private EditText mMessageEditText;
    private Button mSendButton;
    private String mUsername;
    private FirebaseDatabase firebaseDatabase;
    private DatabaseReference databaseReference;
    private ChildEventListener childEventListener;
    private FirebaseAuth firebaseAuth;
    private FirebaseAuth.AuthStateListener authStateListener;

    //For image storage purposes
    private FirebaseStorage firebaseStorage;
    private StorageReference photoStorageReference;
    private FirebaseRemoteConfig firebaseRemoteConfig;

    @Override
```

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

        mUsername = ANONYMOUS;

        firebaseDatabase = FirebaseDatabase.getInstance();
        firebaseAuth = FirebaseAuth.getInstance();
        firebaseStorage = FirebaseStorage.getInstance();
        firebaseRemoteConfig = FirebaseRemoteConfig.getInstance();

        databaseReference =
firebaseDatabase.getReference().child("messages");
        photoStorageReference =
firebaseStorage.getReference().child("chat_photos");


        // Initialize references to views
        mProgressBar = (ProgressBar) findViewById(R.id.progressBar);
        mMessageListView = (ListView)
findViewById(R.id.messageListView);
        mPhotoPickerButton = (ImageButton)
findViewById(R.id.photoPickerButton);
        mMessageEditText = (EditText)
findViewById(R.id.messageEditText);
        mSendButton = (Button) findViewById(R.id.sendButton);

        // Initialize message ListView and its adapter
        List<FriendlyMessage> friendlyMessages = new ArrayList<>();
        mMessageAdapter = new MessageAdapter(this,
R.layout.item_message, friendlyMessages);
        mMessageListView.setAdapter(mMessageAdapter);

        // Initialize progress bar
        mProgressBar.setVisibility(ProgressBar.INVISIBLE);

        // ImagePickerButton shows an image picker to upload a image for
a message
        mPhotoPickerButton.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View view) {
                // TODO: Fire an intent to show an image picker
```

```
            Intent intent= new Intent(Intent.ACTION_GET_CONTENT);
            intent.setType("image/*");
            intent.putExtra(Intent.EXTRA_LOCAL_ONLY, true);
            startActivityForResult(Intent.createChooser(intent, "Complete
action using"), RC_PHOTO_PICKER);
        }
    });



    // Enable Send button when there's text to send
    mMessageEditText.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence charSequence,
int i, int i1, int i2) {
        }

        @Override
        public void onTextChanged(CharSequence charSequence, int i,
int i1, int i2) {
            if (charSequence.toString().trim().length() > 0) {
                mSendButton.setEnabled(true);
            } else {
                mSendButton.setEnabled(false);
            }
        }

        @Override
        public void afterTextChanged(Editable editable) {
        }
    });


    mMessageEditText.setFilters(new InputFilter[]
            {
            new
InputFilter.LengthFilter(DEFAULT_MSG_LENGTH_LIMIT)
            }
    );

    // Send button sends a message and clears the EditText
    mSendButton.setOnClickListener(new View.OnClickListener() {
        @Override
```

```
            public void onClick(View view) {
                //pushing messages to FireBase dB
                FriendlyMessage friendlyMessage = new
FriendlyMessage(mMessageEditText.getText().toString(), mUsername,
null);
                databaseReference.push().setValue(friendlyMessage);

                // Clear input box
                mMessageEditText.setText("");
            }
        });



        //Authentication processes
        authStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth
firebaseAuth) {  //this firebase auth provides present user status
                FirebaseUser user = firebaseAuth.getCurrentUser();
                if (user != null) {
                    onSignInInitialize(user.getDisplayName());
                }
                else
                {
                    onSignOutCleanUp();
                    startActivityForResult(
                            AuthUI.getInstance()
                                    .createSignInIntentBuilder()
                                    .setProviders(Arrays.asList(new
AuthUI.IdpConfig.Builder(AuthUI.EMAIL_PROVIDER).build(),
                                            new
AuthUI.IdpConfig.Builder(AuthUI.GOOGLE_PROVIDER).build(),
                                            new
AuthUI.IdpConfig.Builder(AuthUI.FACEBOOK_PROVIDER).build(),
                                            new
AuthUI.IdpConfig.Builder(AuthUI.TWITTER_PROVIDER).build()))
                                    .build(),
                            RC_SIGN_IN);  //RC is request code which acts as flag
for signin
                }
            }
```

```
        };

        FirebaseRemoteConfigSettings configSettings = new
FirebaseRemoteConfigSettings.Builder()
                .setDeveloperModeEnabled(BuildConfig.DEBUG)
                .build();

        firebaseRemoteConfig.setConfigSettings(configSettings);

        Map<String,Object> defaultConfigMap = new HashMap<>();

defaultConfigMap.put(FRIENDLY_MESSAGE_LENGTH_KEY,DEFAUL
T_MSG_LENGTH_LIMIT);
        firebaseRemoteConfig.setDefaults(defaultConfigMap);
        fetchConfig();



    }


    //onCreate Closed

 @Override
    //Used to come out of the app to home screen
    public void onActivityResult(int requestCode, int resultCode,Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == RC_SIGN_IN) {
            if (resultCode == RESULT_OK) {
                Toast.makeText(getApplicationContext(), "SignIn Successfull",
Toast.LENGTH_SHORT).show();
            } else if (resultCode == RESULT_CANCELED) {
                Toast.makeText(this, "SingIn Cancelled",
