ABSTRACT

In recent years, instant messaging applications have become an integral part of our daily lives, enabling us to stay connected and communicate seamlessly with friends, family, and colleagues. WhatsApp, one of the most popular messaging apps, has revolutionized the way people interact globally. This abstract presents a concept and overview of a WhatsApp clone, a replica of the original application, designed to provide similar functionality while offering flexibility for customization and integration into different platforms.

The transient chat application is a real-time messaging platform designed to provide seamless and efficient communication for users. The application aims to overcome the challenges posed by traditional chat applications, such as extensive user registration and the lack of intelligent features. It leverages the MQTT protocol for real-time message delivery and integrates the Firebase Smart Reply API to provide intelligent suggestions for replies based on the conversation history. The application offers a minimal user registration process, generating unique keys for users to join chat conversations quickly and anonymously. With a focus on scalability and performance, the application utilizes optimized algorithms and data structures to handle a large number of users and concurrent chat conversations. The user interface is intuitive, facilitating easy message input, display, and user identification. Security measures are implemented to protect user privacy and ensure the confidentiality of chat conversations. Through thorough testing and continuous improvement based on user feedback, the transient chat application delivers a reliable and enjoyable communication experience across multiple devices and platforms.

CONTENTS

CHAPTER : 1

INTRODUCTION

The project titled “TRANSIENT CHAT” is messaging application for sending messages. In the real world the communication plays a very vital role. People have been communicating with each other through various applications or mediums. In the beginning people communicated with each other using letters or other sources, as these mediums could take much time to deliver the content. Cell phones are another medium of communication but the drawback is for any limited or small message which need to be passed to another user then phone call is not an ideal way. The developers then looked to implement a text-based communication which would allow an in instant communication service. In 1984, the concept of SMS was developed in the Franco German GSM cooperation by Friedhelm Hillebrand and Bernard Ghillebaert. The limitation of SMS was the limited size i.e., 128 bytes, after the rise of smartphones from a decade many messaging applications have been developed. Some are Bluetooth based and some were internet based such as WhatsApp, WeChat and others. Android is an operating system for mobiles which was developed by google. This operating system allows the applications to be used on mobiles. As it was developed by google, android users can develop mobile applications and can be sold through android application stores such as play store.

ANDROID

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on- screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics. Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance –a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications downloaded. An applications ("apps") published, and over 50 billion April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform. About 70 percent of Android

smartphones run Google's ecosystem; some with vendor- customized user interface and software suite, such as TouchWiz and later One UI by Samsung, and HTC Sense. Competing Android ecosystems and forkww3s include Fire OS (developed by Amazon) or LineageOS. However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding. Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2021, it has over three billion monthly active users, the largest installed base of any operating system, and as of January 2021, the Google Play Store features over 3 million apps.The current stable version is Android 11, released on September 8, 2020.

FIREBASE

Firebase is a NoSQL database which make use of sockets which allows the users to store and retrieve the data from the database . An Android version should be greater than 2.3, android studio 1.5 or higher version, and android studio project are the prerequisites to connect the firebase to an android application. Firebase provides a various kind of services such as

Firebase Authentication: Firebase Authentication is useful to both developers and the users. Developing and maintaining sign-in set-up may be a bit difficult and time taking. Firebase provides an easy API for sign in. It also provides the data backup using real time databases.

Firebase cloud: For storing the data such as video, text, pictures building the infrastructure would be difficult and expensive for a new developer so the firebase provides the platform of cloud storage.

Real time database: It is a cloud hosted NoSQL database. Apart from the authentication, cloud service and real time databases firebase also provides a service for crash reporting

Crash Reporting: when some unexpected crashes occur in any applications it may be difficult to conclude why the application crashed. Firebase provides crash reporting service to deal with these crashes.

This project is concerned of a software application for the establishment of a real time communication services between operators/users. Chat application many-to-many type of communication system where the users will able to exchange the messages among themselves.

1.1 PROBLEM STATEMENT

The problem addressed by the transient chat application is the need for a seamless and efficient communication platform for users to engage in real-time chat conversations. Existing messaging applications often require extensive user registration, rely on centralized servers, and lack intelligent features that enhance user experience.

