

# **ZENCHAT**

(A mobile-based chatting platform)

A Mini-Project Report

Submitted for partial fulfillment of the requirement for the award of the degree of

## **BACHELOR OF COMPUTER APPLICATIONS**

Submitted by

**KARTHIK S (A21CADA27)**

**MOHAMED BASHEER Y (A21CADA34)**

**MURALI D (A21CADA39)**

Under the Guidance of

**Dr. I. BENJAMIN FRANKLIN M.C.A., M.Phil., Ph.D.,**

Assistant Professor, PG Department of Computer Applications

**PG DEPARTMENT OF COMPUTER APPLICATIONS**



**ST. JOSEPH'S COLLEGE OF ARTS & SCIENCE (AUTONOMOUS)**

(Affiliated to Annamalai University, Annamalai Nagar)

**CUDDALORE – 607001**

**APRIL - 2024**

# **CERTIFICATE**

This is to certify that the Mini Project entitled

## **ZENCHAT**

(A mobile-based chatting platform)

Being submitted to the

**ST. JOSEPH'S COLLEGE OF ARTS & SCIENCE (AUTONOMOUS)**

(Affiliated to Annamalai university, Annamalai Nagar)

Submitted by

**KARTHIK S (A21CADA27)**

**MOHAMED BASHEER Y (A21CADA34)**

**MURALI D (A21CADA39)**

For the partial fulfilment for the award of degree of

**BACHELOR OF COMPUTER APPLICATIONS**

is a Bonafide record of work carried out by them, under my guidance and supervision.

**INTERNAL GUIDE**

**HEAD OF THE DEPARTMENT**

Submitted for the viva-voce examination held on \_\_\_\_\_

**Examiners:**

1. \_\_\_\_\_

2. \_\_\_\_\_

## **ACKNOWLEDGEMENT**

It is our earnest and sincere desire and ambition to acquire profound knowledge in the study of Computer Applications. We are grateful to God, the Almighty who has blessed us abundantly and guide us to complete this work.

It is our pleasure to acknowledge the help and guidance that we had received from different personal and to thank them individually.

We grab this opportunity to thank **Rev. Fr. Dr. M. SWAMINATHAN**, Secretary, and **Dr. M. ARUMAI SELVAM**, Principal, St. Joseph's College of Arts & Science (Autonomous), Cuddalore for giving us an opportunity for submitting this project.

We take immense pleasure in conveying our sincere and heartfelt gratitude to **Dr. A. JOHN PRADEEP EBENEZER M.C.A., M.Phil., Ph.D.**, Head of the PG Department of Computer Applications, for extending the laboratory facilities and moral support.

We are greatly indented and truly feel privileged to extend our thanks to our guide **Dr. I. BENJAMIN FRANKLIN M.C.A., M.Phil., Ph.D.**, Assistant Professor, PG Department of Computer Applications for the able guidance and constant encouragement during the development of our project work.

We express our sincere thanks to our Parents for constantly giving us words of courage and motivation to complete this project successfully. We would like to thank Our Friends for their guidance and assistance in the completion of the project.

**KARTHIK S (A21CADA27)**

**MOHAMED BASHEER Y (A21CADA34)**

**MURALI D (A21CADA39)**

## CONTENT

S.NO	DESCRIPTION	PAGE NO
1	ABSTRACT	1
2	HARDWARE AND SOFTWARE REQUIREMENT	2
3	SOFTWARE DESCRIPTION	3
4	PROJECT DESCRIPTION	8
5	SOURCE CODE	10
6	OUTPUT SCEEN	65
7	CONCLUSION	76
8	FUTURE ENHANCEMENT	77
9	BIBLIOGRAPHY	78
10	REFERENCE	79

## **ABSTRACT**

ZenChat is a chat app that makes talking with friends and family, or even getting answers from a chatbot, easy and fun. Built using Java and Firebase, it lets users send messages, share photos, set their status, talk over voice calls, and chat with an intelligent ChatGPT bot. The app uses Firebase for storing messages and photos, making sure that everything users do is updated in real-time and securely saved.

Creating ZenChat was all about giving users a smooth and enjoyable chatting experience. With voice calls, they can talk to each other directly through the app. The ChatGPT chatbot feature makes ZenChat even more special, providing users with a smart companion to chat with anytime. Whether sharing important moments with the status feature or just having a regular text chat, ZenChat is designed to be straightforward and fun for everyone.

Our goal with ZenChat was to mix the simple with the smart, providing an app where people can easily connect, share, and enjoy conversations, all in one place. Testing ZenChat showed that users really liked how easy it was to use and were excited about chatting with the AI bot. This feedback was a big win for us, showing that combining basic chatting features with advanced ones like ChatGPT can really enhance the chatting experience.

In the end, ZenChat is more than just a chat app; it's a new way for people to communicate and share, proving that even the simplest apps can be powerful and fun when they bring something new to the table.

# **HARDWARE AND SOFTWARE REQUIREMENT**

## **HARDWARE REQUIREMENTS**

The hardware requirements are the hardware, which is essential for software to run on a particular system.

- SYSTEM : INTEL(R) CORE(TM) 1.60GHZ
- RAM : 8 GB
- INTERNAL STORAGE : 256 GB .

## **SOFTWARE REQUIREMENTS**

The Software Requirements deal with the analysis of the software, operating system and tools that are essential for the project.

- OPERATING SYSTEM : Android 8
- SOFTWARE: : ANDROID STUDIO
- PROGRAMMING LANGUAGE : JAVA
- BACK END : FIREBASE

# SOFTWARE DESCRIPTION

## XML:

XML (Extensible Markup Language) plays a pivotal role, primarily in defining user interfaces (UI) and configuring app resources. XML offers a structured, hierarchical way to describe data, making it an ideal choice for the layout and resource management in Android apps. Here's an overview of how XML is used in Android development:

### 1. Defining UI Layouts:

**Hierarchical Structure:** XML files are used to create a tree-like structure that mirrors the layout of UI elements on the screen. This structure makes it easy to understand how different elements relate to each other.

**Declarative Syntax:** Developers describe the UI elements (like buttons, text views, and image views) and their properties (such as size, position, and color) in XML files. The Android system then reads these files and renders the UI elements on the screen accordingly.

**Separation of Concerns:** Using XML for UI layouts helps in separating the app's design from the business logic, which is typically written in Java or Kotlin. This separation makes the code cleaner, easier to manage, and more maintainable.

### 2. Managing App Resources:

**Drawable Resources:** XML is used to define shapes, selectors, and layer lists that can be used as backgrounds or icons in the app, offering a flexible way to style UI components.

**Animation Resources:** Android allows for defining animations (such as fade-in, move, rotate) in XML files, which can be applied to UI elements programmatically.

**Value Resources:** XML files within the res/values directory hold simple values in a key-value pair format. These include strings, dimensions, colors, and style definitions. Storing these values in XML files makes them easy to reference and manage, as well as simplifies the process of internationalizing the app.

### 3. Configuration Files:

**Manifest File:** The `AndroidManifest.xml` is a crucial file that contains essential information about the app, such as the app's package name, components (activities, services, broadcast receivers, content providers), permissions, and hardware and software features the app requires.

**Gradle Files for Dependencies:** While not XML, it's worth mentioning that `build.gradle` files, though part of Gradle's domain-specific language, play a similar role in managing app dependencies and SDK versions, akin to how XML manages resources and layouts.

### Advantages of Using XML in Android:

**Readability and Ease of Use:** XML's verbose nature makes it easy to read and understand, even for those new to Android development.

**Tool Support:** Android Studio, the official IDE for Android development, provides extensive support for editing and previewing XML files, making UI design more intuitive through a drag-and-drop interface.

**Scalability:** XML files can be easily edited and managed, allowing for scalable app development. Resources like dimensions and strings can be varied across different device configurations (like screen sizes and languages) without altering the main codebase.

In summary, XML is an integral part of Android app development, facilitating the creation of UI layouts, the management of app resources, and the configuration of app settings, thereby supporting a structured, efficient, and scalable development process.

### JAVA:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. It provides a comprehensive suite of tools for building Android apps, including a code editor, debugger, and emulator. Java, one of the most popular programming languages worldwide, has been a cornerstone of Android app development since the platform's inception. This overview explores the role of Java in



Android Studio, highlighting its significance, benefits, and how it integrates within the Android app development ecosystem.

Java has been the preferred language for Android app development due to its portability, robustness, and an extensive collection of open-source libraries and frameworks. Its object-oriented programming model aligns well with the architecture of Android apps, facilitating the development of modular, scalable, and maintainable applications. The Android SDK (Software Development Kit) itself is largely written in Java, offering a rich set of APIs for developers to build feature-rich apps.

## **Java in Android Studio: Key Features**

**Integrated Development Environment:** Android Studio provides a comprehensive environment for Java development, including code completion, refactoring, and analysis tools tailored for Android development.

**Android SDK Integration:** Seamless integration with the Android SDK allows developers to access a vast library of Java APIs for building complex apps that can leverage device hardware, like cameras and sensors, and perform background tasks.

**Emulator and Device Testing:** Developers can test their Java-based Android applications on a wide range of virtual devices or real hardware directly from Android Studio, facilitating a smooth development and testing workflow.

**Gradle-Based Build System:** Android Studio uses Gradle for its build system, allowing developers to manage dependencies, define custom build configurations, and automate the build process for their Java applications.

**Debugging and Profiling Tools:** The IDE offers powerful debugging and profiling tools, enabling developers to efficiently track down issues in their Java code and optimize app performance.

## **Benefits of Using Java in Android Studio**

**Mature Language:** Java's long history and widespread use have led to a vast ecosystem of libraries and frameworks, which can significantly speed up Android app development.

**Cross-Platform Compatibility:** Java applications can run on any device that supports the Java Virtual Machine (JVM), making it easier to extend Android applications to other platforms.

**Community Support:** A large and active developer community provides extensive resources, documentation, and support for Java developers.

## **FIREBASE:**

Cloud Functions for Firebase is a serverless framework that lets you automatically run backend code in response to events triggered by Firebase features and HTTPS requests. Your JavaScript or TypeScript code is stored in Google's cloud and runs in a managed environment. There's no need to manage and scale your own servers.

Firebase developers can easily integrate with external services for tasks like processing payments and sending SMS messages. Also, developers can include custom logic that is either too heavyweight for a mobile device, or which needs to be secured on a server.

Cloud Functions for Firebase is optimized for Firebase developers:

- Firebase SDK to configure your functions through code.
- Integrated with Firebase Console and Firebase CLI.
- The same triggers as Google Cloud Functions, plus Firebase Realtime Database, Firebase Authentication, and Firebase Analytics triggers.

Build powerful apps. Spin up your backend without managing servers. Effortlessly scale to support millions of users with Firebase databases, machine learning infrastructure, hosting and storage solutions, and Cloud Functions.

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through Realtime listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity.

For developers that need a more full-featured backend, Cloud Functions provides a gateway to the powerful capabilities in Google Cloud Platform.

### **Flexibility:**

The Cloud Firestore data model supports flexible, hierarchical data structures. Store your data in documents, organized into collections. Documents can contain complex nested objects in addition to subcollections.

### **Expressive querying:**

In Cloud Firestore, you can use queries to retrieve individual, specific documents or to retrieve all the documents in a collection that match your query parameters. Your queries can include multiple, chained filters and combine filtering and sorting. They're also indexed by default, so query performance is proportional to the size of your result set, not your data set.

### **Realtime updates:**

Like Realtime Database, Cloud Firestore uses data synchronization to update data on any connected device. However, it's also designed to make simple, one-time fetch queries efficiently.

### **Offline support:**

Cloud Firestore caches data that your app is actively using, so the app can write, read, listen to, and query data even if the device is offline. When the device comes back online, Cloud Firestore synchronizes any local changes back to Cloud Firestore.

### **Designed to scale:**

Cloud Firestore brings you the best of Google Cloud's powerful infrastructure: automatic multi-region data replication, strong consistency guarantees, atomic batch operations, and real transaction support. We've designed Cloud Firestore to handle the toughest database workloads from the world's biggest apps.

## **PROJECT DESCRIPTION**

### **PROBLEM DEFINITION:**

Email, newsgroup and messaging applications provide means for communication among people, but these are one-way mechanisms and they do not provide an easy way to carry on a real-time conversation or discussion with people involved. Chat room extends the one-way messaging concept to accommodate multi-way communication among a set of people.

### **MODULE:**

#### **User Authentication Module:**

This module manages how users sign into your app. It includes features for creating a new account, logging in, and logging out. It might also offer password recovery and support for signing in through social media accounts to make access easy and secure.

#### **Profile Management Module:**

After signing up, users can personalize their profiles within this module. It allows them to add or change their profile picture, set a username, update their status message, and manage privacy settings. It's all about letting users express themselves and manage their personal information.

#### **Contacts Module:**

This module helps users connect with others. Users can search for friends by name or email, send friend requests, accept invitations, and organize their contacts. It's essentially the app's address book, keeping everyone a user wants to chat with just a few taps away.

#### **Chat Module:**

At the heart of ZenChat, this module enables users to send and receive

messages. It supports one-on-one chats, group conversations, and the sharing of photos, videos, and other files. Users can also see when messages are delivered and read, ensuring no message gets lost in the digital ether.

### **Chatbot Integration:**

This feature introduces an AI-powered assistant into the chat environment. Whether it's for answering frequently asked questions, providing support, or just entertaining users with funny responses, the chatbot is there to enhance the user experience without needing human intervention.

### **Notifications Module:**

Notifications keep users informed about new messages, friend requests, or any other important activity. This module ensures that users don't miss out on anything important by alerting them through push notifications or in-app alerts, even when they're not actively using the app.

### **Status Module:**

Users can share temporary updates about their day or post content that disappears after 24 hours. It's a fun way to let friends know what you're up to or share moments without adding them to your permanent profile. Users can view, react to, and reply to the statuses of their contacts.

### **UI/UX Module:**

This module is all about the look and feel of the app. It focuses on designing a user-friendly interface, ensuring the app is easy to navigate, visually appealing, and provides a seamless experience. Good UI/UX design makes the app not just usable but enjoyable.

Each of these modules works together to create a comprehensive and engaging user experience, making ZenChat a versatile and user-friendly chatting app.

