

# **Project Report Format**

**Team ID:** Team-591016

**Project Title:** ChatConnect - A Real-Time Chat and Communication App

## **1. INTRODUCTION**

### **1.1 Project Overview**

Initially, there is a user interface designed for user registration and login. Users should have the capability to securely create accounts and log in. The system is designed to uphold the confidentiality and integrity of user data. For data storage, a Firebase Realtime Database is employed to store user information and messages in JSON format. The backend development encompasses the creation of a server-side application responsible for managing user authentication, message storage, and delivery through APIs.

Furthermore, a user interface is constructed for message composition and interaction. This interface facilitates users in composing and sending messages, viewing message history, and seamlessly interacting within the application. Additionally, another user interface is developed for contacts, allowing users to easily access recently chatted individuals without having to enter the receiver's username every time they send a message.

Our system prioritizes the security of user credentials, data transmission, and prevention of unauthorized access. Rigorous testing is conducted to ensure the app's functionality and to identify and resolve any bugs or issues during the debugging process.

### **1.2 Purpose**

The primary purpose of the chat app is to facilitate efficient and convenient communication among individuals and groups. Here are the core objectives:

1. **Connectivity:** Enable users to connect with friends, family, colleagues, and other acquaintances in real-time, regardless of geographical locations.
2. **Collaboration:** Facilitate teamwork and collaboration through group chats, allowing users to discuss projects, share ideas, and work together efficiently.
3. **Convenience:** Provide a user-friendly interface that is easy to navigate, ensuring that users can engage in conversations effortlessly.
4. **Expressiveness:** Support various forms of communication, including text, images, videos, and other multimedia, allowing users to express themselves in diverse ways.
5. **Privacy and Security:** Prioritize the protection of user data and privacy through secure authentication processes and encryption measures.
6. **Stay Updated:** Keep users informed with instant notifications, ensuring that they are aware of new messages and can respond promptly.

In summary, the chat app serves as a versatile and accessible platform for fostering communication and collaboration, meeting the diverse needs of its users.

## **2. LITERATURE SURVEY**

### **2.1 Existing problem**

The proliferation of diverse chat applications, each offering distinct features and operating on various platforms, gives rise to challenges in interoperability. Users frequently find themselves juggling multiple apps to communicate across different platforms, resulting in fragmentation and inconvenience.

Efficient data storage and management become worrisome when dealing with a high volume of messages and multimedia files, particularly in applications where data retention plays a pivotal role.

Sustaining consistent real-time communication, free from delays or message loss, poses a persistent technical challenge, especially with the expansion of user bases.

Certain applications encounter difficulties in delivering a seamless, intuitive, and user-friendly interface, impacting ease of use and overall user satisfaction.

### **2.2 References**

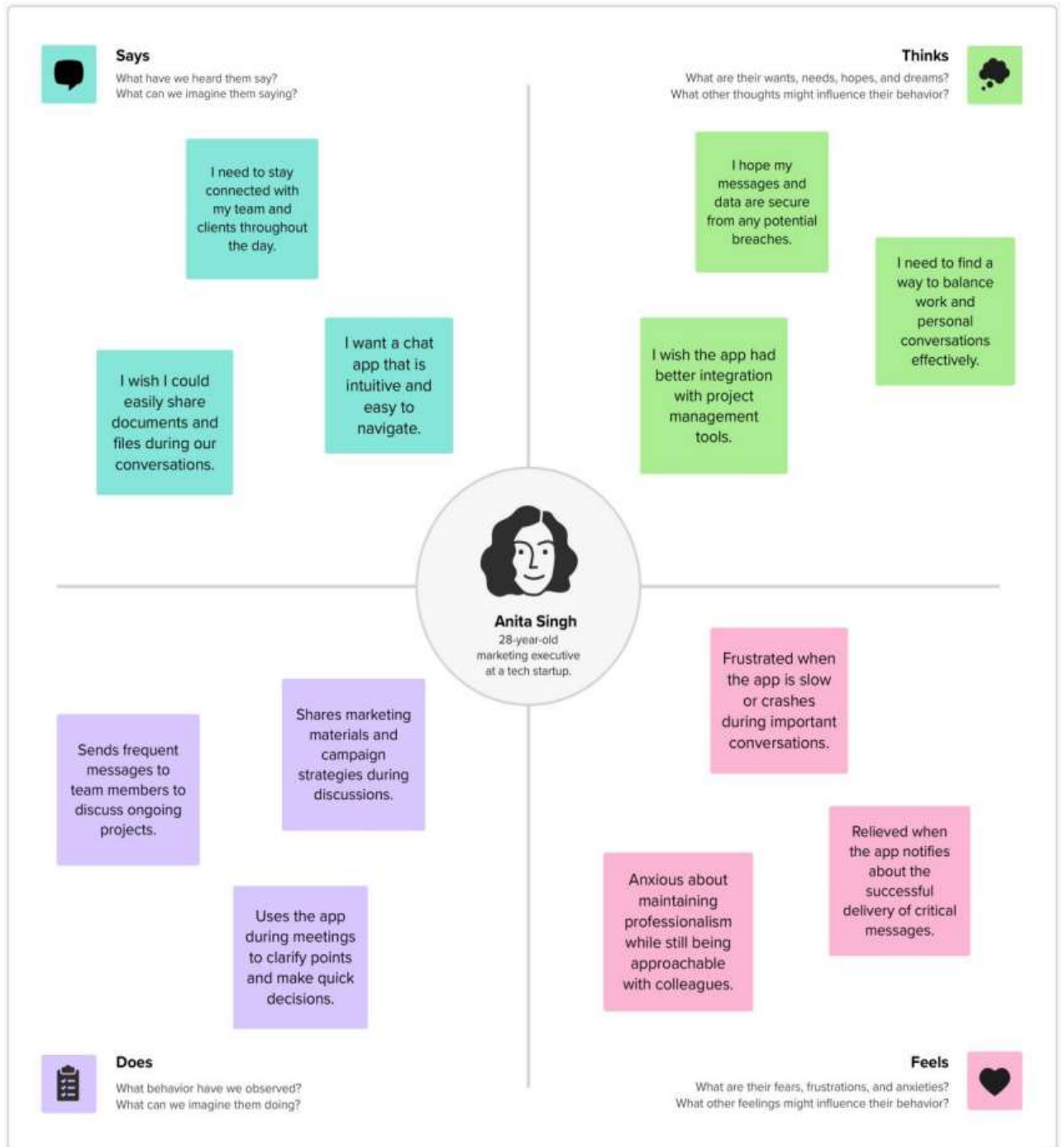
- “Web-ChatLine: An Innovative Chatting Platform” By Amit Kumar Goel, year 2022.
- “Enhanced Chat Application” By Avinash Bamane, year 2012.
- “Reporting with WhatsApp: Mobile Chat Applications’ Impact on Journalistic Practices” By Tomás Dodds, year 2019.
- “Social Network Chatting Apps Network Traffic Optimization” By Pinial Khan Butt, year 2018.