Key Challenges:

- User Registration: Many chat applications require users to create accounts, provide personal information, and go through a registration process, which can be time-consuming and inconvenient.
- Real-time Communication: Implementing a real-time chat system with minimal latency and reliable message delivery is a complex task that requires efficient protocols and technologies.
- Intelligent Features: Traditional chat applications lack intelligent features, such as smart replies, which can enhance user experience by suggesting relevant responses and improving conversation flow.
- Scalability: As the number of users and chat conversations increases, ensuring the scalability and performance of the chat application becomes a challenge, especially when dealing with large-scale deployments.

1.2 OBJECTIVES

- Develop a user-friendly chat application that allows users to engage in real-time conversations without the need for extensive user registration.
- Implement a real-time messaging system using the MQTT protocol to ensure low latency, reliable message delivery, and smooth chat interactions.
- Integrate the Firebase Smart Reply API to provide intelligent suggestions for replies based on the conversation history, enhancing user experience and improving conversation flow.
- Design and optimize the application architecture to handle a large number of users and concurrent chat conversations while maintaining high scalability and performance.
- Create an intuitive user interface with essential features such as message input, message display, and user identification to facilitate seamless communication.
- Incorporate appropriate security measures to protect user privacy and ensure the confidentiality of chat conversations.
- Conduct thorough testing and debugging to identify and resolve any performance or functionality issues, ensuring a stable and reliable chat application.
- Provide seamless interoperability across multiple devices and platforms, enabling users to access and use the chat application on various devices.
- Gather user feedback and continuously improve the application based on user needs, preferences, and emerging technologies.
- Deliver a fully functional and well-documented chat application that meets the requirements and expectations of users, providing an efficient and enjoyable communication experience.