## SOURCE CODE

### SplashScreenActivity.java

```
package com.basheer.whatsappclone.activities.startup;
import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.basheer.whatsappclone.R;
import com.basheer.whatsappclone.activities.main.MainActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

@SuppressLint("CustomSplashScreen")
public class SplashScreenActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_splash_screen);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -
> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
        FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        if(firebaseUser!=null) {
```

```

        new Handler().postDelayed() -> {
            startActivity(new Intent(SplashScreenActivity.this, MainActivity.class));
            finish();
        }, 2000);
    } else {
        new Handler().postDelayed() -> {
            startActivity(new Intent(SplashScreenActivity.this,
WelcomeScreenActivity.class));
            finish();
        }, 2000);
    }
}
}
}

```

### **WelcomeScreenActivity.java**

```

package com.basheer.whatsappclone.activities.startup;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import com.basheer.whatsappclone.R;
import com.basheer.whatsappclone.activities.auth.PhoneNumberActivity;
import com.basheer.whatsappclone.databinding.ActivityWelcomeScreenBinding;

public class WelcomeScreenActivity extends AppCompatActivity {
    ActivityWelcomeScreenBinding binding;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityWelcomeScreenBinding.inflate(getLayoutInflater());
    }
}

```

```

        setContentView(binding.getRoot());

        binding.agreeButton.setOnClickListener(v -> startActivity(new
        Intent(WelcomeScreenActivity.this, PhoneNumberActivity.class)));
    }}

```

### **PhoneNumberActivity.java**

```

package com.basheer.whatsappclone.activities.auth;

import android.content.Intent;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import com.basheer.whatsappclone.databinding.ActivityPhoneNumberBinding;
import com.google.firebase.auth.FirebaseAuth;
import com.hbb20.CountryCodePicker;

public class PhoneNumberActivity extends AppCompatActivity {
    ActivityPhoneNumberBinding binding;
    FirebaseAuth auth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityPhoneNumberBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        auth = FirebaseAuth.getInstance();
        CountryCodePicker ccp = binding.ccp;
        binding.mobileNumber.requestFocus();
        binding.generateOTP.setOnClickListener(v -> {
            if (!binding.mobileNumber.getText().toString().trim().isEmpty()) {
                String code = ccp.getSelectedCountryCode();
                String number = "+" + code + binding.mobileNumber.getText().toString();
                Intent intent = new Intent(PhoneNumberActivity.this, OTPActivity.class);
                intent.putExtra("phoneNumber", number);
                startActivity(intent);
            }
        });
    }
}

```

### **OTPActivity.java**

```

package com.basheer.whatsappclone.activities.auth;

import android.app.ProgressDialog;
import android.content.Context;

```



```

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import com.basheer.whatsappclone.activities.profile.SetupProfileActivity;
import com.basheer.whatsappclone.databinding.ActivityOtpactivityBinding;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import java.util.concurrent.TimeUnit;

```

```

public class OTPActivity extends AppCompatActivity {

```

```

    ActivityOtpactivityBinding binding;

```

```

    FirebaseAuth auth;

```

```

    String verificationId;

```

```

    ProgressDialog dialog;

```

```

    @Override

```

```

    protected void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);

```

```

        binding = ActivityOtpactivityBinding.inflate(getLayoutInflater());

```

```

        setContentView(binding.getRoot());

```

```

        dialog = new ProgressDialog(this);

```

```

        dialog.setMessage("Sending OTP...");

```

```

        dialog.setCancelable(false);

```

```

        dialog.show();

```

```

        auth = FirebaseAuth.getInstance();

```

```

        String phoneNumber = getIntent().getStringExtra("phoneNumber");

```

```

        binding.mobileNumberText.setText(phoneNumber);

```

```

    assert phoneNumber != null;
    PhoneAuthOptions options = PhoneAuthOptions.newBuilder(auth)
        .setPhoneNumber(phoneNumber)
        .setTimeout(30L, TimeUnit.SECONDS)
        .setActivity(OTPActivity.this)
        .setCallbacks(new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
            @Override
            public void onVerificationCompleted(@NonNull PhoneAuthCredential
phoneAuthCredential) {
                }
            @Override
            public void onVerificationFailed(@NonNull FirebaseException e) {
                }
            @Override
            public void onCodeSent(@NonNull String verifyId, @NonNull
PhoneAuthProvider.ForceResendingToken forceResendingToken) {
                super.onCodeSent(verifyId, forceResendingToken);
                dialog.dismiss();
                verificationId = verifyId;
                InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
                imm.toggleSoftInput(InputMethodManager.SHOW_FORCED, 0);
                binding.otpview.requestFocus();
            }
        }).build();
    PhoneAuthProvider.verifyPhoneNumber(options);
    binding.resendOTP.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            PhoneAuthOptions options = PhoneAuthOptions.newBuilder(auth)
                .setPhoneNumber(phoneNumber)
                .setTimeout(30L, TimeUnit.SECONDS)
                .setActivity(OTPActivity.this)
                .setCallbacks(new

```

```

PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
    @Override
    public void onVerificationCompleted(@NonNull PhoneAuthCredential
phoneAuthCredential) {
        }
    @Override
    public void onVerificationFailed(@NonNull FirebaseException e) {
        }
    @Override
    public void onCodeSent(@NonNull String verifyId, @NonNull
PhoneAuthProvider.ForceResendingToken forceResendingToken) {
        super.onCodeSent(verifyId, forceResendingToken);
        verificationId = verifyId;
        InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.toggleSoftInput(InputMethodManager.SHOW_FORCED, 0);
        binding.otpview.requestFocus();
    }
    }).build();
    PhoneAuthProvider.verifyPhoneNumber(options);
}
});
binding.otpview.setOtpCompletionListener(otp -> {
    PhoneAuthCredential credential = PhoneAuthProvider.getCredential(verificationId,
otp);
    auth.signInWithCredential(credential).addOnCompleteListener(task -> {
        if(task.isSuccessful()) {
            Intent intent = new Intent(OTPActivity.this, SetupProfileActivity.class);
            startActivity(intent);
            finishAffinity();
        } else {
            Toast.makeText(OTPActivity.this, "Wrong OTP",
Toast.LENGTH_SHORT).show();}
    });
});

```

```
});  
}}
```

## SetupProfileActivity.java

```
package com.basheer.whatsappclone.activities.profile;  
import static com.google.common.io.Files.getFileExtension;  
import android.annotation.SuppressLint;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Bundle;  
import android.os.Environment;  
import android.provider.MediaStore;  
import android.text.TextUtils;  
import android.util.Log;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.Toast;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.content.FileProvider;  
import com.basheer.whatsappclone.R;  
import com.basheer.whatsappclone.databinding.ActivitySetupProfileBinding;  
import com.basheer.whatsappclone.activities.main.MainActivity;  
import com.bumptech.glide.Glide;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.android.material.bottomsheet.BottomSheetDialog;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.firestore.FirebaseFirestore;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
import com.google.firebase.storage.UploadTask;  
import java.io.File;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Locale;  
import java.util.Objects;  
  
public class SetupProfileActivity extends AppCompatActivity {  
  
    ActivitySetupProfileBinding binding;  
    ProgressDialog progressDialog;
```

```

        String TAG = "SetupProfileActivity";

        FirebaseUser firebaseUser;

        FirebaseFirestore firestore;

        BottomSheetDialog bottomSheetDialog;

        Uri imageUri;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            binding = ActivitySetupProfileBinding.inflate(getLayoutInflater());
            setContentView(binding.getRoot());
            firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
            firestore = FirebaseFirestore.getInstance();
            firestore.collection("Users").document(firebaseUser.getId())
                .get()
                .addOnCompleteListener(task -> {
                    if (task.isSuccessful()) {
                        binding.name.setText(task.getResult().getString("userName"));
                        Glide.with(SetupProfileActivity.this).load(task.getResult().getString("imageProfile")).placeholder(R.drawable.avatar).into(binding.profileImage);
                    } else {
                        Log.w(TAG, "Error getting documents.", task.getException());
                    }
                });
            progressDialog = new ProgressDialog(this);
            progressDialog.setMessage("Updating Profile..");
            progressDialog.setCancelable(false);
            initButtonClick();
        }
        private void initButtonClick() {
            binding.setupProfile.setOnClickListener(v -> {
                if(TextUtils.isEmpty(binding.name.getText().toString())) {
                    Toast.makeText(SetupProfileActivity.this, "Please enter your name",
                        Toast.LENGTH_SHORT).show();
                } else {
                    uploadToFirebaseStorage();
                }
            });
            binding.floating.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    showBottomSheetPickPhoto();
                }
            });
        }
        private void showBottomSheetPickPhoto() {
            @SuppressWarnings("InflateParams") View view =
            getLayoutInflater().inflate(R.layout.bottom_sheet_dialog, null);
            ((View) view.findViewById(R.id.gallery)).setOnClickListener(new

```

```

View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openGalery();
        bottomSheetDialog.dismiss();
    }
});
((View) view.findViewById(R.id.camera)).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openCamera();
        bottomSheetDialog.dismiss();
    }
});
bottomSheetDialog = new BottomSheetDialog(this);
bottomSheetDialog.setContentView(view);
Objects.requireNonNull(bottomSheetDialog.getWindow()).addFlags(WindowManager.Layo
utParams.FLAG_TRANSLUCENT_STATUS);
bottomSheetDialog.setOnDismissListener(new DialogInterface.OnDismissListener() {
    @Override
    public void onDismiss(DialogInterface dialog) {
        bottomSheetDialog = null;
    }
});
bottomSheetDialog.show();
}
@SuppressWarnings("QueryPermissionsNeeded")
private void openCamera() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    String timeStamp = new SimpleDateFormat("yyyMMdd_HH:mm:ss",
Locale.getDefault()).format(new Date());
    String imageFileName = "IMG_" + timeStamp + ".jpg";
    try {
        File file = File.createTempFile("IMG_" + timeStamp, ".jpg",
getExternalFilesDir(Environment.DIRECTORY_PICTURES));
        imageUri = FileProvider.getUriForFile(this, "com.basheer.whatsappclone.provider",
file);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
        intent.putExtra("listPhotoName", imageFileName);
        startActivityForResult(intent, 440);
    } catch (Exception ignored) {
    }
}

private void openGalery() {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");

```

```

        startActivityForResult(intent, 45);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(data != null) {
            if (data.getData() != null) {
                imageUri = data.getData();
                Glide.with(SetupProfileActivity.this).load(imageUri).placeholder(R.drawable.avatar).into(binding.profileImage);
            }
        }
        if (requestCode == 440 && resultCode == RESULT_OK) {
            Glide.with(SetupProfileActivity.this).load(imageUri).placeholder(R.drawable.avatar).into(binding.profileImage);
        }
    }

    private void uploadToFirebaseStorage() {
        if(imageUri!=null) {
            progressDialog.show();
            StorageReference storageReference =
                FirebaseStorage.getInstance().getReference().child("ProfileImages/" +
                System.currentTimeMillis() + "." + getFileExtension(String.valueOf(imageUri)));
            storageReference.putFile(imageUri).addOnSuccessListener(new
                OnSuccessListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                        Task<Uri> urlTask = taskSnapshot.getStorage().getDownloadUrl();
                        while (!urlTask.isSuccessful());
                        Uri downloadUrl = urlTask.getResult();
                        final String sdownload_url = String.valueOf(downloadUrl);
                        HashMap<String, Object> hashMap = new HashMap<>();
                        hashMap.put("imageProfile", sdownload_url);

```

```

        hashMap.put("userName", binding.name.getText().toString());
        progressDialog.dismiss();
        firestore.collection("Users").document(firebaseUser.getUid()).update(hashMap).addOnSuccess
        ssListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(SetupProfileActivity.this, "Upload successfully",
                Toast.LENGTH_SHORT).show();
                Intent intent = new Intent(SetupProfileActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }
    });
} else {
    updateProfile();
}
}

private void updateProfile() {
    HashMap<String, Object> hashMap = new HashMap<>();
    hashMap.put("userName", binding.name.getText().toString());
    firestore.collection("Users").document(firebaseUser.getUid()).update(hashMap).addOnSuccess
    ssListener(aVoid -> {
        Toast.makeText(SetupProfileActivity.this, "Profile updated successfully",
        Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(SetupProfileActivity.this, MainActivity.class);
        startActivity(intent);
        finish();
    }).addOnCompleteListener(task -> progressDialog.dismiss()).addOnFailureListener(e ->
{
    progressDialog.dismiss();
    Toast.makeText(SetupProfileActivity.this, "Failed to update profile: " +

```



```
e.getMessage(), Toast.LENGTH_SHORT).show();
    });
}
}
```

## **MainActivity.java**

```
package com.basheer.whatsappclone.activities.main;
import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.view.GestureDetector;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.ImageButton;
import android.widget.Toast;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.Toolbar;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.viewpager.widget.ViewPager;
import com.basheer.whatsappclone.R;
import com.basheer.whatsappclone.activities.chat.ContactsActivity;
import com.basheer.whatsappclone.activities.chatbot.ChatBotActivity;
import com.basheer.whatsappclone.databinding.ActivityMainBinding;
import com.basheer.whatsappclone.menu.CallsFragment;
import com.basheer.whatsappclone.menu.ChatsFragment;
```

```

import com.basheer.whatsappclone.menu.StatusFragment;
import com.basheer.whatsappclone.activities.settings.SettingsActivity;
import com.basheer.whatsappclone.model.Status;
import com.basheer.whatsappclone.model.UserStatus;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.messaging.FirebaseMessaging;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;

```

```

public class MainActivity extends AppCompatActivity {
    ActivityMainBinding binding;
    private int currentFabAction = -1;
    DatabaseReference reference;
    FirebaseUser firebaseUser;
    FirebaseDatabase database = FirebaseDatabase.getInstance();
    String senderID, name, imageUrl;
    private GestureDetector gestureDetector;
    private float xCoOrdinate, yCoOrdinate;
    FirebaseFirestore firestore;
    ProgressDialog dialog;
    private boolean isMoving = false;