Toast.LENGTH_SHORT).show();
                finish();
            } else if (requestCode == RC_PHOTO_PICKER && resultCode
== RESULT_OK) {
                //For inserting an Image
                Uri selectedImageUri = data.getData();
                StorageReference photoref =
photoStorageReference.child(selectedImageUri.getLastPathSegment());
```

```
photoref.putFile(selectedImageUri).addOnSuccessListener(this, new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot
taskSnapshot) {
                    // Upload succeeded
                    Log.d(TAG, "uploadFromUri:onSuccess");

                    // Get the public download URL
                    @SuppressWarnings("Visible for testing") Uri
mDownloadUrl = taskSnapshot.getDownloadUrl();
                    FriendlyMessage friendlyMessage = new
FriendlyMessage(null, mUsername, mDownloadUrl.toString());
                    databaseReference.push().setValue(friendlyMessage);


                    // [START_EXCLUDE]
                    //   updateUI(mAuth.getCurrentUser());
                    // [END_EXCLUDE]
                }
            })
                    .addOnFailureListener(this, new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception exception) {
                            // Upload failed
                            Log.w(TAG, "uploadFromUri:onFailure", exception);

                            Uri mDownloadUrl = null;
                            Toast.makeText(getApplicationContext(), "Error:
upload failed", Toast.LENGTH_SHORT).show();
                            // updateUI(mAuth.getCurrentUser());
                            // [END_EXCLUDE]
                        }
                    });
        }

    }
}



@Override
public boolean onCreateOptionsMenu(Menu menu) {
```

```
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
//-------------------------------------------------------------------------------------//
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.sign_out_menu:
                AuthUI.getInstance().signOut(this);          //signout process
                return true;

            default:
                return super.onOptionsItemSelected(item);
        }
    }
//-------------------------------------------------------------------------------------//
    protected void onPause()   //called when a call or message iterrupts
current Activity
    {
        super.onPause();
        if(authStateListener != null)
        {
            firebaseAuth.removeAuthStateListener(authStateListener);
        }

        detachReadListener();  //here it does all the clearing activity when
        mMessageAdapter.clear();                              // it is in Pause
    }

    protected void onResume()   //called when u resume the activity
    {
        super.onResume();
        firebaseAuth.addAuthStateListener(authStateListener);
    }
//-------------------------------------------------------------------------------------//
    private void onSignInInitialize(String username) {
        mUsername = username;
        attachReadListener();                  //reading is done only if
authenticated
    }

    private void onSignOutCleanUp() {            //on signout username is
```

```
made anonomous
        mUsername = ANONYMOUS;
        mMessageAdapter.clear();
        detachReadListener();

    }
//--------------------------------------------------------------------------------//
    protected void attachReadListener() {
        if (childEventListener == null) {
            childEventListener = new ChildEventListener() {
                @Override
                public void onChildAdded(DataSnapshot dataSnapshot, String
s) {
                    FriendlyMessage friendlyMessage =
dataSnapshot.getValue(FriendlyMessage.class);
                    mMessageAdapter.add(friendlyMessage);
                    //datasnapshot holds the message that is added
                }
                @Override
                public void onChildChanged(DataSnapshot dataSnapshot,
String s) {}
                @Override
                public void onChildRemoved(DataSnapshot dataSnapshot) {}
                @Override
                public void onChildMoved(DataSnapshot dataSnapshot, String
s) {}
                @Override
                public void onCancelled(DatabaseError databaseError) {}
            };
            databaseReference.addChildEventListener(childEventListener);

        }
    }


    protected void detachReadListener() {
        if (childEventListener != null) {
            databaseReference.removeEventListener(childEventListener);
            childEventListener = null;
        }
    }
//--------------------------------------------------------------------------------//
```

```
//Remoteconfig Functions
    public void fetchConfig()
    {
        long cacheExpiration = 3600;


if(firebaseRemoteConfig.getInfo().getConfigSettings().isDeveloperMode
Enabled())
        {
            cacheExpiration = 0;
        }

        firebaseRemoteConfig.fetch(cacheExpiration)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        firebaseRemoteConfig.activateFetched();
                        applyRetrievedLengthLimit();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Log.w(TAG,"Error Fetching Config",e);
                        applyRetrievedLengthLimit();
                    }
                });
    }

//Alter the message length

    public void applyRetrievedLengthLimit()
    {
        Long friendly_msg_length =
firebaseRemoteConfig.getLong(FRIENDLY_MESSAGE_LENGTH_KEY)
;
        mMessageEditText.setFilters(new InputFilter[]{new
InputFilter.LengthFilter(friendly_msg_length.intValue())});
        Log.d(TAG,FRIENDLY_MESSAGE_LENGTH_KEY + "=" +
friendly_msg_length);}}
//-----------------------------------------------------------------------------------//
```