CHAPTER : 2

SYSTEM DESIGN

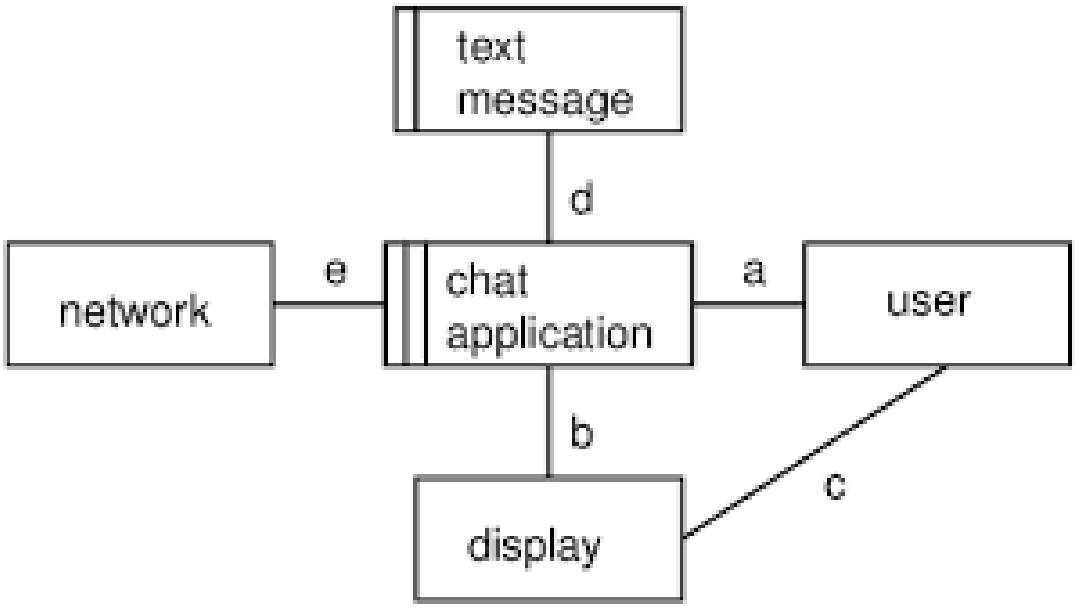

Fig 2.1) Device to Device Interaction

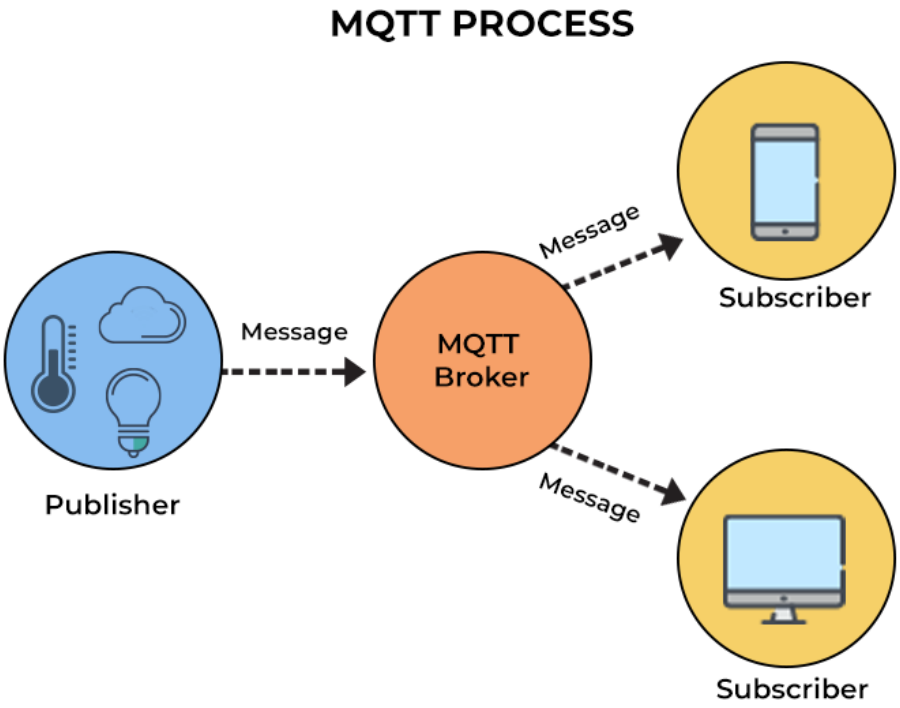

Fig 2.2) Device to Device Interaction

2.1 System Configuration

1. Operating System - Server: Windows , Client: Android 4.0 or higher
2. Development Environment - Android Studio
3. Firebase – Realtime database

2.2 Hardware Configuration

1. Processor – IntelCorei5 or above.
2. Ram – 4GB
3. Hard Disk – 64GB

2.3 Software Development Kit (SDK)

A software development kit (SDK or "devkit") is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To create applications you have to download this software development kit. For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you require an iOS SDK with swift language, and to develop MS Windows apps you require the .net language. There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include Google and Facebook.

2.4 Android Studio

Android Studio is an integrated development environment (IDE) for developing for the Android platform. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

Fig 2.3 Android Studio

1. GENERATING UNIQUE ID

The user must login in with the unique id . On successful login the user enters into his/her profile.

2. CHAT

The user must unique id in order to start chat , the user must have unique id in order to send messages to those who are online.

3. HOME SCREEN

After the client is logged in, the client can choose to send a message, the unique id of users who are registered to the application is displayed, and be able to get back when the client wants to, which will be by logging out by hitting the back .

4. CHAT WINDOW

When the client wants to message the user, the client clicks the unique id , the user can send a message to another user.

CHAPTER : 3

IMPLEMENTATION

```
private void processMessage(MqttMessage message) {
String messageContent = message.toString();
char a = messageContent.charAt(messageContent.length() - 1);

if (a == generatedChar) {
// Do not show the message
} else {
// Show the message
User user = new User("Paul", "df", null);
Date date = Calendar.getInstance().getTime();
String messageToDisplay = messageContent.substring(0, messageContent.length() - 1);
Message message1 = new Message("Doe", messageToDisplay, date, user);
sentMessageAdapter.addToStart(message1, true);
mFirebaseTextMessages.add(FirebaseTextMessage.createForRemoteUser(messageToDispla
y, System.currentTimeMillis(), "a"));
suggestReplies();
}
}
```

Code 3.1 Adding Dependencies To build.gradle and syncing

```
//Method to subscribe to a specific topic in this case the Unique ID
private void subscribe(final String topic) {
int qos = 1;
try {
IMqttToken subToken = client.subscribe(topic, qos);
subToken.setActionCallback(new IMqttActionListener() {
@Override
public void onSuccess(IMqttToken asyncActionToken) {
// The subscription was successful
}

@Override
public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
// The subscription could not be performed, handle the failure
}
});
} catch (MqttException e) {
e.printStackTrace();
}
client.setCallback(new MqttCallback() {
@Override
public void connectionLost(Throwable cause) {
// Handle the connection loss
}
```

```
@Override
public void messageArrived(String topic, MqttMessage message) throws Exception {
// Handle the received message
}

@Override
public void deliveryComplete(IMqttDeliveryToken token) {
// Handle the completion of message delivery
}
});
}
```