```

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    binding = ActivityMainBinding.inflate(getLayoutInflater());  
    setContentView(binding.getRoot());  
    reference = FirebaseDatabase.getInstance().getReference();  
    firestore = FirebaseFirestore.getInstance();  
    firebaseUser = FirebaseAuth.getInstance().getCurrentUser();  
    dialog = new ProgressDialog(this);  
    dialog.setMessage("Uploading Image ...");  
    dialog.setCancelable(false);  
    setUpWithViewPager(binding.viewPager);  
    binding.tabLayout.setupWithViewPager(binding.viewPager);  
    binding.viewPager.setCurrentItem(0);  
    changeFabIcon(0);  
    binding.viewPager.addOnPageChangeListener(new  
ViewPager.OnPageChangeListener() {  
        @Override  
        public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels)  
{  
            }  
        @Override  
        public void onPageSelected(int position) {  
            changeFabIcon(position);  
        }  
        @Override  
        public void onPageScrollStateChanged(int state) {  
            }  
    });  
    FirebaseMessaging.getInstance()  
        .getToken()  
        .addOnSuccessListener(new OnSuccessListener<String>() {  
            @Override  
            public void onSuccess(String token) {
```

```

        HashMap<String, Object> map = new HashMap<>();
        map.put("token", token);
        firestore.collection("Users").document(firebaseUser.getId()).update(map);
        //Toast.makeText(MainActivity.this, token, Toast.LENGTH_SHORT).show();
    }
});

Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);

ImageButton floatingImageButton = findViewById(R.id.chatbot);
gestureDetector = new GestureDetector(this, new
GestureDetector.SimpleOnGestureListener() {
    @Override
    public void onLongPress(MotionEvent e) {
        super.onLongPress(e);
        isMoving = true; // Enable movement on long press
    }
    @Override
    public boolean onDown(MotionEvent e) {
        return true; // Necessary to detect the following gestures
    }
});

floatingImageButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (!isMoving) { // Prevents launching the activity if the button was moved
            Intent intent = new Intent(MainActivity.this, ChatBotActivity.class);
            startActivity(intent);
        }
    }
});

floatingImageButton.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View view, MotionEvent event) {
        gestureDetector.onTouchEvent(event); // Pass the event to the GestureDetector

```

```

        if (!isMoving) {
            return false; // If not in moving mode, let the click event be handled
        }
        switch (event.getActionMasked()) {
            case MotionEvent.ACTION_DOWN:
                xCoOrdinate = view.getX() - event.getRawX();
                yCoOrdinate = view.getY() - event.getRawY();
                break;
            case MotionEvent.ACTION_MOVE:
                view.animate()
                    .x(event.getRawX() + xCoOrdinate)
                    .y(event.getRawY() + yCoOrdinate)
                    .setDuration(0)
                    .start();
                break;
            case MotionEvent.ACTION_UP:
            case MotionEvent.ACTION_CANCEL:
                isMoving = false; // Disable movement mode when the touch is released
                break;
        }
        return true;
    }
});
}

public void setUpWithViewPager(ViewPager viewPager) {
    MainActivity.SectionsPagerAdapter adapter = new
SectionsPagerAdapter(getSupportFragmentManager());
    adapter.addFragment(new ChatsFragment(), "chats");
    adapter.addFragment(new StatusFragment(), "status");
    adapter.addFragment(new CallsFragment(), "calls");
    viewPager.setAdapter(adapter);
}

public static class SectionsPagerAdapter extends FragmentPagerAdapter {
    private final List<Fragment> mfragmentList = new ArrayList<>();

```

```

private final List<String> mfragmentTitleList = new ArrayList<>();

    public SectionsPagerAdapter(FragmentManager manager) {
        super(manager);
    }

    @NonNull

    public Fragment getItem(int position) {return mfragmentList.get(position);}

    @Override

    public int getCount() {return mfragmentList.size();}

    public void addFragment(Fragment fragment, String title) {
        mfragmentList.add(fragment);
        mfragmentTitleList.add(title);
    }

    public CharSequence getPageTitle(int position) {return
mfragmentTitleList.get(position);}
    }

    private void changeFabIcon(final int index) {
        binding.fabTab.setVisibility(View.GONE);
        binding.editTab.setVisibility(View.GONE);
        currentFabAction = index;
        new Handler().postDelayed(new Runnable() {
            @SuppressWarnings("UseCompatLoadingForDrawables")
            @Override
            public void run() {
                switch (index) {
                    case 0:
                        binding.fabTab.setVisibility(View.VISIBLE);
                        binding.editTab.setVisibility(View.GONE);
                        binding.fabTab.setImageDrawable(getDrawable(R.drawable.chat_new_img));
                        break;
                    case 1:
                        binding.fabTab.setVisibility(View.VISIBLE);
                        binding.editTab.setVisibility(View.VISIBLE);
                        binding.fabTab.setImageDrawable(getDrawable(R.drawable.camera_icon));
                        break;
                }
            }
        }, 100);
    }
}

```

```

        case 2:
            binding.fabTab.setVisibility(View.GONE);
            break;
        }
    }
    }, 400);
    onPerformClick();
}

private void onPerformClick() {
    binding.fabTab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch (currentFabAction) {
                case 0:
                    startActivity(new Intent(MainActivity.this, ContactsActivity.class));
                    break;
                case 1:
                    Intent intent = new Intent();
                    intent.setType("image/*");
                    intent.setAction(Intent.ACTION_GET_CONTENT);
                    startActivityForResult(intent, 75);
                    break;
                case 2:
                    Toast.makeText(MainActivity.this, "Call page",
Toast.LENGTH_SHORT).show();
            }
        }
    });
    binding.editTab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(MainActivity.this, "Edit Status",
Toast.LENGTH_SHORT).show();
        }
    });
}

```

```

        });
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int itemId = item.getItemId();
        if (itemId == R.id.action_settings) {
            startActivity(new Intent(MainActivity.this, SettingsActivity.class));
        }
        if (itemId == R.id.search) {
            Toast.makeText(this, "search", Toast.LENGTH_SHORT).show();
        }
        if (itemId == R.id.action_linked_devices) {
            Toast.makeText(this, "action_linked_devices",
Toast.LENGTH_SHORT).show();
        }
        if (itemId == R.id.action_new_broadcast) {
            Toast.makeText(this, "action_new_broadcast",
Toast.LENGTH_SHORT).show();
        }
        if (itemId == R.id.action_starred_messages) {
            Toast.makeText(this, "action_starred_messages",
Toast.LENGTH_SHORT).show();
        }
        if (itemId == R.id.action_payments) {
            Toast.makeText(this, "action_payments", Toast.LENGTH_SHORT).show();
        }
        if (itemId == R.id.action_new_group) {
            Toast.makeText(this, "action_new_group", Toast.LENGTH_SHORT).show();
        }
    }

```



```

        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onResume() {
        super.onResume();

        String currentId = FirebaseAuth.getInstance().getUid();
        assert currentId != null;
        reference.child("presence").child(currentId).setValue("Online");
    }

    @Override
    protected void onPause() {
        super.onPause();

        String currentId = FirebaseAuth.getInstance().getUid();
        assert currentId != null;
        reference.child("presence").child(currentId).setValue("Offline");
    }

    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(data != null) {
            if(data.getData() != null) {
                dialog.show();

                FirebaseStorage storage = FirebaseStorage.getInstance();
                Date date = new Date();
                StorageReference reference =
storage.getReference().child("status").child(date.getTime() + "");
                reference.putFile(data.getData()).addOnCompleteListener(new
OnCompleteListener<UploadTask.TaskSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<UploadTask.TaskSnapshot>
task) {
                        if(task.isSuccessful()) {
                            reference.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {

```

```

        @Override
        public void onSuccess(Uri uri) {
            senderID = FirebaseAuth.getInstance().getUid();
            FirebaseFirestore.getInstance().collection("Users").document(senderID).get().addOnComple
            eListener(new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot>
task) {
                    if (task.isSuccessful()) {
                        DocumentSnapshot document = task.getResult();
                        if (document != null && document.exists()) {
                            name = document.getString("userName");
                            imageUrl = document.getString("imageProfile");
                            UserStatus userStatus = new UserStatus();
                            userStatus.setName(name);
                            userStatus.setProfileImage(imageUrl);
                            userStatus.setLastUpdated(date.getTime());
                            HashMap<String, Object> obj = new HashMap<>();
                            obj.put("name", userStatus.getName());
                            obj.put("profileImage", userStatus.getProfileImage());
                            obj.put("lastUpdated", userStatus.getLastUpdated());
                            String imageUrl = uri.toString();
                            Status status = new Status(imageUrl,
userStatus.getLastUpdated());
                            database.getReference().child("stories")
                                .child(senderID)
                                .updateChildren(obj);
                            database.getReference().child("stories")
                                .child(FirebaseAuth.getInstance().getUid())
                                .child("statuses")
                                .push()
                                .setValue(status);
                            dialog.dismiss();
                        }
                    }
                }
            });
        }
    }
}

```

```

        }
    });
}
});
}}
});
}}}}

```

## **ContactsActivity.java**

```

package com.basheer.whatsappclone.activities.chat;
import android.annotation.SuppressLint;
import android.content.ContentResolver;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.view.View;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import com.basheer.whatsappclone.adapter.ContactsAdapter;
import com.basheer.whatsappclone.databinding.ActivityContactsBinding;
import com.basheer.whatsappclone.model.Users;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;
import java.util.List;

```

```

public class ContactsActivity extends AppCompatActivity {
    ActivityContactsBinding binding;
    List<Users> list = new ArrayList<>();
    ContactsAdapter adapter;
    FirebaseUser firebaseUser;
    FirebaseFirestore firestore;
    public static final int REQUEST_READ_CONTACTS = 79;
    ArrayList mobileArray;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityContactsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        binding.recyclerView.setLayoutManager(new LinearLayoutManager(this));
        firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        firestore = FirebaseFirestore.getInstance();
        if(firebaseUser != null) {
            getContactsFromPhone();
        }
        if (mobileArray != null) {
            getContactList();
        }
        binding.backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
    private void getContactsFromPhone() {
        if (ContextCompat.checkSelfPermission(this,
android.Manifest.permission.READ_CONTACTS)
            == PackageManager.PERMISSION_GRANTED) {
            mobileArray = getAllPhoneContacts();

```

```

        } else {
            requestPermission();
        }
    }

    private void requestPermission() {
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            android.Manifest.permission.READ_CONTACTS)) {
            // show UI part if you want here to show some rationale !!!
        } else {
            ActivityCompat.requestPermissions(this, new
            String[]{android.Manifest.permission.READ_CONTACTS},
                REQUEST_READ_CONTACTS);
        }
        if (ActivityCompat.shouldShowRequestPermissionRationale(this,
            android.Manifest.permission.READ_CONTACTS)) {
        } else {
            ActivityCompat.requestPermissions(this, new
            String[]{android.Manifest.permission.READ_CONTACTS},
                REQUEST_READ_CONTACTS);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
    permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQUEST_READ_CONTACTS) {
            if (grantResults.length > 0 && grantResults[0] ==
            PackageManager.PERMISSION_GRANTED) {
                mobileArray = getAllPhoneContacts();
            } else {
                finish();
            }
        }
    }
}

```

```

@SuppressLint("Range")
private ArrayList getAllPhoneContacts() {
    ArrayList<String> phoneList = new ArrayList<>();
    ContentResolver cr = getContentResolver();
    Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
        null, null, null, null);
    if ((cur != null ? cur.getCount() : 0) > 0) {
        while (cur.moveToNext()) {
            String id = cur.getString(
                cur.getColumnIndex(ContactsContract.Contacts._ID));
            if (cur.getInt(cur.getColumnIndex(
ContactsContract.Contacts.HAS_PHONE_NUMBER)) > 0) {
                Cursor pCur = cr.query(
                    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                    null,
                    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?",
                    new String[]{id}, null);
                while (true) {
                    assert pCur != null;
                    if (!pCur.moveToNext()) break;
                    String phoneNo = pCur.getString(pCur.getColumnIndex(
                        ContactsContract.CommonDataKinds.Phone.NUMBER));
                    phoneList.add(removeWhiteSpace(phoneNo));
                }
                pCur.close();
            }
        }
    }
    if (cur != null) {
        cur.close();
    }
    return phoneList;
}

private void getContactList() {

```

```

        firestore.collection("Users").get().addOnSuccessListener(new
OnSuccessListener<QuerySnapshot>() {
    @Override
    public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
        for (QueryDocumentSnapshot snapshots : queryDocumentSnapshots) {
            String userID = snapshots.getString("userID");
            String userName = snapshots.getString("userName");
            String imageUrl = snapshots.getString("imageProfile");
            String desc = snapshots.getString("bio");
            String phone = snapshots.getString("userPhone")
            Users user = new Users();
            user.setUserID(userID);
            user.setBio(desc);
            user.setUserName(userName);
            user.setImageProfile(imageUrl);
            user.setUserPhone(phone);
            assert userID != null;
            if(!userID.equals(firebaseUser.getUid())) {
                if (mobileArray.contains(removeFirstThreeChars(user.getUserPhone())) {
                    list.add(user);
                }
            }
        }
        adapter = new ContactsAdapter(list, ContactsActivity.this);
        binding.recyclerView.setAdapter(adapter);
    }
});
}

public String removeWhiteSpace(String number){
    return number.replaceAll("\\s", "");
}

public static String removeFirstThreeChars(String originalString) {
    return originalString.substring(3);
}

```

```
}
```

## **ChatActivity.java**

```
package com.basheer.whatsappclone.activities.chat;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.Vibrator;
import android.provider.MediaStore;
import android.text.Editable;
import android.text.TextUtils;
import android.text.TextWatcher;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import com.android.volley.AuthFailureError;
```



```
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.android.volley.toolbox.Volley;
import com.basheer.whatsappclone.R;
import com.basheer.whatsappclone.activities.profile.UserProfileActivity;
import com.basheer.whatsappclone.adapter.ChatsAdapter;
import com.basheer.whatsappclone.databinding.ActivityChatBinding;
import com.basheer.whatsappclone.manage.ChatService;
import com.basheer.whatsappclone.manage.FirebaseService;
import com.basheer.whatsappclone.model.Chats;
import com.basheer.whatsappclone.dialog.DialogReviewSendImage;
import com.bumptech.glide.Glide;
import com.devlomi.record_view.OnRecordListener;
import com.google.firebase.auth.FirebaseAuth;
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.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.UploadTask;
import org.json.JSONObject;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.concurrent.TimeUnit;
```