### **2.3 Problem Statement Definition**

Create and build a simple chat application that facilitates instant text communication among users. The application should offer a user-friendly interface for mobile platforms, incorporating essential features such as User Authentication, Theme Preferences, and the ability to change passwords.


### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas



### 3.2 Ideation & Brainstorming

#### Step-1: Team Gathering, Collaboration and Select the Problem Statement



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare  
🕒 1 hour to collaborate  
👥 2-8 people recommended

**➔ Before you collaborate**  
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes

---

**1 Team gathering**  
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.

**2 Set the goal**  
Think about the problem you'll be focusing on solving in the brainstorming session.

**3 Learn how to use the facilitation tools**  
Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) ➔

**1 Define your problem statement**  
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

---

**Problem**

How might we [your problem statement]?

**Key rules of brainstorming**  
To run an smooth and productive session

- 👤 Stay in topic
- 💡 Encourage wild ideas
- ⏸️ Defer judgment
- 👂 Listen to others
- 📱 Go for volume
- 👁️ If possible, be visual

#### Step-2: Brainstorm, Idea Listing and Grouping

**2**

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

**TIP** 💡

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

**Rati Dubey**

- User Profiles
- Customization Options
- Real-time Messaging
- Multimedia Support

**Agrima Sharma**

- Group Chats
- Push Notifications
- Feedback
- Chat Backup

3

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

**TIP**

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

User  
Authentication  
and Profiles

Multimedia  
Support

Real-time  
Messaging

### Step-3: Idea Prioritization

4

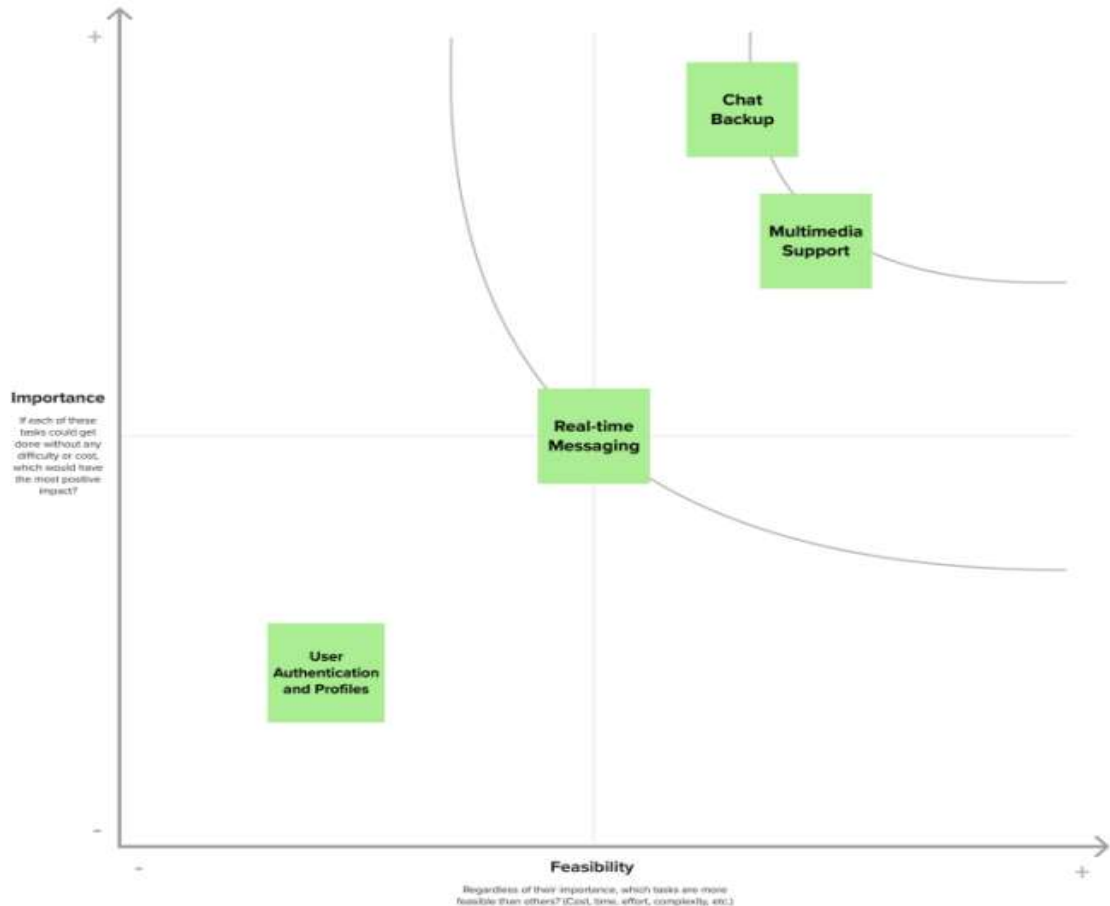
#### Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

🕒 20 minutes

**TIP**

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the user pointer holding the H key on the keyboard.