Code 3.2 MQTT Subscription Method

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

getSupportActionBar().hide();
final Handler handler = new Handler();
final Runnable r = new Runnable() {
public void run() {
Intent intent = new Intent(SplashScreen.this, MainActivity.class);
startActivity(intent);
}
};
handler.postDelayed(r, 2500);
}
```

Code 3.3 Splash Screen Activity

```
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.util.Log;

import com.google.firebase.ml.naturallanguage.FirebaseNaturalLanguage;
import com.google.firebase.ml.naturallanguage.smartreply.FirebaseSmartReply;
import com.google.firebase.ml.naturallanguage.smartreply.FirebaseTextMessage;
import com.google.firebase.ml.naturallanguage.smartreply.SmartReplySuggestion;
import com.google.firebase.ml.naturallanguage.smartreply.SmartReplySuggestionResult;

import org.eclipse.paho.android.service.MqttAndroidClient;
import org.eclipse.paho.client.mqttv3.IMqttActionListener;
import org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;
import org.eclipse.paho.client.mqttv3.IMqttToken;
import org.eclipse.paho.client.mqttv3.MqttCallback;
import org.eclipse.paho.client.mqttv3.MqttClient;
```

```
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;

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

public class SmartReplyMqttClient {
private static final String MQTT_BROKER = "tcp://broker.hivemq.com:1883";

private Context context;
private MqttAndroidClient client;
private FirebaseSmartReply smartReply;
private List<FirebaseTextMessage> conversationHistory;

public SmartReplyMqttClient(Context context) {
this.context = context;
conversationHistory = new ArrayList<>();
smartReply = FirebaseNaturalLanguage.getInstance().getSmartReply();
}

public void connectAndSubscribe(String topic) {
String clientId = MqttClient.generateClientId();
client = new MqttAndroidClient(context, MQTT_BROKER, clientId);

try {
IMqttToken token = client.connect();
token.setActionCallback(new IMqttActionListener() {
@Override
public void onSuccess(IMqttToken asyncActionToken) {
Log.d("MQTT", "Connected to MQTT broker");
subscribe(topic);
}

@Override
public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
Log.e("MQTT", "Failed to connect to MQTT broker", exception);
}
});
} catch (MqttException e) {
e.printStackTrace();
}
}

private void subscribe(String topic) {
int qos = 1;
try {
IMqttToken subToken = client.subscribe(topic, qos);
subToken.setActionCallback(new IMqttActionListener() {
```

```
@Override
public void onSuccess(IMqttToken asyncActionToken) {
Log.d("MQTT", "Subscribed to topic: " + topic);
}

@Override
public void onFailure(IMqttToken asyncActionToken, Throwable exception) {
Log.e("MQTT", "Failed to subscribe to topic: " + topic, exception);
}
});
} catch (MqttException e) {
e.printStackTrace();
}

client.setCallback(new MqttCallback() {
@Override
public void connectionLost(Throwable cause) {
Log.e("MQTT", "Connection lost", cause);
}

@Override
public void messageArrived(String topic, MqttMessage message) throws Exception {
String receivedMessage = new String(message.getPayload());
processMessage(receivedMessage);
}

@Override
public void deliveryComplete(IMqttDeliveryToken token) {
// Message delivery complete
}
});
}

public void sendMessage(String topic, String message) {
try {
MqttMessage mqttMessage = new MqttMessage();
mqttMessage.setPayload(message.getBytes());
client.publish(topic, mqttMessage);
} catch (MqttException e) {
e.printStackTrace();
}
}

private void processMessage(String message) {
String messageContent = message.trim();

// Add the received message to the conversation history
conversationHistory.add(FirebaseTextMessage.createForRemoteUser(messageContent,
```

```
System.currentTimeMillis(), "remoteUserId"));

// Generate smart reply suggestions
smartReply.suggestReplies(conversationHistory)
.addOnSuccessListener(result -> {
if (result.getStatus() == SmartReplySuggestionResult.STATUS_SUCCESS) {
List<SmartReplySuggestion> suggestions = result.getSuggestions();
if (!suggestions.isEmpty()) {
// Get the suggested reply
SmartReplySuggestion suggestion = suggestions.get(0);
String replyText = suggestion.getText();