```

public class ChatActivity extends AppCompatActivity {
    ActivityChatBinding binding;
    ValueEventListener valueEventListener;
    private String receiverID;
    ChatsAdapter adapter;
    DatabaseReference reference;
    private DatabaseReference chatMessagesReference;
    FirebaseStorage storage;
    List<Chats> list = new ArrayList<>();
    String userProfile, userName, mobileNumber, token;
    private boolean isActionShown = false;
    private ChatService chatService;
    private MediaRecorder mediaRecorder;
    String senderRoom, receiverRoom;
    Uri imageUri;
    String audio_path;
    String sTime;
    private static final int REQUEST_AUDIO_PERMISSION_CODE = 332;
    static int PERMISSION_CODE = 100;
    String senderUid;
    String receiverUid;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityChatBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        initialize();
        initButtonClick();
        Intent intent = getIntent();
        senderUid = FirebaseAuth.getInstance().getUid();
        receiverUid = intent.getStringExtra("userID");
        senderUid = FirebaseAuth.getInstance().getUid();
        senderRoom = senderUid + receiverID;
        receiverRoom = receiverUid + senderUid;
    }
}

```

```

adapter = new ChatsAdapter(list, this, senderRoom, receiverRoom);
binding.messageContent.setAdapter(adapter);
LinearLayoutManager layoutManager = new LinearLayoutManager(this);
layoutManager.setStackFromEnd(true);
binding.messageContent.setLayoutManager(layoutManager);
chatMessagesReference = FirebaseDatabase.getInstance().getReference()
    .child("chats")
    .child(senderRoom)
    .child("messages");
valueEventListener = new ValueEventListener() {
    @SuppressWarnings("NotifyDataSetChanged")
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        List<Chats> updatedList = new ArrayList<>();
        for (DataSnapshot snapshot : dataSnapshot.getChildren()) {
            Chats chat = snapshot.getValue(Chats.class);
            if (chat != null) {
                chat.setMessageId(snapshot.getKey());
                updatedList.add(chat);
            }
        }
        runOnUiThread(() -> {
            list.clear();
            list.addAll(updatedList);
            adapter.notifyDataSetChanged();
            binding.messageContent.smoothScrollToPosition(adapter.getItemCount());
        });
    }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
        // Handle possible errors
        Log.e("ChatActivity", "Listener was cancelled");
    }
};

```

```

reference.child("presence").child(receiverUid).addValueEventListener(new
ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if(snapshot.exists()) {
            String status = snapshot.getValue(String.class);
            assert status != null;
            if(!status.isEmpty()) {
                if(status.equals("Offline")) {
                    binding.status.setVisibility(View.GONE);
                } else {
                    binding.status.setText(status);
                    binding.status.setVisibility(View.VISIBLE);
                }
            }
        }
    }
    @Override
    public void onCancelled(@NonNull DatabaseError error) {
    }
});
final Handler handler = new Handler();
binding.messageBox.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
    }
    @Override
    public void afterTextChanged(Editable s) {
        reference.child("presence").child(senderUid).setValue("typing...");
        handler.removeCallbacksAndMessages(null);
        handler.postDelayed(userStoppedTyping, 1000);
    }
}
final Runnable userStoppedTyping = new Runnable() {

```

```

        @Override
        public void run() {
            reference.child("presence").child(senderUid).setValue("Online");
        }
    };
});
}

public void initialize() {
    reference = FirebaseDatabase.getInstance().getReference();
    Intent intent = getIntent();
    userName = intent.getStringExtra("userName");
    receiverID = intent.getStringExtra("userID");
    userProfile = intent.getStringExtra("userProfile");
    mobileNumber = intent.getStringExtra("mobileNumber");
    chatService = new ChatService(this, receiverID);
    if (receiverID != null) {
        binding.userName.setText(userName);
        assert userProfile != null;
        if (userProfile.isEmpty()) {
            binding.userProfile.setImageResource(R.drawable.avatar);
        } else {
            Glide.with(this).load(userProfile).into(binding.userProfile);
        }
    }
    binding.messageBox.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        }

        @SuppressWarnings("UseCompatLoadingForDrawables")
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            if (TextUtils.isEmpty(binding.messageBox.getText().toString())) {
                binding.recordButton.setVisibility(View.VISIBLE);
            } else {
                binding.recordButton.setVisibility(View.INVISIBLE);
            }
        }
    });
}

```

```

        binding.sendButton.setVisibility(View.VISIBLE);
    }
}

@Override
public void afterTextChanged(Editable s) {
}

});

binding.call.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(ContextCompat.checkSelfPermission(ChatActivity.this,
Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(ChatActivity.this, new
String[]{Manifest.permission.CALL_PHONE}, PERMISSION_CODE);
        }
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_CALL);
        intent.setData(Uri.parse("tel:"+mobileNumber));
        startActivity(intent);
    }
});

binding.recordButton.setRecordView(binding.recordView);
binding.recordView.setLockEnabled(true);
binding.recordView.setRecordLockImageView(findViewById(R.id.record_lock));
binding.recordView.setCancelBounds(8);
binding.recordView.setSmallMicColor(Color.parseColor("#FA1100"));
binding.recordView.setSlideToCancelText("Slide to cancel");
binding.recordView.setLessThanSecondAllowed(false);
binding.recordView.setCounterTimeColor(Color.parseColor("#777373"));
binding.recordView.setShimmerEffectEnabled(true);
binding.recordView.setTimeLimit(30000);
binding.recordView.setTrashIconColor(Color.parseColor("#000000"));
binding.recordView.setRecordButtonGrowingAnimationEnabled(true);
binding.recordButton.setScaleUpTo(1.5f);

```

```

binding.recordLock.setDefaultCircleColor(Color.parseColor("#2C977D"));
binding.recordLock.setCircleLockedColor(Color.parseColor("#FFFFFFFF"));
binding.recordLock.setLockColor(Color.WHITE);
binding.recordButton.setSendIconResource(R.drawable.send_icon);
binding.recordView.setOnRecordListener(new OnRecordListener() {
    @Override
    public void onStart() {
        if (!checkPermissionFromDevice()) { // Notice the '!' to invert the condition
            requestPermission();
        } else {
            binding.cardlayout.setVisibility(View.INVISIBLE);
            startRecord();
            Vibrator vibrator = (Vibrator)
getSystemService(Context.VIBRATOR_SERVICE);
            if (vibrator != null) {
                vibrator.vibrate(100); // Ensure vibrator is not null
            }
        }
    }
    @Override
    public void onCancel() {
        binding.cardlayout.setVisibility(View.VISIBLE);
        if (mediaRecorder != null) { // Check if mediaRecorder is not null
            try {
                mediaRecorder.reset();
            } catch (Exception ignored) {
            }
        }
    }
    @Override
    public void onFinish(long recordTime, boolean limitReached) {
        binding.cardlayout.setVisibility(View.VISIBLE);
        try {
            sTime = getHumanTimeText(recordTime);
            stopRecord();
        } catch (Exception ignored) {
        }
    }
}

```

```

    }}
    @Override
    public void onLessThanSecond() {
        //When the record time is less than One Second
        binding.cardlayout.setVisibility(View.VISIBLE);
    }
    @Override
    public void onLock() {
        //When Lock gets activated
        Log.d("RecordView", "onLock");
    }
});
binding.recordView.setOnBasketAnimationEndListener(() ->
binding.cardlayout.setVisibility(View.VISIBLE));
}
private
void stopRecord() {
    try {
        if (mediaRecorder != null) {
            mediaRecorder.stop();
            mediaRecorder.reset();
            mediaRecorder.release();
            mediaRecorder = null;
            chatService.sendVoice(audio_path);
        }
    } catch (Exception ignored) {
    }
}
@SuppressWarnings("DefaultLocale")
private String getHumanTimeText(Long milliseconds) {
    return String.format("%02d",
        TimeUnit.MILLISECONDS.toSeconds(milliseconds) -
            TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(milliseconds)));
}

```



```

private void initButtonClick() {
    binding.sendButton.setOnClickListener(v -> {
        if (!TextUtils.isEmpty(binding.messageBox.getText().toString())) {
            chatService.sendTextMsg(binding.messageBox.getText().toString());
            binding.messageBox.setText("");
        }
    });
    binding.backButton.setOnClickListener(v -> finish());
    binding.userProfile.setOnClickListener(v -> startActivity(new Intent(ChatActivity.this,
UserProfileActivity.class)
        .putExtra("userID", receiverID)
        .putExtra("imageProfile", userProfile)
        .putExtra("mobileNumber", mobileNumber)
        .putExtra("userName", userName)));
    binding.attachment.setOnClickListener(v -> {
        if(isActionShown) {
            binding.layoutShow.setVisibility(View.GONE);
            isActionShown = false;
        } else {
            binding.layoutShow.setVisibility(View.VISIBLE);
            isActionShown = true;
        }
    });
    binding.gallery.setOnClickListener(v -> openGalery());
    binding.camera.setOnClickListener(v -> openCamera());
    binding.cameraIcon.setOnClickListener(v -> openCamera());
}

private boolean checkPermissionFromDevice() {
    int write_external_storage_result = ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE);
    int record_audio_result = ContextCompat.checkSelfPermission(this,
Manifest.permission.RECORD_AUDIO);
    return write_external_storage_result == PackageManager.PERMISSION_DENIED ||
record_audio_result == PackageManager.PERMISSION_DENIED;
}

```

```

}

private void requestPermission() {
    ActivityCompat.requestPermissions(this, new String[] {
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
        Manifest.permission.RECORD_AUDIO,
    }, REQUEST_AUDIO_PERMISSION_CODE);
}

private void startRecord() {
    setUpMediaRecorder();
    try {
        mediaRecorder.prepare();
        mediaRecorder.start();
    } catch (IOException ignored) {
    }
}

private void setUpMediaRecorder() {
    File audioDir = getExternalFilesDir(Environment.DIRECTORY_MUSIC);
    assert audioDir != null;
    if (!audioDir.exists() && !audioDir.mkdirs()) {
        Log.e("setUpMediaRecorder", "Failed to create directory for audio recordings.");
        return;
    }
    String fileName = UUID.randomUUID().toString() + "_audio_record.m4a";
    File audioFile = new File(audioDir, fileName);
    audio_path = audioFile.getAbsolutePath();
    mediaRecorder = new MediaRecorder();
    try {
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
        mediaRecorder.setOutputFile(audio_path);
    } catch (Exception e) {
        Log.e("setUpMediaRecorder", "Exception setting up media recorder", e);
    }
}

```

```

private void openGalery() {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    startActivityForResult(intent, 45);
}

@SuppressWarnings("QueryPermissionsNeeded")
private void openCamera() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, 440);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (resultCode == RESULT_OK && data != null && data.getData() != null) {
        imageUri = data.getData();
        try {
            Bitmap bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(),
imageUri);
            reviewImage(bitmap);
        } catch (IOException ignored) {
        }
    }

    if (requestCode == 440 && resultCode == RESULT_OK) {
        if (data != null && data.getExtras() != null) {
            ProgressDialog progressDialog = new ProgressDialog(ChatActivity.this);
            progressDialog.setMessage("Sending image...");
            progressDialog.show();
            Bitmap imageBitmap = (Bitmap) data.getExtras().get("data");
            if (imageBitmap != null) {
                ByteArrayOutputStream baos = new ByteArrayOutputStream();
                imageBitmap.compress(Bitmap.CompressFormat.JPEG, 100, baos);
            }
        }
    }
}

```

```

        byte[] dataImage = baos.toByteArray();
        UploadTask uploadTask =
FirebaseStorage.getInstance().getReference().child("ImagesChats/").putBytes(dataImage);
        uploadTask.addOnFailureListener(exception -> {
            progressDialog.dismiss();
        }).addOnSuccessListener(taskSnapshot -> {
            taskSnapshot.getStorage().getDownloadUrl().addOnSuccessListener(uri -> {
                String imageUrl = uri.toString();
                chatService.sendImage(imageUrl);
                progressDialog.dismiss();
            });
        });
    }
}

private void reviewImage(Bitmap bitmap) {
    new DialogReviewSendImage(ChatActivity.this, bitmap).show() -> {
        if (imageUri != null) {
            ProgressDialog progressDialog = new ProgressDialog(ChatActivity.this);
            progressDialog.setMessage("Sending image...");
            progressDialog.show();
            binding.layoutShow.setVisibility(View.GONE);
            isActionShown = false;
            new FirebaseService(ChatActivity.this).uploadImageToFirebaseStorage(imageUri,
new FirebaseService.OnCallBack() {
                @Override
                public void onUploadSuccess(String imageUrl) {
                    chatService.sendImage(imageUrl);
                    progressDialog.dismiss();
                }
            }
            @Override
            public void onUploadFailed(Exception e) {
                progressDialog.dismiss();
            }
        });
    });
}

```

```

    }
});
}
@Override
protected void onResume() {
    super.onResume();
    chatMessagesReference.addValueEventListener(valueEventListener);
    String currentId = FirebaseAuth.getInstance().getUid();
    assert currentId != null;
    reference.child("presence").child(currentId).setValue("Online");
}
@Override
protected void onPause() {
    super.onPause();
    if (valueEventListener != null) {
        chatMessagesReference.removeEventListener(valueEventListener);
    }
    String currentId = FirebaseAuth.getInstance().getUid();
    assert currentId != null;
    reference.child("presence").child(currentId).setValue("Online");
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.chat_main, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    return super.onOptionsItemSelected(item);
}
public void sendNotification(String name, String message, String token) {
    try {
        RequestQueue queue = Volley.newRequestQueue(this);
        String url = "https://fcm.googleapis.com/fcm/send";
        JSONObject data = new JSONObject();
        data.put("title", name);
        data.put("body", message);
    }
}

```

```

JSONObject notificationData = new JSONObject();
notificationData.put("notification", data);
notificationData.put("to", token);
JsonObjectRequest request = new JsonObjectRequest(url, notificationData
    , new Response.Listener<JSONObject>() {
        @Override
        public void onResponse(JSONObject jsonObject) {
            Toast.makeText(ChatActivity.this, "success", Toast.LENGTH_SHORT).show();
        }
    }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError volleyError) {
            Toast.makeText(ChatActivity.this, volleyError.getLocalizedMessage(),
Toast.LENGTH_SHORT).show();}
    }) {
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            HashMap<String, String> map = new HashMap<>();
            String Key =
"Key=AAAADymcVLA:APA91bFcLRinaRhDHd350DEfZwVJ7vO215TFJtI6NfQS5F_Hm
_OA_wMSazfEpGUeyr6Kjw1d8Fz0unB_8qVMUmgq706V74aDsXF7nAvoad5WYGFAsm
QvsO3ILy8Kx-rg33HjAPTjuN7a";
            map.put("Authorization", Key);
            map.put("Content-Type", "application/json");
            return map;
        }
    };
    queue.add(request);
} catch (Exception ignored) {}
}}

```

### **ChatBotActivity.java**