FriendlyMessage.java

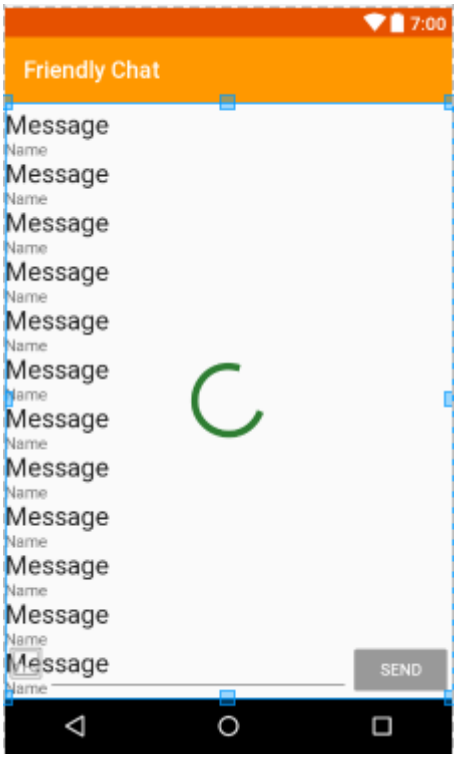
```
package com.example.android.friendlychat;

public class FriendlyMessage {

    private String text;
    private String name;
    private String photoUrl;

    public FriendlyMessage() {
    }

    public FriendlyMessage(String text, String name, String photoUrl) {
        this.text = text;
        this.name = name;
        this.photoUrl = photoUrl;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPhotoUrl() {
        return photoUrl;
    }

    public void setPhotoUrl(String photoUrl) {
        this.photoUrl = photoUrl;
    }
}
```

MessageAdapter.java

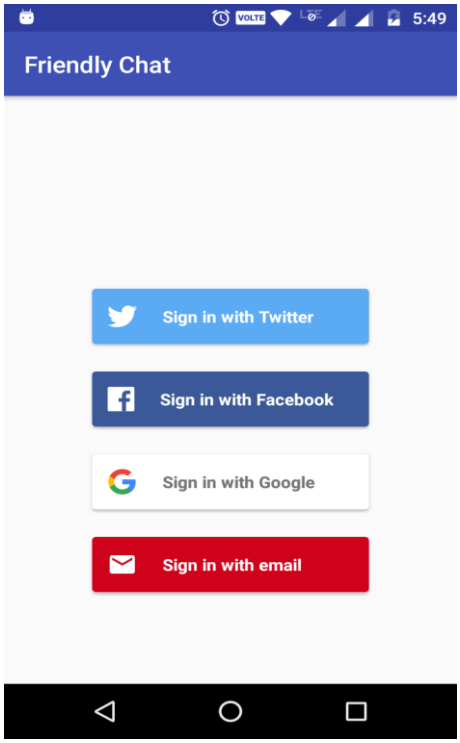
```
package com.google.firebase.udacity.friendlychat;

import android.app.Activity;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.bumptech.glide.Glide;

import java.util.List;

public class MessageAdapter extends ArrayAdapter<FriendlyMessage>
{
    public MessageAdapter(Context context, int resource,
List<FriendlyMessage> objects) {
        super(context, resource, objects);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {
        if (convertView == null) {
            convertView = ((Activity)
getContext()).getLayoutInflater().inflate(R.layout.item_message, parent,
false);
        }

        ImageView photoImageView = (ImageView)
convertView.findViewById(R.id.photoImageView);
        TextView messageTextView = (TextView)
convertView.findViewById(R.id.messageTextView);
        TextView authorTextView = (TextView)
convertView.findViewById(R.id.nameTextView);

        FriendlyMessage message = getItem(position);

        boolean isPhoto = message.getPhotoUrl() != null;
        if (isPhoto) {
```