// Send the smart reply as a message
sendMessage(MQTT_TOPIC, replyText);
}
}
})
.addOnFailureListener(e -> Log.e("Firebase", "Failed to generate smart reply", e));
}

public boolean isOnline() {
ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = cm.getActiveNetworkInfo();
return networkInfo != null && networkInfo.isConnected();
}
}
```

Code 3.4 Smart Replay System

CHAPTER : 4

RESULTS

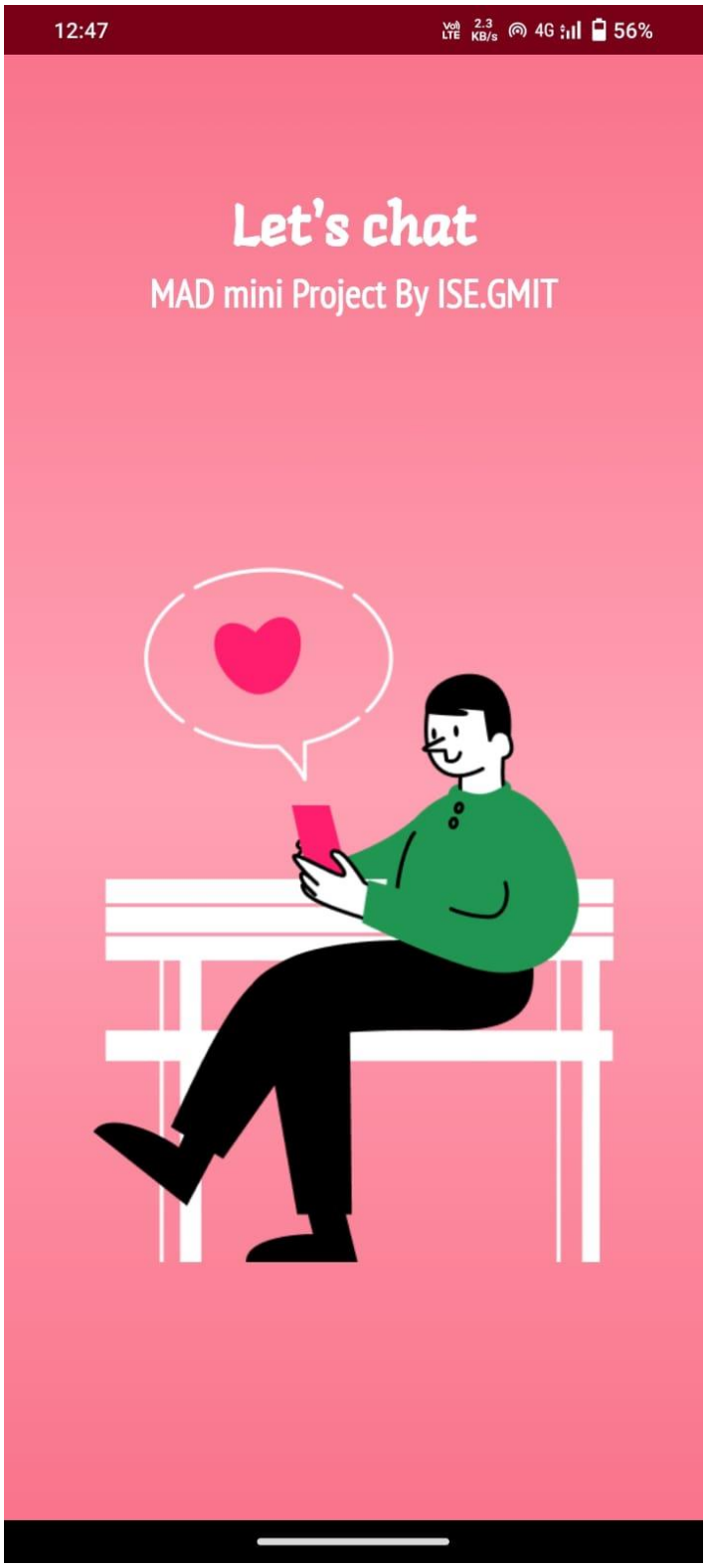

Fig 6.1 First Screen

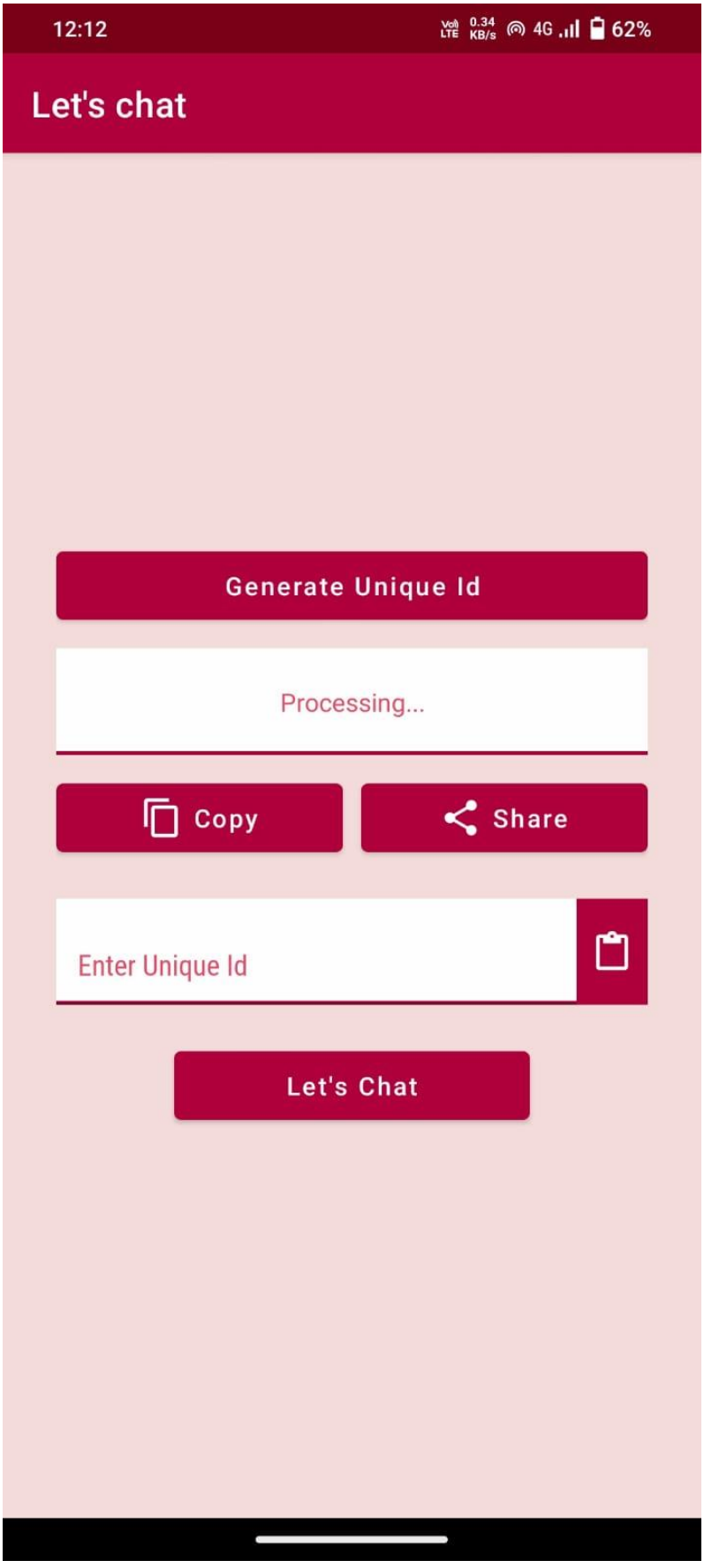

Fig 6.2 Home Screen

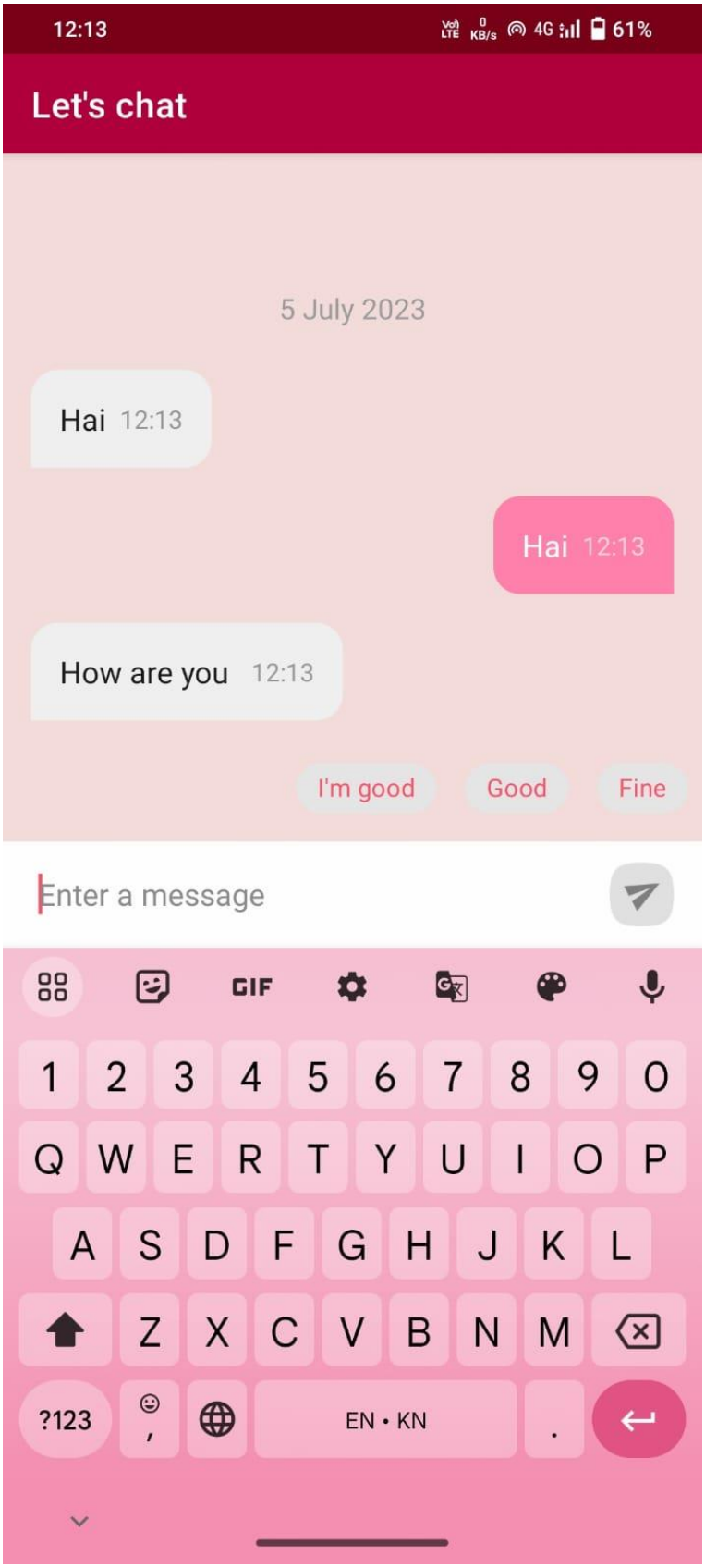

Fig 6.3 Send/Reply Message

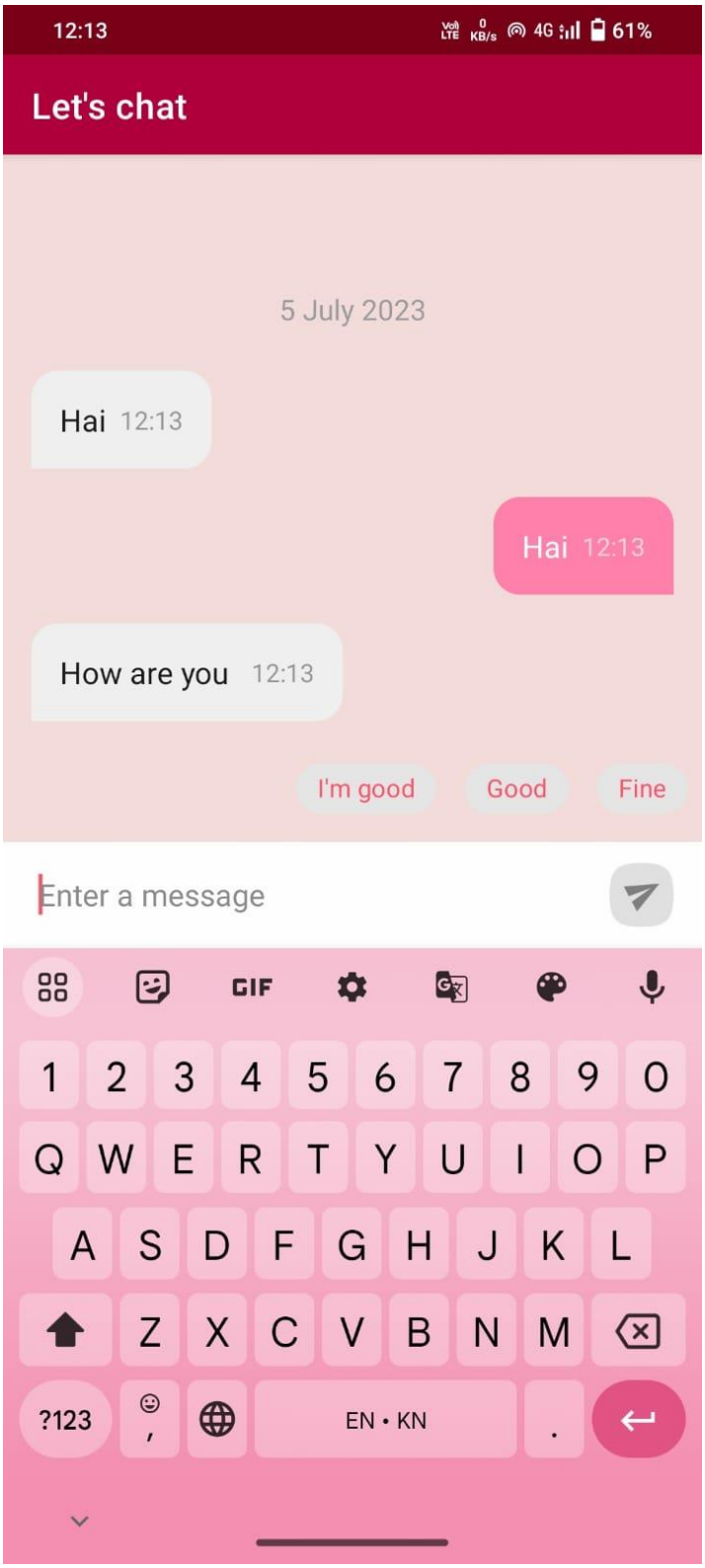

Fig 6.4 suggestion message to reply

CHAPTER : 5

CONCLUSION

In conclusion, the development of a real-time communication application using Firebase on the Android platform has successfully achieved the goals of providing a platform for registered