```

package com.basheer.whatsappclone.activities.chatbot;
import android.os.Bundle;

```

```

import android.view.View;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import com.basheer.whatsappclone.adapter.ChatBotMessageAdapter;
import com.basheer.whatsappclone.databinding.ActivityChatBotBinding;
import com.basheer.whatsappclone.model.ChatBotMessage;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import okhttp3.Call;
import okhttp3.Callback;
import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class ChatBotActivity extends AppCompatActivity {
    ActivityChatBotBinding binding;
    List<ChatBotMessage> messageList;
    ChatBotMessageAdapter messageAdapter;
    public static final MediaType JSON = MediaType.get("application/json;
charset=utf-8");
    OkHttpClient client = new OkHttpClient();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityChatBotBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        messageList = new ArrayList<>();

```

```

messageAdapter = new ChatBotMessageAdapter(messageList);
binding.messageContent.setAdapter(messageAdapter);
LinearLayoutManager llm = new LinearLayoutManager(this);
llm.setStackFromEnd(true);
binding.messageContent.setLayoutManager(llm);
binding.sendButton.setOnClickListener((v)->{
    String question = binding.messageBox.getText().toString().trim();
    addToChat(question,ChatBotMessage.SENT_BY_ME);
    binding.messageBox.setText("");
    callAPI(question);
});
binding.backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});
}

void addToChat(String message,String sentBy){
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            messageList.add(new ChatBotMessage(message,sentBy));
            messageAdapter.notifyDataSetChanged();
            binding.messageContent.smoothScrollToPosition(messageAdapter.getItemCount());
        }
    });
}

void addResponse(String response){
    messageList.remove(messageList.size()-1);
    addToChat(response,ChatBotMessage.SENT_BY_BOT);}

void callAPI(String question){
    //okhttp
    messageList.add(new ChatBotMessage("Typing...

```



```

",ChatBotMessage.SENT_BY_BOT));
JSONObject jsonBody = new JSONObject();
try {
    jsonBody.put("model","gpt-3.5-turbo-instruct");
    jsonBody.put("prompt",question);
    jsonBody.put("max_tokens",4000);
    jsonBody.put("temperature",0);
} catch (JSONException e) {
    e.printStackTrace();}
RequestBody body = RequestBody.create(jsonBody.toString(),JSON);
Request request = new Request.Builder()
    .url("https://api.openai.com/v1/completions")
    .header("Authorization","Bearer sk-
S4fIvJ8qIhZ3QlsBb92ZT3BlbkFJtJuz0NNbGVCump9EFvSa")
    .post(body)
    .build();
client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(@NonNull Call call, @NonNull IOException e) {
        addResponse("Failed to load response due to "+e.getMessage());}
    @Override
    public void onResponse(@NonNull Call call, @NonNull Response response) throws
IOException {
        if(response.isSuccessful()){
            JSONObject jsonObject = null;
            try {
                jsonObject = new JSONObject(response.body().string());
                JSONArray jsonArray = jsonObject.getJSONArray("choices");
                String result = jsonArray.getJSONObject(0).getString("text");
                addResponse(result.trim());
            } catch (JSONException e) {
                e.printStackTrace();}
        }else{
            addResponse("Failed to load response due to "+response.body().toString());

```

```
    }}  
});  
}}
```

## **ProfileActivity.java**

```
package com.basheer.whatsappclone.activities.profile;  
import static com.google.common.io.Files.getFileExtension;  
import android.annotation.SuppressLint;  
import android.app.AlertDialog;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.drawable.BitmapDrawable;  
import android.graphics.drawable.Drawable;  
import android.net.Uri;  
import android.os.Bundle;  
import android.os.Environment;  
import android.provider.MediaStore;  
import android.text.TextUtils;  
import android.view.View;  
import android.view.WindowManager;  
import android.widget.EditText;  
import android.widget.Toast;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityOptionsCompat;  
import androidx.core.content.FileProvider;  
import com.basheer.whatsappclone.R;  
import com.basheer.whatsappclone.common.Common;  
import com.basheer.whatsappclone.databinding.ActivityProfileBinding;  
import com.basheer.whatsappclone.activities.startup.SplashScreenActivity;  
import com.bumptech.glide.Glide;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;
```

```

import com.google.android.material.bottomsheet.BottomSheetDialog;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Locale;
import java.util.Objects;

public class ProfileActivity extends AppCompatActivity {
    ActivityProfileBinding binding;
    FirebaseUser firebaseUser;
    FirebaseFirestore firestore;
    BottomSheetDialog bottomSheetDialog, bottomSheetDialogEditName;
    Uri imageUri;
    ProgressDialog dialog;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityProfileBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        dialog = new ProgressDialog(this);
        dialog.setMessage("Updating..");
        dialog.setCancelable(false);
        firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
        firestore = FirebaseFirestore.getInstance();
        if(firebaseUser!=null) {
            getInfo();
        }
        initActionClick();
    }

```

```

binding.backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();}
});
}

public void initActionClick() {
    binding.floating.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showBottomSheetPickPhoto();}
    });

    binding.linearLayoutforname.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showBottomSheetEditName();}
    });

    binding.userProfileImage.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            binding.userProfileImage.invalidate();
            Drawable dr = binding.userProfileImage.getDrawable();
            Common.IMAGE_BITMAP = ((BitmapDrawable)dr.getCurrent()).getBitmap();
            ActivityOptionsCompat activityOptionsCompat =
ActivityOptionsCompat.makeSceneTransitionAnimation(ProfileActivity.this,
binding.userProfileImage, "image");

            Intent intent = new Intent(ProfileActivity.this, ViewProfileZoomActivity.class);
            startActivity(intent, activityOptionsCompat.toBundle());}
    });

    binding.signOut.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            showDialogSignOut();}
    });
}

```

```

binding.backButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();}
});
}

private void showDialogSignOut() {
    AlertDialog.Builder builder = new AlertDialog.Builder(ProfileActivity.this);
    builder.setMessage("Do you want to sign out ?");
    builder.setPositiveButton("Sign out", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
            FirebaseAuth.getInstance().signOut();
            startActivity(new Intent(ProfileActivity.this, SplashScreenActivity.class));}
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();}
    });
    AlertDialog alertDialog = builder.create();
    alertDialog.show();}

private void showBottomSheetEditName() {
    @SuppressWarnings("InflateParams") View view =
getLayoutInflater().inflate(R.layout.bottom_sheet_edit_name, null);
    ((View) view.findViewById(R.id.cancel)).setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            bottomSheetDialogEditName.dismiss();}
    });
    EditText editText = view.findViewById(R.id.editText);
    ((View) view.findViewById(R.id.save)).setOnClickListener(new

```

```

View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if(TextUtils.isEmpty(editText.getText().toString())) {
            Toast.makeText(ProfileActivity.this, "Name can't be empty",
Toast.LENGTH_SHORT).show();
        } else {
            updateName(editText.getText().toString());
            bottomSheetDialogEditName.dismiss();
        }
    });
    bottomSheetDialogEditName = new BottomSheetDialog(this);
    bottomSheetDialogEditName.setContentView(view);
    Objects.requireNonNull(bottomSheetDialogEditName.getWindow()).addFlags(WindowMan
ager.LayoutParams.FLAG_TRANSLUCENT_STATUS);
    bottomSheetDialogEditName.setOnDismissListener(new
DialogInterface.OnDismissListener() {
        @Override
        public void onDismiss(DialogInterface dialog) {
            bottomSheetDialogEditName = null;}
    });
    bottomSheetDialogEditName.show();}
    private void updateName(String newUserName) {
        firestore.collection("Users").document(firebaseUser.getId()).update("userName",
newUserName).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                getInfo();}
        });
    }
    private void showBottomSheetPickPhoto() {
        @SuppressWarnings("InflateParams") View view =
getLayoutInflater().inflate(R.layout.bottom_sheet_dialog, null);
        ((View) view.findViewById(R.id.gallery)).setOnClickListener(new

```

```

View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openGalery();
        bottomSheetDialog.dismiss();}
    });
    ((View) view.findViewById(R.id.camera)).setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openCamera();
        bottomSheetDialog.dismiss();}
    });
    bottomSheetDialog = new BottomSheetDialog(this);
    bottomSheetDialog.setContentView(view);
Objects.requireNonNull(bottomSheetDialog.getWindow()).addFlags(WindowManager.Layo
utParams.FLAG_TRANSLUCENT_STATUS);

    bottomSheetDialog.setOnDismissListener(new DialogInterface.OnDismissListener() {
        @Override
        public void onDismiss(DialogInterface dialog) {
            bottomSheetDialog = null;}
    });
    bottomSheetDialog.show();}
@SuppressLint("QueryPermissionsNeeded")
private void openCamera() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    String timeStamp = new SimpleDateFormat("yyyMMdd_HHmmss",
Locale.getDefault()).format(new Date());
    String imageFileName = "IMG_" + timeStamp + ".jpg";
    try {
        File file = File.createTempFile("IMG_" + timeStamp, ".jpg",
getExternalFilesDir(Environment.DIRECTORY_PICTURES));
        imageUri = FileProvider.getUriForFile(this, "com.basheer.whatsappclone.provider",
file);

```

```

        intent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
        intent.putExtra("listPhotoName", imageFileName);
        startActivityForResult(intent, 440);
    } catch (Exception ignored) {
    }
}

private void openGalery() {
    Intent intent = new Intent();
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    startActivityForResult(intent, 45);}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(data != null) {
        if (data.getData() != null) {
            imageUri = data.getData();
            uploadToFirebaseStorage();
        }
    }
    if (requestCode == 440 && resultCode == RESULT_OK) {
        uploadToFirebaseStorage();
    }
}

private void uploadToFirebaseStorage() {
    if(imageUri!=null) {
        dialog.show();

        StorageReference storageReference =
FirebaseStorage.getInstance().getReference().child("ProfileImages/" +
System.currentTimeMillis() + "." + getFileExtension(String.valueOf(imageUri)));
        storageReference.putFile(imageUri).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                Task<Uri> urlTask = taskSnapshot.getStorage().getDownloadUrl();
                while (!urlTask.isSuccessful());
                Uri downloadUrl = urlTask.getResult();

```



```

        final String sdownload_url = String.valueOf(downloadUrl);
        HashMap<String, Object> hashMap = new HashMap<>();
        hashMap.put("imageProfile", sdownload_url);
        dialog.dismiss();

        firestore.collection("Users").document(firebaseUser.getUid()).update(hashMap).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(ProfileActivity.this, "Upload successfully",
                Toast.LENGTH_SHORT).show();
                getInfo();}
        });
    }
});
}}

public void getInfo()
{ firestore.collection("Users").document(firebaseUser.getUid()).get().addOnSuccessListener(
new OnSuccessListener<DocumentSnapshot>() {
    @Override
    public void onSuccess(DocumentSnapshot documentSnapshot) {
        String userName = documentSnapshot.getString("userName");
        String userPhone = documentSnapshot.getString("userPhone");
        String profileImage = documentSnapshot.getString("imageProfile");
        binding.userName.setText(userName);
        binding.userPhone.setText(userPhone);
        Glide.with(ProfileActivity.this).load(profileImage).placeholder(R.drawable.avatar).into(binding.userProfileImage);}
    });
}}

```

### **UserProfileActivity.java**

```

package com.basheer.whatsappclone.activities.profile;

import android.content.Intent;

import android.os.Bundle;

```

```

import android.view.View;
import androidx.appcompat.app.AppCompatActivity;
import com.basheer.whatsappclone.R;
import com.basheer.whatsappclone.databinding.ActivityUserProfileBinding;
import com.bumptech.glide.Glide;

public class UserProfileActivity extends AppCompatActivity {
    ActivityUserProfileBinding binding;
    String receiverID;
    String userProfile;
    String userName;
    String mobileNumber;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityUserProfileBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());
        Intent intent = getIntent();
        userName = intent.getStringExtra("userName");
        receiverID = intent.getStringExtra("userID");
        userProfile = intent.getStringExtra("imageProfile");
        mobileNumber = intent.getStringExtra("mobileNumber");
        if (receiverID != null) {
            binding.titleName.setText(userName);
            binding.userName.setText(userName);
            binding.mobileNumber.setText(mobileNumber);
            if (userProfile == null || userProfile.isEmpty()) {
                binding.userProfileImage.setImageResource(R.drawable.avatar);
            } else {
                Glide.with(this).load(userProfile).into(binding.userProfileImage);
            }
        }
        binding.backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }
}

```