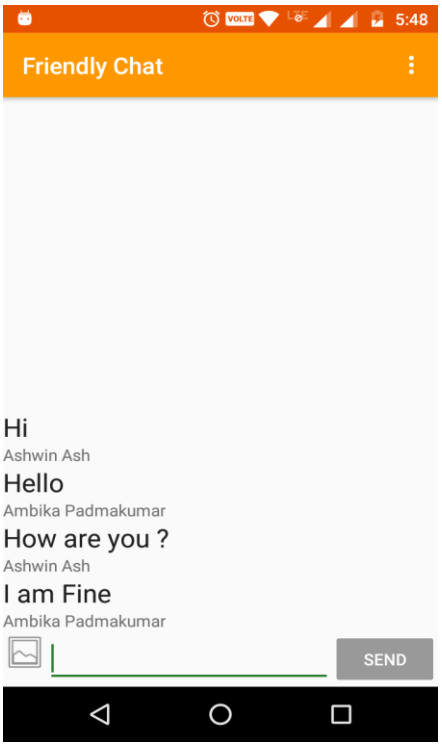
```
            messageTextView.setVisibility(View.GONE);  //View.GONE does
not occupy any space if invisible .
            photoImageView.setVisibility(View.VISIBLE);
            Glide.with(photoImageView.getContext())
                    .load(message.getPhotoUrl())
                    .into(photoImageView);
        } else {
            messageTextView.setVisibility(View.VISIBLE);
            photoImageView.setVisibility(View.GONE);
            messageTextView.setText(message.getText());
        }
        authorTextView.setText(message.getName());

        return convertView;
    }
}
```

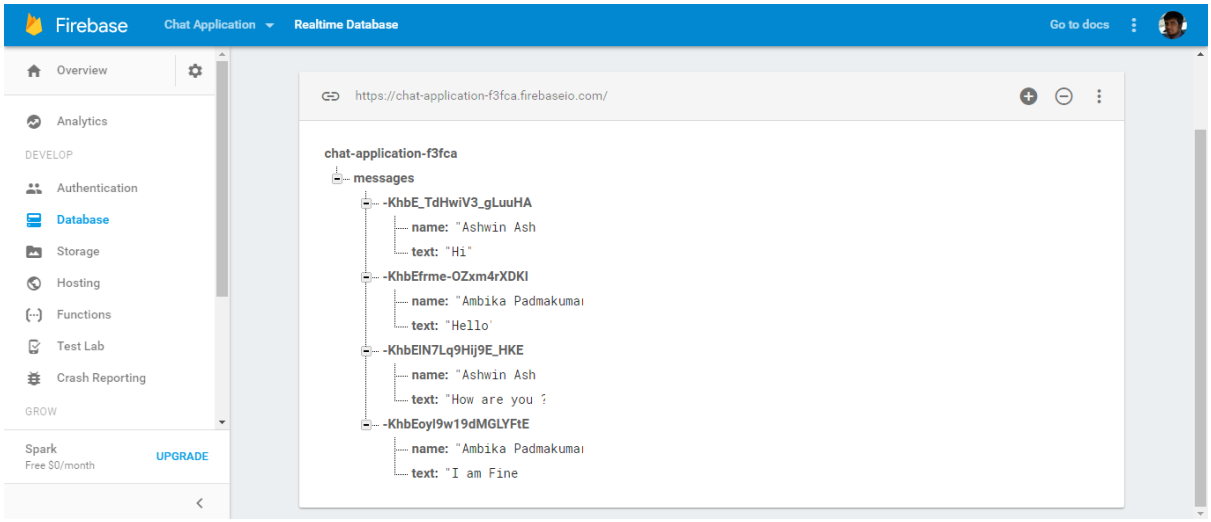
Output:


7:00
Friendly Chat
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
Message
Name
SEND


7:00
Friendly Chat
Message
Name


5:49
Friendly Chat
Sign in with Twitter
Sign in with Facebook
Sign in with Google
Sign in with email


5:48
Friendly Chat
Hi
Ashwin Ash
Hello
Ambika Padmakumar
How are you ?
Ashwin Ash
I am Fine
Ambika Padmakumar
SEND



Conclusion:

Hence an Android Chat Application is developed using Firebase as the backend which can be used in realtime to communicate with people by sending text messages .