## **4. REQUIREMENT ANALYSIS**

### **4.1 Functional requirement**

1. User Registration and Authentication
  - Users should be able to create accounts with a unique username and password.
  - The app should have a secure authentication process to protect user accounts.
2. Contact Management
  - Users should be able to view contacts.
  - The app should provide a feature to search for and send friend requests.
3. Real-time Messaging
  - Users should be able to send text messages in real-time.
4. Message History
  - The app should store and display chat history.

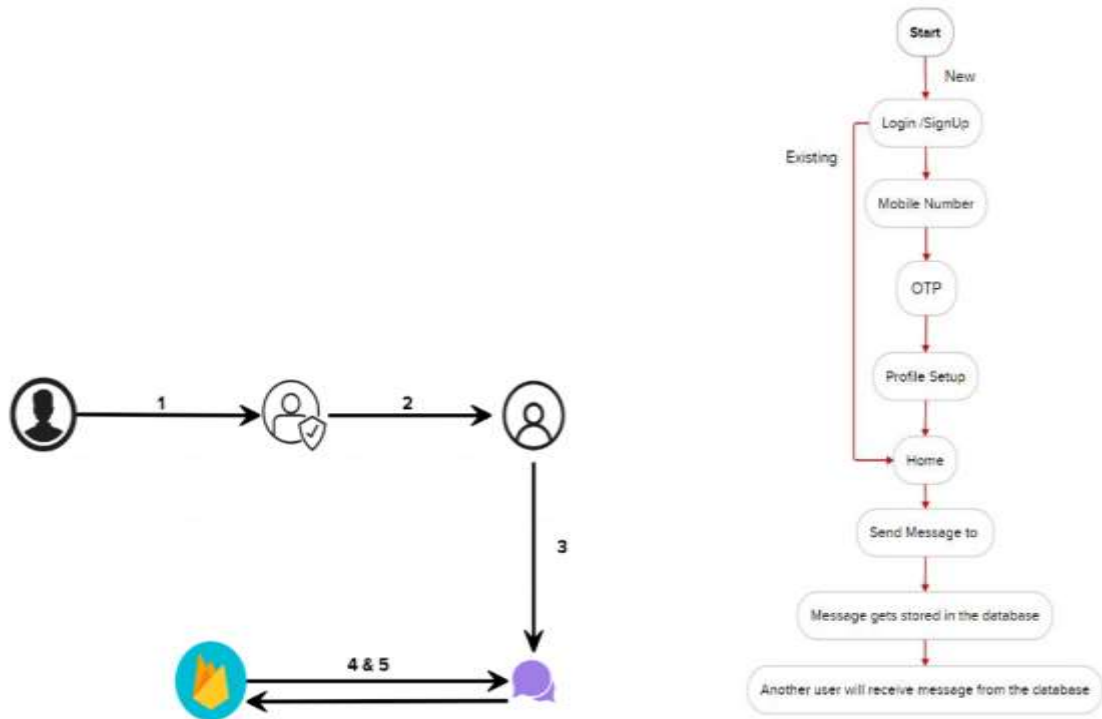
### **4.2 Non-Functional requirements**

1. Performance
  - The app should have low latency in delivering messages.
  - It should be able to handle many concurrent users.
2. Scalability
  - The system should be able to scale seamlessly as the user base grows.
  - It should support additional users and features without significant performance degradation.
3. Reliability
  - The app should have high availability, minimizing downtime.
  - It should recover gracefully from failures and ensure data integrity.
4. Usability
  - The user interface should be intuitive and user-friendly.

## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams & User Stories

#### Data Flow Diagrams:

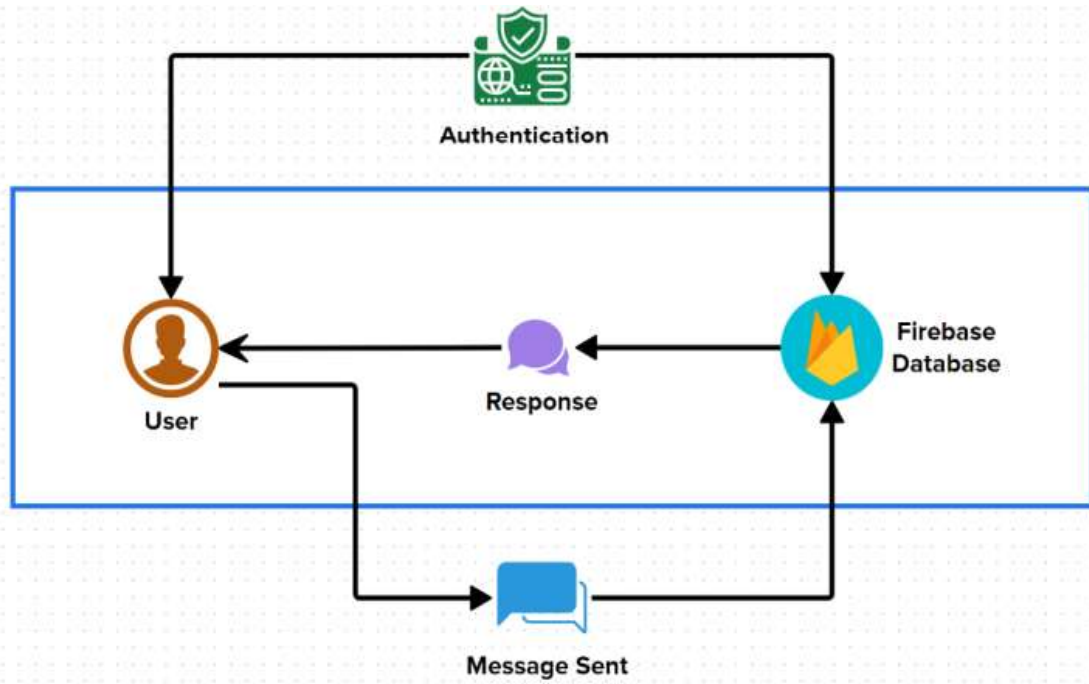


1. User Login & Two Factor Authentication Using Smartphone Generated One Time Password.
2. Profile Setup.
3. Now users can send messages to another user.
4. The message is stored in the firebase database.
5. The message is stored in the database until the other user receives the message.