```
});  
}}
```

## **SettingsActivity.java**

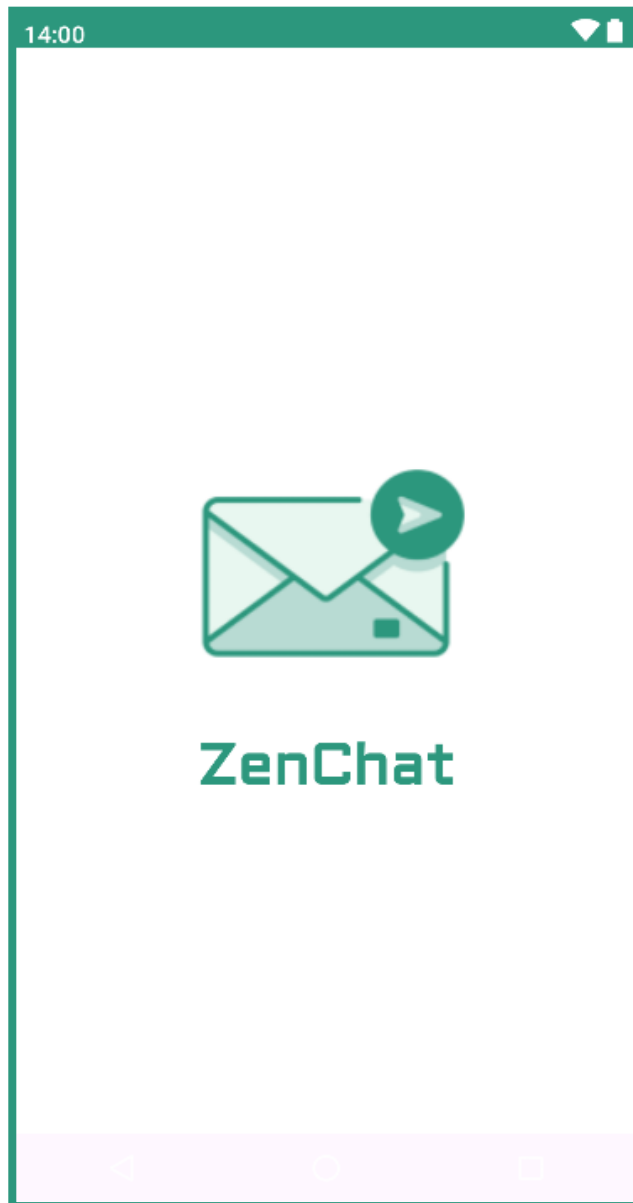
```
package com.basheer.whatsappclone.activities.settings;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import androidx.appcompat.app.AppCompatActivity;  
import com.basheer.whatsappclone.R;  
import com.basheer.whatsappclone.activities.profile.ProfileActivity;  
import com.basheer.whatsappclone.databinding.ActivitySettingsBinding;  
import com.bumptech.glide.Glide;  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.firebase.auth.FirebaseAuth;  
import com.google.firebase.auth.FirebaseUser;  
import com.google.firebase.firestore.DocumentSnapshot;  
import com.google.firebase.firestore.FirebaseFirestore;  
public class SettingsActivity extends AppCompatActivity {  
    ActivitySettingsBinding binding;  
    FirebaseUser firebaseUser;  
    FirebaseFirestore firestore;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = ActivitySettingsBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());  
        firebaseUser = FirebaseAuth.getInstance().getCurrentUser();  
        firestore = FirebaseFirestore.getInstance();  
        if(firebaseUser!=null) {  
            getInfo();}  
        initClickAction();  
        binding.backButton.setOnClickListener(new View.OnClickListener() {  
            @Override
```