users to exchange text messages in real-time. By leveraging Firebase's real-time database and cloud services, the application enables instant messaging between users, ensuring a seamless and responsive communication experience.

The project implemented user registration and authentication features, ensuring that only registered users can access and utilize the messaging platform. Firebase's backend services facilitated secure user authentication and storage of user information, enhancing the overall security of the application.

Throughout the development process, the project team prioritized the reliability and efficiency of the messaging platform. Firebase's real-time database feature enabled the application to update and synchronize messages in real-time, providing users with a seamless and immediate messaging experience.

The application's reliance on an active internet connection for communication ensures that users can send and receive messages without any significant delays. Additionally, the utilization of Firebase's cloud services guarantees scalability and availability, allowing the application to handle a growing number of users and messages.

Overall, the real-time communication application successfully utilizes Firebase's features and services to create a robust and efficient platform for Android users to exchange text messages. The project's implementation of user registration, authentication, real-time messaging, and reliance on Firebase's backend services ensures a reliable and secure communication experience for the application's users.

CHAPTER : 6

REFERENCES

Here are some references that you can use for your report on the transient chat application:

MQTT Protocol:

OASIS MQTT Technical Committee. (n.d.). MQTT Version 3.1.1 Specification. Retrieved from https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

Firebase Smart Reply API:

Firebase Documentation. (n.d.). Smart Reply. Retrieved from https://firebase.google.com/docs/ml-kit/generate-smart-replies

Android Development:

Android Developers Documentation. (n.d.). Official Android Developer Documentation. Retrieved from https://developer.android.com/docs

Glide Image Loading Library:

Bump tech. (n.d.). Glide: Image Loading Library. Retrieved from https://github.com/bumptech/glide

Eclipse Paho MQTT Client Library for Android:

Eclipse Paho Project. (n.d.). Paho Android Service. Retrieved from https://github.com/eclipse/paho.mqtt.android

Connectivity Manager Documentation:

Android Developers Documentation. (n.d.). Connectivity Manager. Retrieved from https://developer.android.com/reference/android/net/ConnectivityManager