#### User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by Two Factor Authentication Using Smartphone Generated One Time Password.	I can access my account / dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application after otp verification	I can directly access the home screen	High	Sprint-1
	Dashboard	USN-3	As a user, I can set my profile	I can set my profile picture/name	Medium	Sprint - 1
		USN - 4	As a user, I can select a contact and message them.	I can send select contact and message.	Medium	Sprint-1
Administrator		ASN-1	As an administrator, I can set firebase database to store contacts.	Database to store the user's data.	High	Sprint-1
			As an administrator, I can set firebase database to store messages.	Database to store messages between the users.	High	Sprint-2

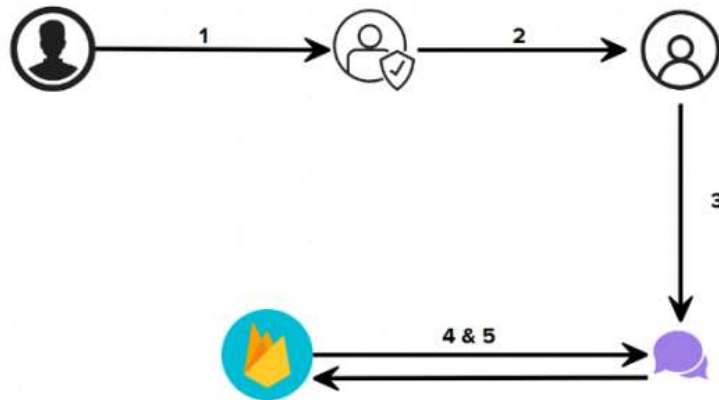
## 5.2 Solution Architecture





## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Technical Architecture



1. User Login & Two Factor Authentication Using Smartphone Generated One Time Password.
2. Profile Setup.
3. Now users can send messages to another user.
4. The message is stored in the firebase database.
5. The message is stored in the database until the other user receives the message.

### 6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering two Factor Authentication Using Smartphone Generated One Time Password	2	High	1
Sprint-1		USN-2	As a user, I will receive confirmation sms	1	High	1
Sprint-2		USN-3	As a user, I can set my profile.	1	Medium	1
Sprint-3	Dashboard	USN-5	As a user, I can select a contact and send message	1	Medium	1
		USN-5	As a user, I can receive a message from another user	1	Medium	1

### 6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	26 Oct 2023	01 Nov 2023	20	01 Nov 2023
Sprint-2	20	6 Days	01 Nov 2023	07 Nov 2023	20	07 Nov 2023
Sprint-3	20	6 Days	07 Nov 2023	07 Nov 2023	20	13 Nov 2023

## 7. CODING & SOLUTIONING

### 7.1 Real Time Messaging:

```
package com.example.spillthetea
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.LinearLayout
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
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.R
import com.google.firebase.database.ValueEventListener
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import java.lang.Exception
import java.util.ArrayList
class ChatActivity : AppCompatActivity() {
    var firebaseUser: FirebaseUser? = null
    var reference: DatabaseReference? = null
    var chatList = ArrayList<Chat>()
    var topic = ""
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat)
        chatRecyclerView.layoutManager = LinearLayoutManager(this, LinearLayout.VERTICAL, false)
        var intent = getIntent()
        var userId = intent.getStringExtra("userId")
        var userName = intent.getStringExtra("userName")
        imgBack.setOnClickListener {
            onBackPressed()
        }
        firebaseUser = FirebaseAuth.getInstance().currentUser
        reference = FirebaseDatabase.getInstance().getReference("Users").child(userId!!)
        reference!!.addValueEventListener(object : ValueEventListener {
            override fun onCancelled(error: DatabaseError) {
                TODO("Not yet implemented")
            }
        })
        override fun onDataChange(snapshot: DataSnapshot) {
            val user = snapshot.getValue(User::class.java)
            tvUserName.text = user!!.userName
            if (user.profileImage == "") {
                imgProfile.setImageResource(R.drawable.profile_image)
            } else {
                Glide.with(this@ChatActivity).load(user.profileImage).into(imgProfile)
            }
        }
    }
}
```

```

    })
    btnSendMessage.setOnClickListener {
        var message: String = etMessage.text.toString()
        if (message.isEmpty()) {
            Toast.makeText(applicationContext, "message is empty", Toast.LENGTH_SHORT).show()
            etMessage.setText("")
        } else {
            sendMessage(firebaseUser!!.uid, userId, message)
            etMessage.setText("")
            topic = "/topics/$userId"
        }
    }
}
readMessage(firebaseUser!!.uid, userId)
}

private fun sendMessage(senderId: String, receiverId: String, message: String) {
    var reference: DatabaseReference? = FirebaseDatabase.getInstance().getReference()
    var hashMap: HashMap<String, String> = HashMap()
    hashMap.put("senderId", senderId)
    hashMap.put("receiverId", receiverId)
    hashMap.put("message", message)
    reference!!.child("Chat").push().setValue(hashMap)
}

fun readMessage(senderId: String, receiverId: String) {
    val databaseReference: DatabaseReference =
        FirebaseDatabase.getInstance().getReference("Chat")
    databaseReference.addValueEventListener(object : ValueEventListener {
        override fun onCancelled(error: DatabaseError) {
            TODO("Not yet implemented")
        }
        override fun onDataChange(snapshot: DataSnapshot) {
            chatList.clear()
            for (dataSnapshot: DataSnapshot in snapshot.children) {
                val chat = dataSnapshot.getValue(Chat::class.java)
                if (chat!!.senderId.equals(senderId) && chat!!.receiverId.equals(receiverId) ||
                    chat!!.senderId.equals(receiverId) && chat!!.receiverId.equals(senderId)
                ) {
                    chatList.add(chat)
                }
            }
            val chatAdapter = ChatAdapter(this@ChatActivity, chatList)
            chatRecyclerView.adapter = chatAdapter
        }
    })
}
}
}



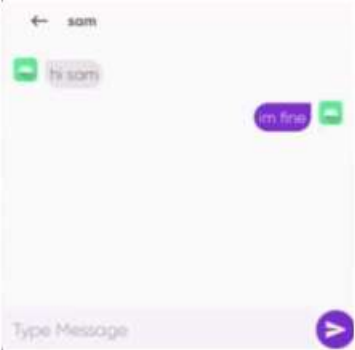
```