```

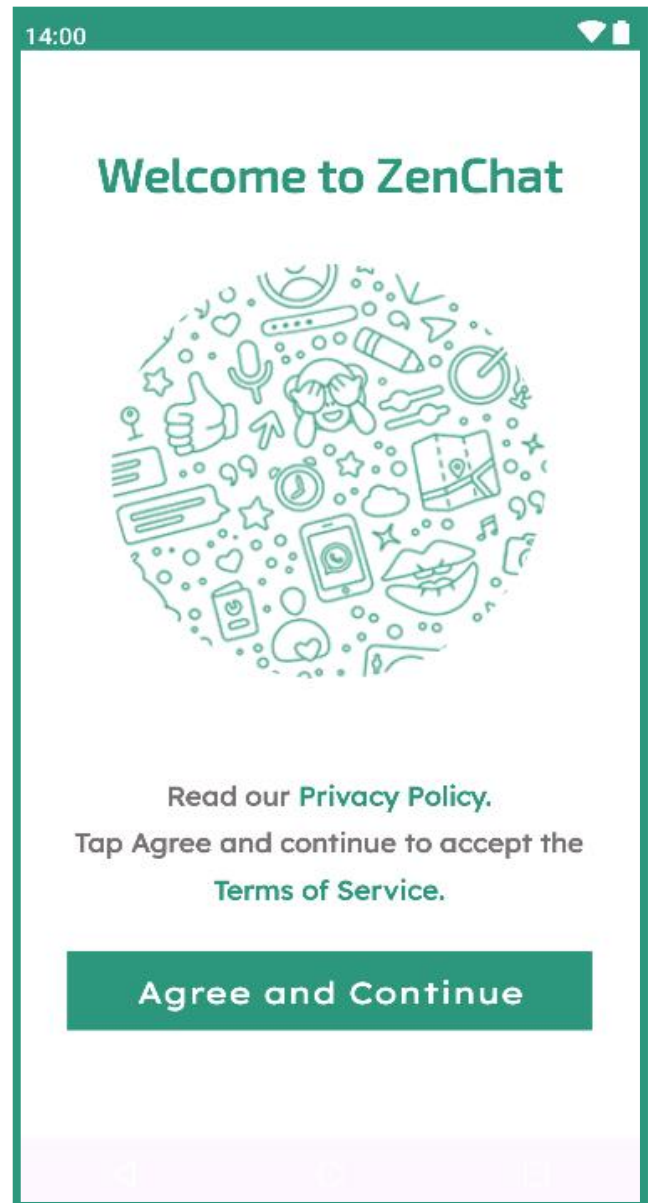
        public void onClick(View v) {
            finish();}
    });
}
public void initClickAction() {
    binding.linearlayoutForProfile.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(SettingsActivity.this, ProfileActivity.class));}
    });
    binding.backButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();}
    });
}
public void getInfo() {
    firestore.collection("Users").document(firebaseUser.getId()).get().addOnSuccessListener(n
    ew OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            String userName = documentSnapshot.getString("userName");
            String profileImage = documentSnapshot.getString("imageProfile");
            binding.userName.setText(userName);
            Glide.with(SettingsActivity.this).load(profileImage).placeholder(R.drawable.avatar).into(bin
            ding.userProfileImage);}
        });
    }}
}

```


## OUTPUT SCREEN



**Screen**




**Welcome Screen**



## OTP Verification

We will send you an **One Time Password**  
on this mobile number


**Enter Mobile Number**

 ▼

123 - 456 - 7890

**Generate OTP**

**Phone Number Verification**



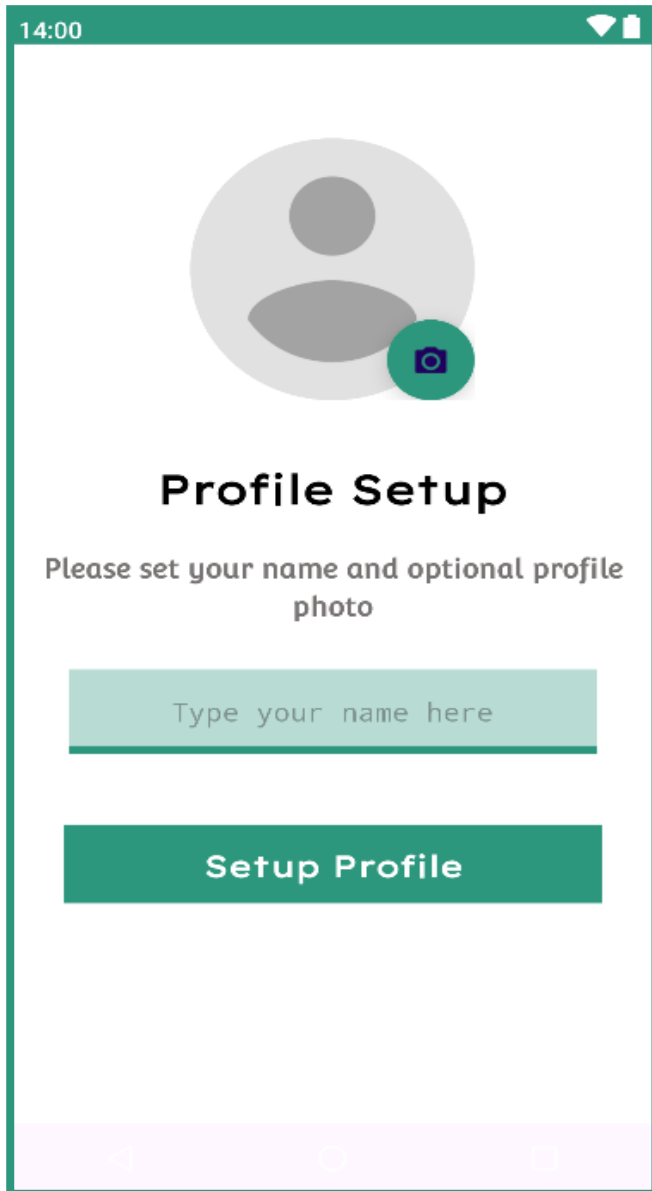
## OTP Verification

Enter OTP send to **+9190876653421**

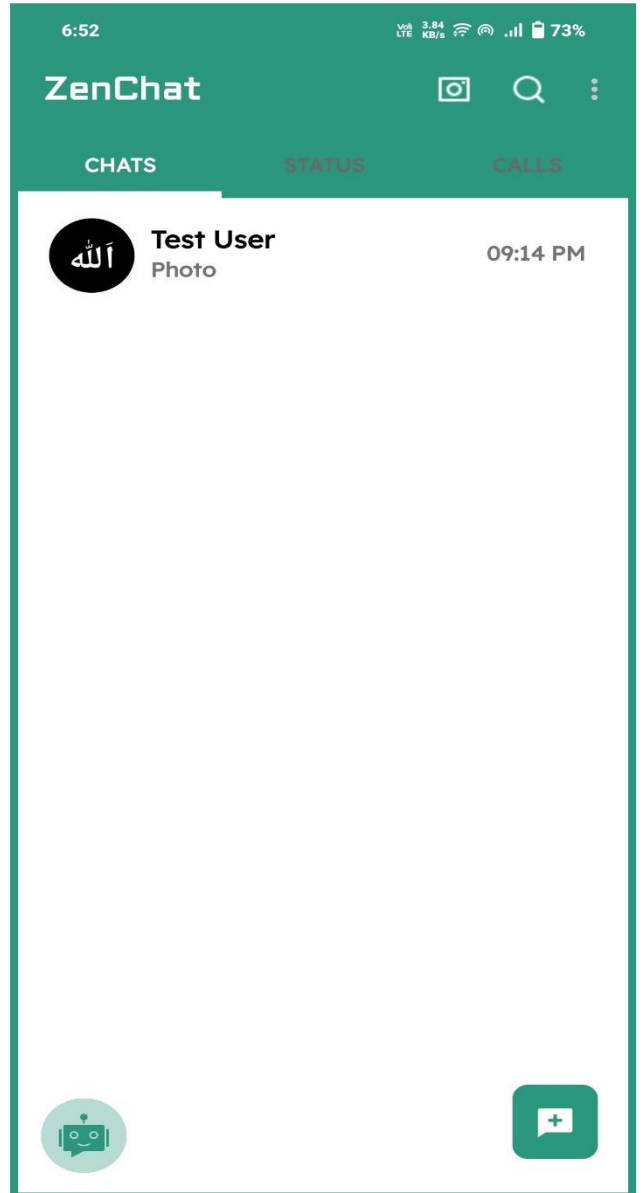
Don't receive the OTP?  
**RESEND OTP**

**Verify**

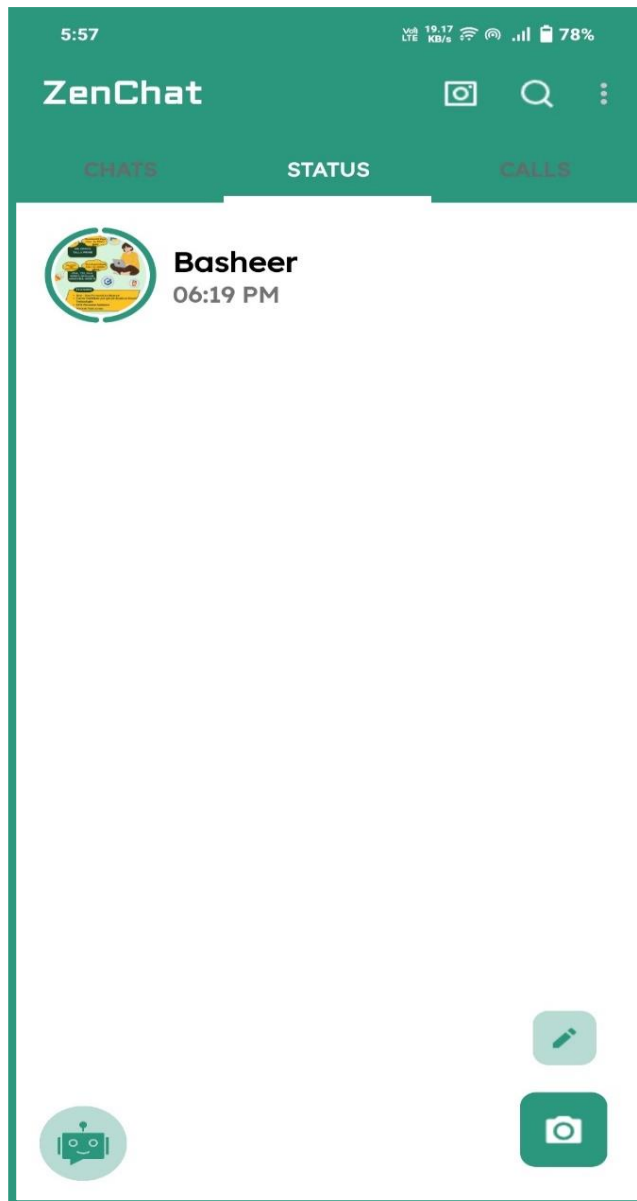
**OTP Verification Page**



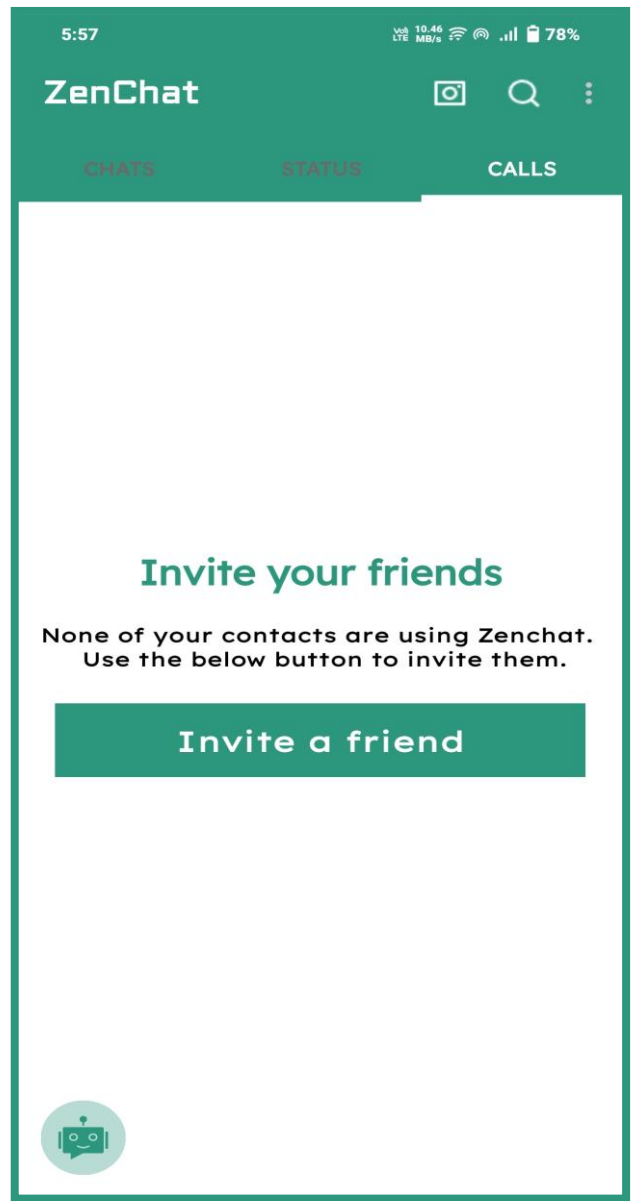
**Profile Setup Page**



**Main Page**



Status Page

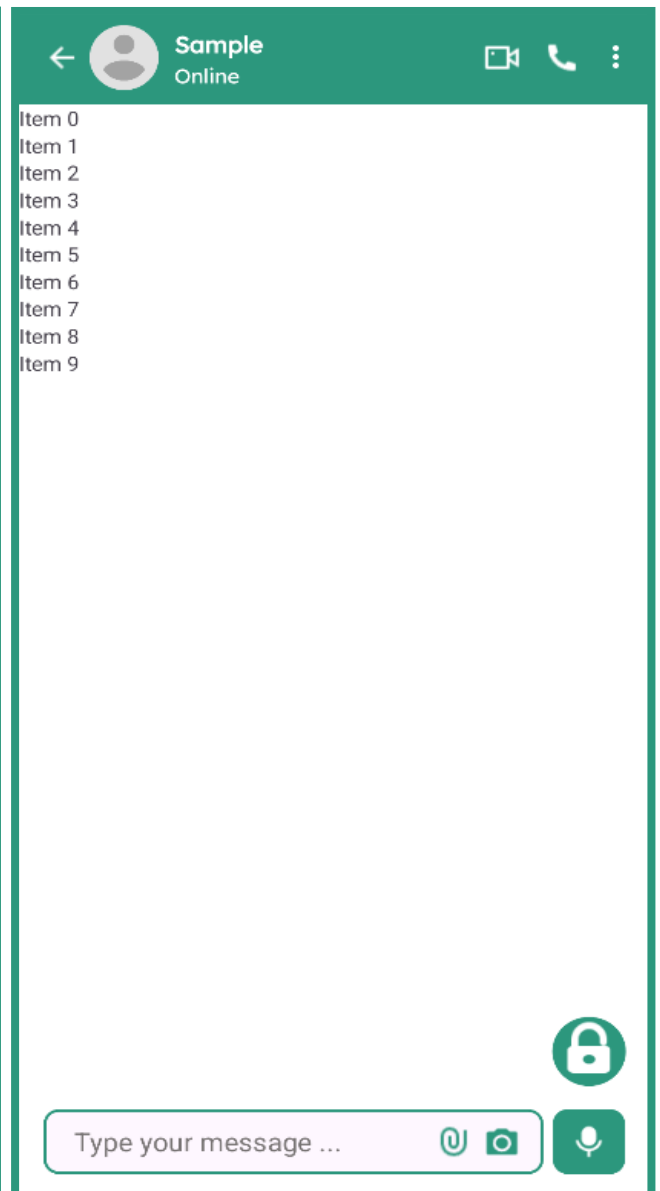


Call History Page

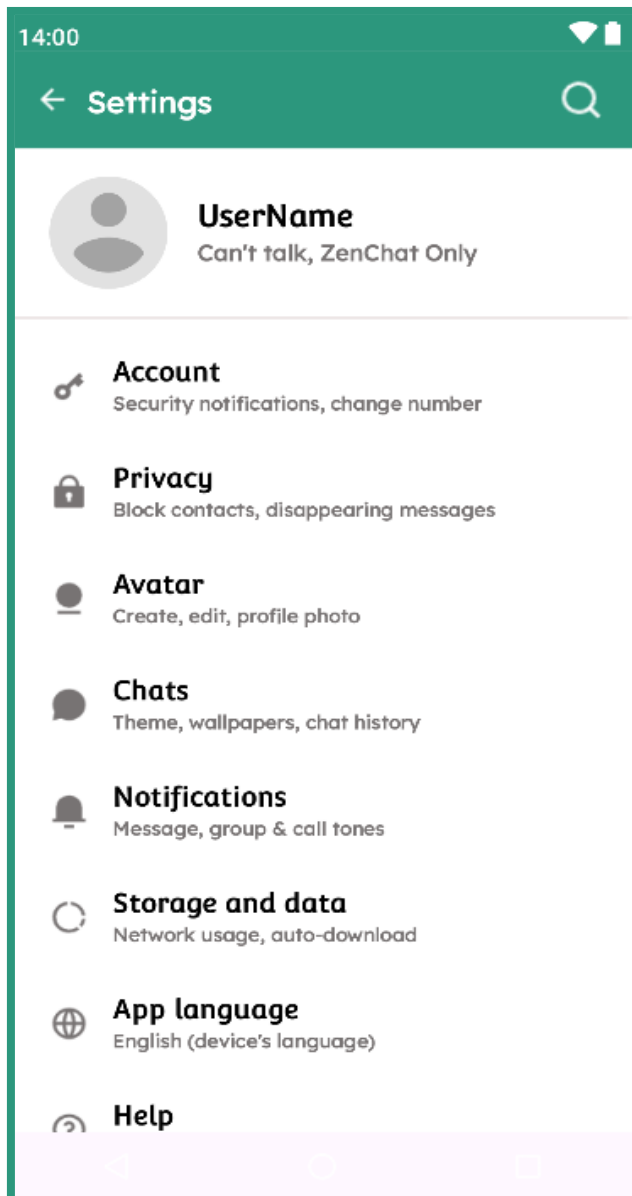




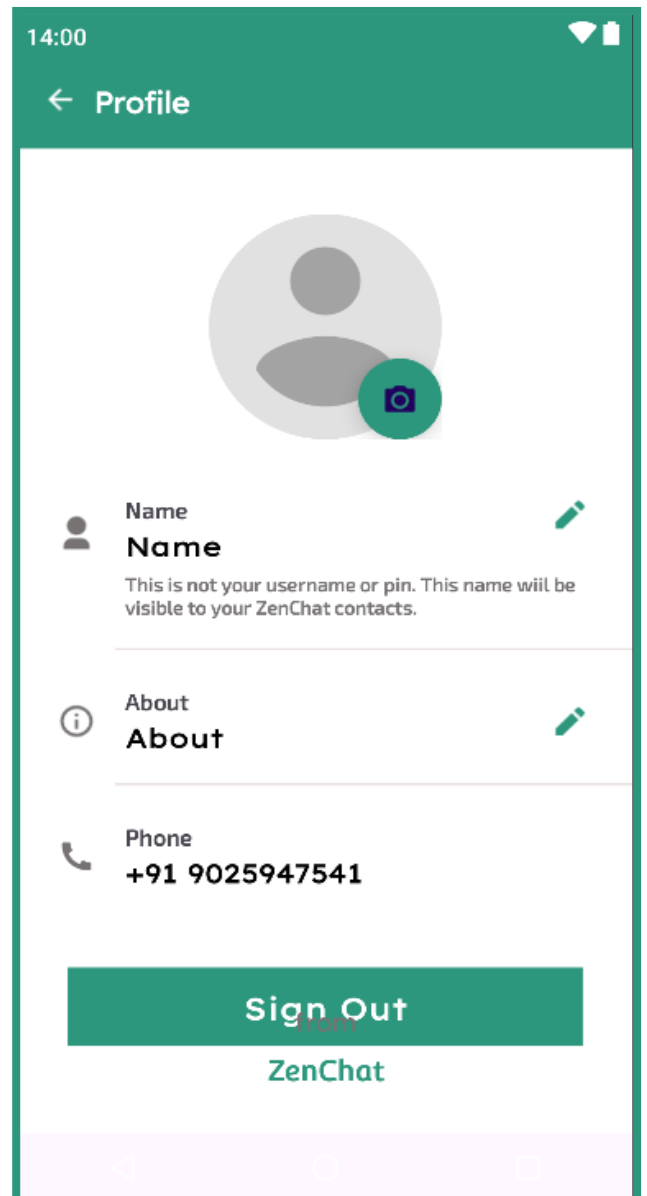
**List Contact Page**



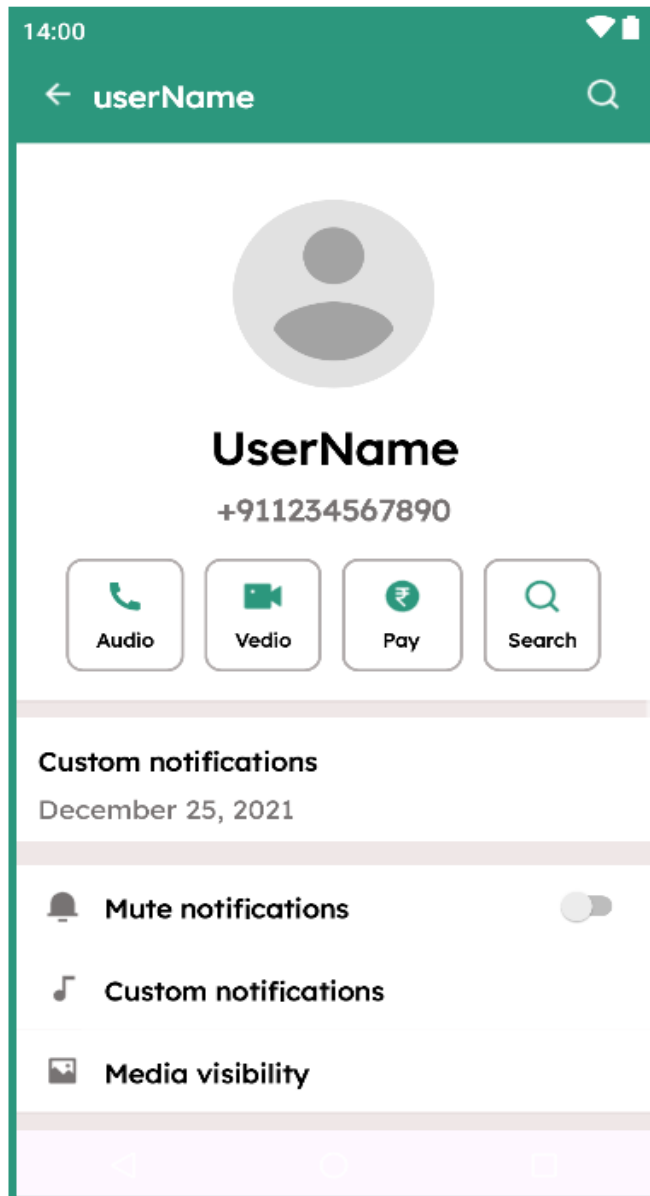
**Chatting Page**



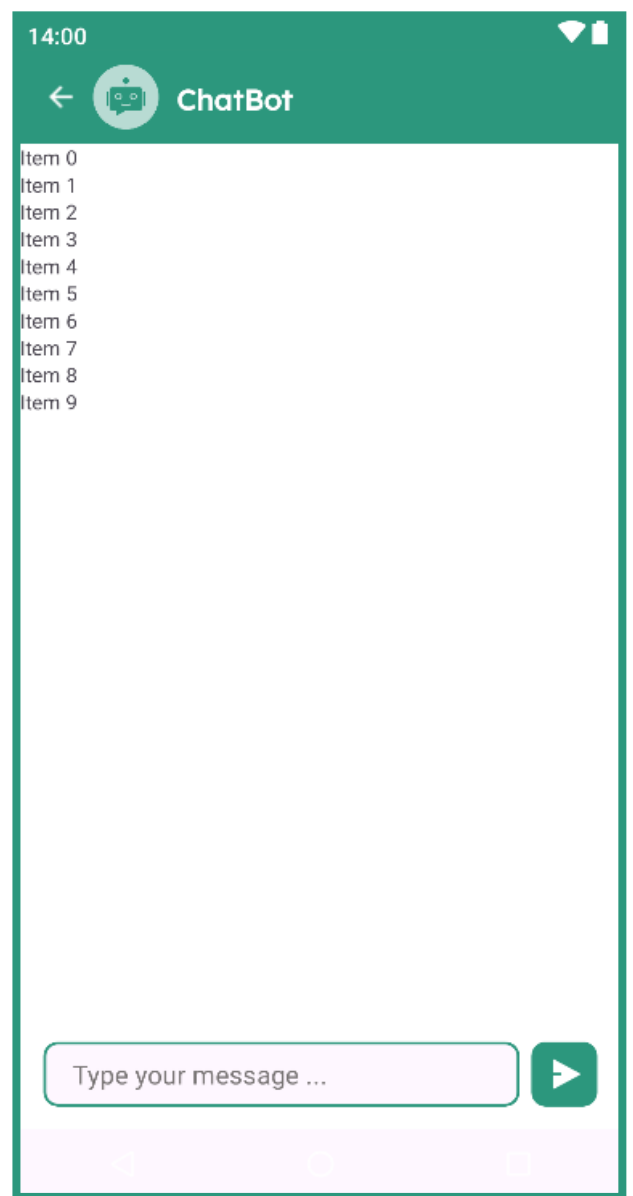
**Settings Page**



**User Profile Page**



**Receiver Profile PageSplash**



**Chatbot Chatting Page**

# DATABASE

## 1. Authentication

WhatsApp Clone ▾

Authentication

Users | Sign-in method | Templates | Usage | Settings | Extensions

Add user ↺ ⋮

Identifier	Providers	Created ↓	Signed in	User UID
+919876543210	📞	10 Apr 2024	10 Apr 2024	XpVUNE0EEtYHR3ieBokt9CH...
+919566898382	📞	9 Apr 2024	9 Apr 2024	sch6N6rV1nUsuAzlbsantf9QP...
+919080465177	📞	9 Apr 2024	9 Apr 2024	DGSmiSTMNFPKOGDXH3SHZT...
+919843782108	📞	27 Mar 2024	27 Mar 2024	7wz1M460icTewEGHeuAnBz...
+919025947541	📞	24 Mar 2024	10 Apr 2024	vpPFMqL02s596SHs57TqcOw...
+919600263688	📞	23 Mar 2024	28 Mar 2024	I7Tt2Rj862MgYbdAdaXEeqdH...
+911234567890	📞	22 Mar 2024	10 Apr 2024	I0rhlJ75bEdouWVxh2oVWlxH...

Rows per page: 50 1 ~ 7 of 7 < >

## 2. Chats

WhatsApp Clone ▾

Realtime Database

Data | Rules | Backups | Usage | Extensions

🛡️ Protect your Realtime Database resources from abuse, such as billing fraud or phishing [Configure App Check](#) ✕

https://whatsapp-clone-a58de-default-rtdb.firebaseio.com