## 7.2 User Security:

```
package com.example.spillthetea
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.example.spillthetea.databinding.ActivityVerificationBinding
import com.google.firebase.auth.FirebaseAuth
class VerificationActivity : AppCompatActivity() {
    var binding : ActivityVerificationBinding? = null
    var auth: FirebaseAuth? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityVerificationBinding.inflate(layoutInflater)
        setContentView(binding!!.root)
        auth = FirebaseAuth.getInstance()
        if (auth!!.currentUser != null){
            val intent = Intent(this@VerificationActivity,MainActivity::class.java)
            startActivity(intent)
            finish()
        }
        supportActionBar?.hide()
        binding!!.editNumber.requestFocus()
        binding!!.continueBtn.setOnClickListener {
            val intent = Intent(this@VerificationActivity,OTPActivity::class.java)
            intent.putExtra("phoneNumber",binding!!.editNumber.text.toString())
            startActivity(intent)
        }
    }
}
```

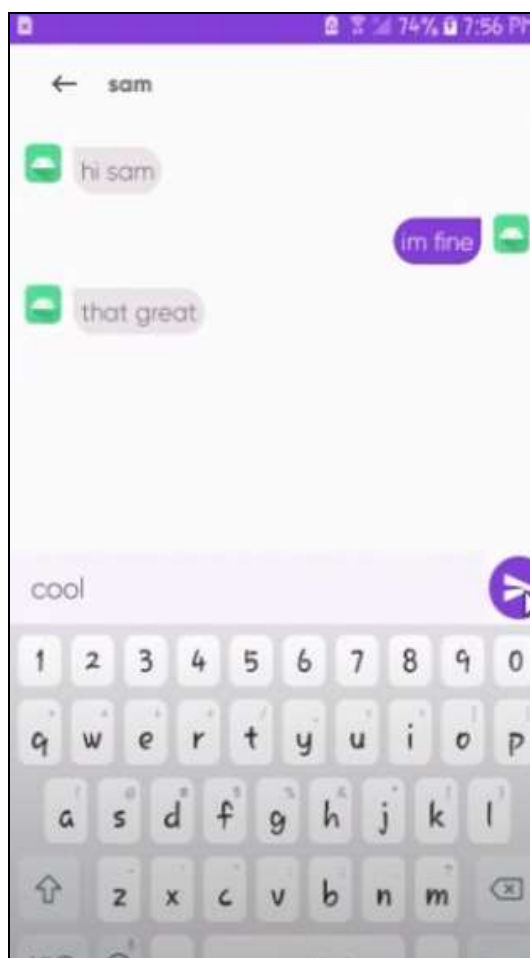
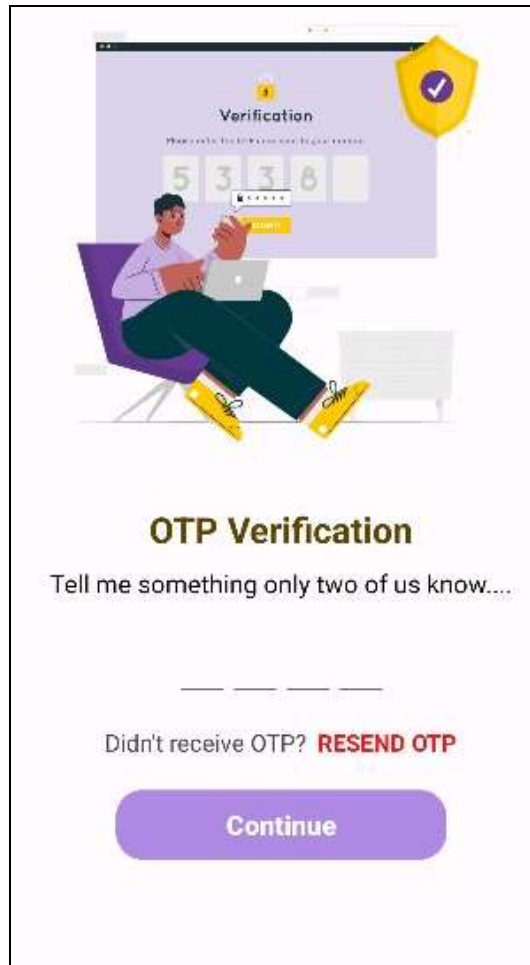
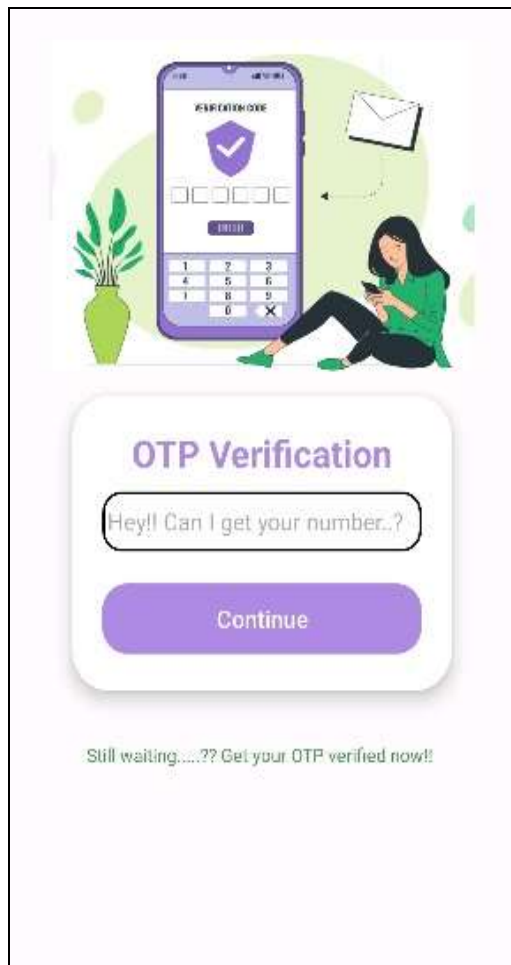
8. PERFORMANCE TESTING

8.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
	Metrics	App Launch Time-26s Screen Render Time - 60ms	<div></div> <div>App launch time</div>
	Usage	App Size-5.7mb Customer Experience-	<div></div> <div></div>
	Performance	Error and Crash Rates -0	none

## 9. RESULTS

### 9.1 Output Screenshots



## 10. ADVANTAGES & DISADVANTAGES

### Advantages:

1. **Real-Time Communication:** Users can engage in instant, real-time conversations, fostering quick and efficient communication.
2. **Global Connectivity:** Enables communication across geographic boundaries, allowing users to connect with friends, family, or colleagues anywhere in the world.
3. **Cost-Effective:** Chat apps often provide a cost-effective alternative to traditional messaging or phone calls, especially for international communication.
4. **Multimedia Sharing:** Users can share a wide range of multimedia files, including images, videos, and documents, enhancing the richness of communication.
5. **Group Collaboration:** Group chat features facilitate collaboration among teams or communities, streamlining discussions and information sharing.
6. **Notification Alerts:** Push notifications keep users informed about new messages and activities, ensuring timely responses.
7. **Convenience and Accessibility:** Offers a convenient and easily accessible platform for communication, available on various devices with internet connectivity.
8. **Record of Conversations:** Chat apps typically maintain a record of conversations, allowing users to review past interactions and retrieve information.
9. **User Engagement:** Emojis, stickers, and other expressive features enhance user engagement, making conversations more dynamic and enjoyable.
10. **Integration with Other Services:** Some chat apps integrate with external services, providing additional functionalities like file sharing, task management, or calendar synchronization.

### Disadvantages:

1. **Privacy Concerns:** The potential for privacy breaches, especially if the app lacks robust security measures or if users are not cautious about sharing sensitive information.
2. **Distraction and Overuse:** Continuous notifications and constant connectivity may contribute to distraction and overuse, impacting productivity and personal time.
3. **Security Vulnerabilities:** Some chat apps may be susceptible to security vulnerabilities, exposing users to risks such as hacking, phishing, or malware attacks.
4. **Limited Emotional Nuance:** Text-based communication may lack the nuance of face-to-face conversations, leading to potential misunderstandings or misinterpretations of tone.
5. **Dependency on Internet Connection:** Requires a stable internet connection, and disruptions in connectivity can hinder communication.
6. **Fragmentation and Multiple Apps:** Users may need to use multiple chat apps to communicate with different groups, leading to fragmentation and potential confusion.
7. **Data Usage:** Extensive use of multimedia files and constant syncing can contribute to increased data usage, impacting users with limited data plans.
8. **User Interface Issues:** Some chat apps may have complex or unintuitive user interfaces, leading to a less-than-optimal user experience.
9. **Potential for Misuse:** Chat apps can be misused for cyberbullying, harassment, or spreading false information, highlighting the need for effective moderation and reporting mechanisms.
10. **Overreliance on Technology:** A potential overreliance on digital communication may lead to reduced face-to-face interactions, impacting social skills and relationships.

## **11. CONCLUSION**

In summary, a messaging application presents numerous advantages in terms of immediate communication, global connectivity, and cost-effectiveness. Users can partake in real-time conversations, fostering a lively and interactive communication experience. The widespread availability of these applications on various devices enables users to stay connected at any time and from any location.

Nevertheless, it is crucial to be cautious of potential drawbacks associated with messaging apps. Concerns about privacy, reliance on internet connectivity, and the risk of security issues emphasize the importance of implementing robust measures to safeguard user data. Additionally, the potential for misunderstandings and the absence of face-to-face interaction can influence the overall quality of communication.

As technology progresses, developers will concentrate on addressing these challenges through enhanced security features, improved user education, and considerate design that prioritizes both functionality and user experience. Ultimately, a well-designed fundamental messaging app has the potential to significantly improve communication and connectivity, as long as it aligns with user expectations and maintains a balance between convenience and security.