```
https://whatsapp-clone-a58de-default-rtdb.firebaseio.com/
├── ChatList
│   ├── XpVUNE0EEtYHR3ieBokt9CHTgCP2
│   │   ├── I0rhlJ75bEdouWVxh2oVWlxHTXN2
│   │   │   └── chatid: "I0rhlJ75bEdouWVxh2oVWlxHTXN2"
│   │   └── I0rhlJ75bEdouWVxh2oVWlxHTXN2
│   │       ├── XpVUNE0EEtYHR3ieBokt9CHTgCP2
│   │       └── chatid: "XpVUNE0EEtYHR3ieBokt9CHTgCP2"
├── chats
│   ├── XpVUNE0EEtYHR3ieBokt9CHTgCP2I0rhlJ75bEdouWVxh2oVWlxHTXN2
│   │   ├── lastMsg: "Hi"
│   │   ├── lastMsgTime: 1712751629736
│   │   └── messages
│   │       ├── -Nv6wxUoTT9jbNFbv19I
│   │       │   ├── date: "10-04-2024"
│   │       │   ├── feeling: -1
│   │       │   ├── messageId: "-Nv6wxUJJmb3wLR51Y"
│   │       │   ├── receiver: "I0rhlJ75bEdouWVxh2oVWlxHTXN2"
│   │       │   └── sender: "XpVUNE0EEtYHR3ieBokt9CHTgCP2"
└── Database location: United States (us-central1)
```

### 3. User Information

WhatsApp Clone

Cloud Firestore

Protect your Cloud Firestore resources from abuse, such as billing fraud or phishing

Configure App Check

Panel view

Query builder

Users > XpVUNE0EEtYH.

More in Google Cloud

(default)

Users

XpVUNE0EEtYHR3ieBokt9CHTgCP2

+ Start collection

Users

+ Add document

XpVUNE0EEtYHR3ieBokt9CHTgCP2

18rh1J75bEdouWVxh2oVW1xHTXN2

+ Start collection

+ Add field

bio: "Can't talk only ZenChat"

dateOfBirth: ""

email: ""

gender: ""

imageCover: ""

imageProfile: "https://firebasestorage.googleapis.com/v0/b/whatsapp-clone-a58de.appspot.com/o/ProfileImages%2F1712748956197?alt=media&token=dbaa1018-fc85-4aab-a970-abdaf7791a8d"

status: ""

token: "e\_dHwM6QTC2R1RcbGc5IQs:APA91bHGA17wdoX3uYcuLrroPxsRr5oi8xF-9vIhJoab44vVQ6YxTQ0B5e91pWe\_WJmUJn38duGJE0Y3vZ-vkChJDVTnR6f1ydSribio2OFbo3IErbYlrvPF1JAB8YvVlkn\_egvq3Y"

userID: "XpVUNE0EEtYHR3ieBokt9CHTgCP2"

userName: "Test User 2"

userPhone: "+919876543210"

Database location: nam5

### 4. Firebase Storage

WhatsApp Clone

Storage

Files

Rules

Usage

Extensions

gs://whatsapp-clone-a58de.appspot.com

Upload file

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	ImagesChats/	—	Folder	—
<input type="checkbox"/>	ProfileImages/	—	Folder	—
<input type="checkbox"/>	status/	—	Folder	—

## 5. User Profile Image Storage




WhatsApp Clone


Storage

FilesRulesUsageExtensions

gs://whatsapp-clone-a58de.appspot.com > ProfileImages

Upload file

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 1712744870034.	396.61 KB	image/jpeg	10 Apr 2024
<input type="checkbox"/>	 1712748956197.	7.26 KB	image/jpeg	10 Apr 2024
<input type="checkbox"/>	 1712751795957.	7.26 KB	image/jpeg	10 Apr 2024



1712744870034.

Name

[1712744870034.](#)

Size

406,132 bytes

Type

image/jpeg

Created

10 Apr 2024, 15:57:53

Updated

10 Apr 2024, 15:57:53

File location

Other metadata

## 6. Chat Images Storage




WhatsApp Clone


Storage

FilesRulesUsageExtensions

gs://whatsapp-clone-a58de.appspot.com > ImagesChats

Upload file

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 1712762930808.jpg	1.66 MB	image/jpeg	10 Apr 2024
<input type="checkbox"/>	 1712978839951.jpg	1.81 MB	image/jpeg	13 Apr 2024
<input type="checkbox"/>	 1712978860866.jpg	418.64 KB	image/jpeg	13 Apr 2024



1712978839951.jpg

Name

[1712978839951.jpg](#)

Size

1,896,355 bytes

Type

image/jpeg

Created

13 Apr 2024, 08:57:26

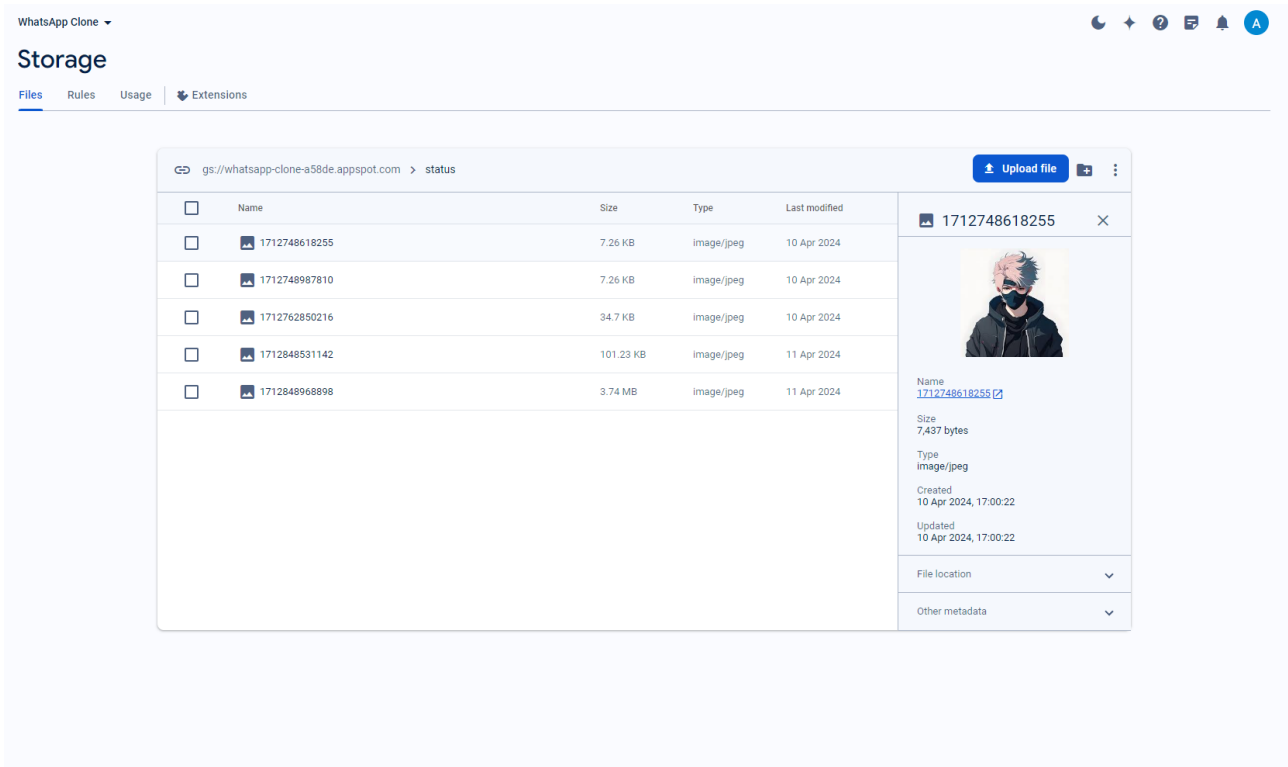
Updated

13 Apr 2024, 08:57:26

File location

Other metadata

# 7. User Status Images Storage



## **CONCLUSION & FUTURE ENHANCEMENT**

### **CONCLUSION:**

In conclusion, this project has successfully demonstrated the development of a robust, real-time chatting application using Java and Firebase. The primary objective of this project was to design and implement a user-friendly chat interface that allows users to communicate instantaneously, supported by Firebase's powerful backend services for real-time data synchronization and user authentication.

Throughout the development process, several challenges were encountered, particularly in ensuring efficient data handling and real-time updates of messages. These challenges were systematically addressed using Firebase's Firestore for seamless message storage and retrieval, and Firebase Authentication to securely manage user access and identity verification.

The application features a simple yet effective user interface that allows users to engage in private and group chats. Advanced features such as message notifications, online status indicators, and data encryption were integrated to enhance user interaction and privacy. The use of Java provided a strong and scalable foundation for building the application's functionality, ensuring that the system is maintainable and extendable.

Overall, this project not only fulfils the initial requirements but also provides a solid platform for future enhancements. The knowledge gained from using Java in conjunction with Firebase's real-time database and authentication services will be invaluable in further explorations and implementations of real-time, cloud-based applications.



## FUTURE ENHANCEMENT

- **Voice and Video Calling:** Implement real-time voice and video calling features to expand communication options.
- **File Sharing Capabilities:** Enable users to share files, and videos directly through the chat interface.
- **End-to-End Encryption:** Enhance security by incorporating end-to-end encryptions for all messages and calls.
- **User Interface Customization:** Allow users to customize chat themes, fonts, and backgrounds for a more personalized experience.
- **Group Chat Enhancements:** Add features like admin controls, group invitations, and member search functionality in group chats.
- **Offline Messaging:** Develop functionality for sending and receiving messages even when offline, syncing them when connectivity is restored.
- **Machine Learning Integration:** Utilize machine learning for predictive text input and smart replies based on user habits.
- **Privacy Settings:** Add more granular privacy settings, allowing users to control who can see their online status, last seen, and profile information.
- **Cloud Storage Integration:** Offer users the option to save backups of their chats and media to cloud storage services.

## **BIBLIOGRAPHY**

### **1. "Programming Java: A Beginner's Comprehensive Guide" by Herbert Schildt**

- This book offers a deep dive into Java programming, covering everything from basics to advanced topics. It's excellent for understanding the foundations and more complex aspects of Java, which are crucial for building robust applications.

### **2. "Firebase Essentials - Android Edition" by Neil Smyth**

- This book focuses specifically on integrating Firebase with Android applications. It covers Firebase's core services, including real-time databases, authentication, and analytics, which are integral to developing a chatting app.

### **3. "Professional Android 4th Edition" by Reto Meier, Ian Lake, and others**

- For comprehensive coverage of Android development, including how to effectively utilize Java in the context of Android, this book is a staple. It provides practical coding examples and explains the integration of various technologies, essential for any Android developer using Firebase.

### **4. "The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform" by Laurence Moroney**

- Written by a Google developer advocate, this book provides comprehensive coverage of Firebase, offering detailed guidance on how to leverage Firebase services, including the real-time database and authentication features, within your Android apps.

### **5. "Java: The Complete Reference" by Herbert Schildt**

- An excellent resource for both beginner and advanced Java programmers, this book covers all aspects of Java in depth, including its use in network communications, which is crucial for any real-time chatting application.

## REFERENCE

### Websites

- Android Tutorial from <https://www.geeksforgeeks.org/android-tutorial>

### Online Documentation

- Google Firebase. (n.d.). Firebase Documentation. Retrieved April 1, 2023, from <https://firebase.google.com/docs>
- Android Developers. (n.d.). Build a Responsive UI with Constraint Layout. Retrieved March 28, 2023, from <https://developer.android.com/training/constraint-layout>
- OpenAI. (2023). ChatGPT: Optimizing Language Models for Dialogue. Retrieved from <https://openai.com/research/chatgpt>

### Online Video

- Muhammad Ali's Coding Café - Make an Android App like WhatsApp - Android Chat App using Firebase - Firebase Group Chat Android - Android Group Chat App using Firebase. [Video]. YouTube. <https://youtube.com/playlist?list=PLxefhmF0pcPmtoud8f64EpgapkclClIj&si=4hXwGb7e9J6Sb1jU>
- Coding Bunch - Android Studio, WhatsApp Clone With Chatting Feature, Firebase Database. [Video]. YouTube. [https://youtu.be/NOY1-11xDQA?si=5ns7WLn\\_CTLJbtrW](https://youtu.be/NOY1-11xDQA?si=5ns7WLn_CTLJbtrW)
- Mian Speaks - Complete Android Chatting App like WhatsApp and Instagram. [Video]. YouTube. [https://youtu.be/F\\_UemS493IM?si=sOTgbvaEqzSkE177](https://youtu.be/F_UemS493IM?si=sOTgbvaEqzSkE177)
- Easy Tuto - How to create ChatGPT android app, Full Tutorial. [Video]. YouTube. [https://youtu.be/ahhze\\_u5ZUs?si=RFEOY7M6Cht9n59t](https://youtu.be/ahhze_u5ZUs?si=RFEOY7M6Cht9n59t)