## 12. FUTURE SCOPE

The future scope of a real-time chat and communication app like ChatConnect could be vast and dynamic, as it largely depends on the evolving needs of users and technological advancements. Here are some potential directions for the future development and expansion of ChatConnect:

1. Integration of Emerging Technologies:
  - AI and Chatbots: Enhance user experience by incorporating advanced AI and natural language processing for smarter and more interactive conversations.
  - Augmented Reality (AR) and Virtual Reality (VR): Implement AR/VR features for immersive communication experiences, such as virtual meetings or shared virtual spaces.
2. Multi-Platform Compatibility:
  - Extend the app's compatibility across various devices and platforms, including smartphones, tablets, desktops, smartwatches, and smart TVs.
3. Enhanced Security and Privacy:
  - Strengthen security measures to ensure end-to-end encryption, secure file sharing, and protection against cyber threats.
  - Implement privacy features, giving users more control over their data and communication settings.
4. Collaboration and Productivity Features:
  - Integrate collaboration tools like document sharing, real-time editing, and project management features to make ChatConnect a comprehensive platform for work and productivity.
5. Voice and Video Improvements:
  - Improve voice and video call quality and introduce innovative features such as 3D audio, background effects, or real-time language translation during conversations.
6. Localization and Global Expansion:
  - Localize the app for different languages and cultures, making it more accessible and user-friendly for a global audience.
  - Expand server infrastructure to ensure low-latency communication across various regions.
7. Personalization and Customization:
  - Provide users with more customization options for their profiles, chat themes, and notification settings.
  - Implement machine learning algorithms to understand user preferences and tailor the app's interface accordingly.
8. Integration with Ecosystem:
  - Integrate ChatConnect with other popular apps, services, and platforms to create a seamless ecosystem for users, such as integrating with calendar apps, social media platforms, or task management tools.
9. Monetization Strategies:
  - Explore various monetization models, such as premium features, subscription plans, or targeted advertising (while respecting user privacy), to sustain the app's development and growth.
10. User Feedback and Community Engagement:
  - Continuously gather user feedback to identify areas for improvement and new feature requests.
  - Foster a strong community around the app, encouraging user engagement through forums, support channels, and user-generated content.

Remember, the success of ChatConnect in the future will depend on staying adaptive to user needs, technological advancements, and market trends. Regular updates, innovation, and a responsive approach to user feedback will be key in ensuring the app's sustained relevance and success.

## 13. APPENDIX

### Source Code

#### a. MainActivity.kt

```
package com.example.spillthetea
import android.app.ProgressDialog
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.recyclerview.widget.GridLayoutManager
import com.example.spillthetea.adapter.UserAdapter
import com.example.spillthetea.databinding.ActivityMainBinding
import com.example.spillthetea.model.User
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.ValueEventListener
class MainActivity : AppCompatActivity() {
    var binding : ActivityMainBinding? = null
    var database: FirebaseDatabase? = null
    var users:ArrayList<User>? = null
    var usersAdapter: UserAdapter? = null
    var dialog: ProgressDialog? =null
    var user:User? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding!!.root)
        dialog = ProgressDialog(this@MainActivity)
        dialog!!.setMessage("Uploading Image...")
        dialog!!.setCancelable(false)
        database = FirebaseDatabase.getInstance()
        users = ArrayList<User>()
        usersAdapter = UserAdapter(this@MainActivity, users!!)
        val layoutManager = GridLayoutManager(this@MainActivity, 2)
        binding!!.mRec.layoutManager = layoutManager
        database!!.reference.child("users")
            .child(FirebaseAuth.getInstance().uid!!)
            .addValueEventListener(object : ValueEventListener {
                override fun onDataChange(snapshot: DataSnapshot) {
                    user = snapshot.getValue(User::class.java)
                }
                override fun onCancelled(error: DatabaseError){}
            })
        binding!!.mRec.adapter = usersAdapter
        database!!.reference.child("users").addValueEventListener(object :ValueEventListener{
            override fun onDataChange(snapshot: DataSnapshot) {
                users!!.clear()
                for (snapshot1 in snapshot.children) {
                    val user: User? = snapshot1.getValue(User::class.java)
                    if (!user!!.uid.equals(FirebaseAuth.getInstance().uid)) users!!.add(user)
                }
            }
        })
    }
}
```

```
        override fun onCancelled(error: DatabaseError) {}
    })
}
override fun onResume() {
    super.onResume()
    val currentId = FirebaseAuth.getInstance().uid
    database!!.reference.child("presence")
        .child(currentId!!).setValue("Online")
}
override fun onPause() {
    super.onPause()
    val currentId = FirebaseAuth.getInstance().uid
    database!!.reference.child("presence")
        .child(currentId!!).setValue("Offline")
}
}
```

**b. ChatActivity.kt**

```
package com.example.spillthetea
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.LinearLayout
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
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.R
import com.google.firebase.database.ValueEventListener
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import java.lang.Exception
import java.util.ArrayList
class ChatActivity : AppCompatActivity() {
    var firebaseUser: FirebaseUser? = null
    var reference: DatabaseReference? = null
    var chatList = ArrayList<Chat>()
    var topic = ""
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat)
        chatRecyclerView.layoutManager = LinearLayoutManager(this, LinearLayout.VERTICAL, false)
        var intent = getIntent()
        var userId = intent.getStringExtra("userId")
        var userName = intent.getStringExtra("userName")
        imgBack.setOnClickListener {
            onBackPressed()
        }
        firebaseUser = FirebaseAuth.getInstance().currentUser
        reference = FirebaseDatabase.getInstance().getReference("Users").child(userId!!)
        reference!!.addValueEventListener(object : ValueEventListener {
            override fun onCancelled(error: DatabaseError) {
                TODO("Not yet implemented")
            }
        })
        override fun onDataChange(snapshot: DataSnapshot) {
            val user = snapshot.getValue(User::class.java)
            tvUserName.text = user!!.userName
            if (user.profileImage == "") {
                imgProfile.setImageResource(R.drawable.profile_image)
            } else {
                Glide.with(this@ChatActivity).load(user.profileImage).into(imgProfile)
            }
        }
    }
}
```

```

btnSendMessage.setOnClickListener {
    var message: String = etMessage.text.toString()
    if (message.isEmpty()) {
        Toast.makeText(applicationContext, "message is empty", Toast.LENGTH_SHORT).show()
        etMessage.setText("")
    } else {
        sendMessage(firebaseUser!!.uid, userId, message)
        etMessage.setText("")
        topic = "/topics/$userId"
    }
}
readMessage(firebaseUser!!.uid, userId)
}

private fun sendMessage(senderId: String, receiverId: String, message: String) {
    var reference: DatabaseReference? = FirebaseDatabase.getInstance().getReference()
    var hashMap: HashMap<String, String> = HashMap()
    hashMap.put("senderId", senderId)
    hashMap.put("receiverId", receiverId)
    hashMap.put("message", message)
    reference!!.child("Chat").push().setValue(hashMap)
}

fun readMessage(senderId: String, receiverId: String) {
    val databaseReference: DatabaseReference =
        FirebaseDatabase.getInstance().getReference("Chat")
    databaseReference.addValueEventListener(object : ValueEventListener {
        override fun onCancelled(error: DatabaseError) {
            TODO("Not yet implemented")
        }
        override fun onDataChange(snapshot: DataSnapshot) {
            chatList.clear()
            for (dataSnapshot: DataSnapshot in snapshot.children) {
                val chat = dataSnapshot.getValue(Chat::class.java)
                if (chat!!.senderId.equals(senderId) && chat!!.receiverId.equals(receiverId) ||
                    chat!!.senderId.equals(receiverId) && chat!!.receiverId.equals(senderId)
                ) {
                    chatList.add(chat)
                }
            }
            val chatAdapter = ChatAdapter(this@ChatActivity, chatList)
            chatRecyclerView.adapter = chatAdapter
        }
    })
}
}

```

**c. OTPActivity.kt**

```
package com.example.spillthetea
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.LinearLayout
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.bumptech.glide.Glide
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
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.R
import com.google.firebase.database.ValueEventListener
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import java.lang.Exception
import java.util.ArrayList
class ChatActivity : AppCompatActivity() {
    var firebaseUser: FirebaseUser? = null
    var reference: DatabaseReference? = null
    var chatList = ArrayList<Chat>()
    var topic = ""
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_chat)
        chatRecyclerView.layoutManager = LinearLayoutManager(this, LinearLayout.VERTICAL, false)
        var intent = getIntent()
        var userId = intent.getStringExtra("userId")
        var userName = intent.getStringExtra("userName")
        imgBack.setOnClickListener {
            onBackPressed()
        }
        firebaseUser = FirebaseAuth.getInstance().currentUser
        reference = FirebaseDatabase.getInstance().getReference("Users").child(userId!!)
        reference!!.addValueEventListener(object : ValueEventListener {
            override fun onCancelled(error: DatabaseError) {
                TODO("Not yet implemented")
            }
        })
        override fun onDataChange(snapshot: DataSnapshot) {
            val user = snapshot.getValue(User::class.java)
            tvUserName.text = user!!.userName
            if (user.profileImage == "") {
                imgProfile.setImageResource(R.drawable.profile_image)
            } else {
                Glide.with(this@ChatActivity).load(user.profileImage).into(imgProfile)
            }
        }
    }
}
```

```

btnSendMessage.setOnClickListener {
    var message: String = etMessage.text.toString()
    if (message.isEmpty()) {
        Toast.makeText(applicationContext, "message is empty", Toast.LENGTH_SHORT).show()
        etMessage.setText("")
    } else {
        sendMessage(firebaseUser!!.uid, userId, message)
        etMessage.setText("")
        topic = "/topics/$userId"
    }
}
readMessage(firebaseUser!!.uid, userId)
}

private fun sendMessage(senderId: String, receiverId: String, message: String) {
    var reference: DatabaseReference? = FirebaseDatabase.getInstance().getReference()
    var hashMap: HashMap<String, String> = HashMap()
    hashMap.put("senderId", senderId)
    hashMap.put("receiverId", receiverId)
    hashMap.put("message", message)
    reference!!.child("Chat").push().setValue(hashMap)
}

fun readMessage(senderId: String, receiverId: String) {
    val databaseReference: DatabaseReference =
        FirebaseDatabase.getInstance().getReference("Chat")
    databaseReference.addValueEventListener(object : ValueEventListener {
        override fun onCancelled(error: DatabaseError) {
            TODO("Not yet implemented")
        }
        override fun onDataChange(snapshot: DataSnapshot) {
            chatList.clear()
            for (dataSnapshot: DataSnapshot in snapshot.children) {
                val chat = dataSnapshot.getValue(Chat::class.java)
                if (chat!!.senderId.equals(senderId) && chat!!.receiverId.equals(receiverId) ||
                    chat!!.senderId.equals(receiverId) && chat!!.receiverId.equals(senderId)
                ) {
                    chatList.add(chat)
                }
            }
            val chatAdapter = ChatAdapter(this@ChatActivity, chatList)
            chatRecyclerView.adapter = chatAdapter
        }
    })
}
}

```

#### d. SetupProfileActivity.kt

```
package com.example.spillthetea
import android.app.ProgressDialog
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.example.spillthetea.databinding.ActivitySetupProfileBinding
import com.example.spillthetea.model.User
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.storage.FirebaseStorage
import java.util.Date

class SetupProfileActivity : AppCompatActivity() {
    var binding: ActivitySetupProfileBinding? = null
    var auth: FirebaseAuth? = null
    var database: FirebaseDatabase? = null
    var storage: FirebaseStorage? = null
    var selectedImage: Uri? = null
    var dialog: ProgressDialog? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySetupProfileBinding.inflate(layoutInflater)
        setContentView(binding!!.root)
        dialog!!.setMessage("Updating Profile...")
        dialog!!.setCancelable(false)
        database = FirebaseDatabase.getInstance()
        storage = FirebaseStorage.getInstance()
        auth = FirebaseAuth.getInstance()
        supportActionBar?.hide()
        binding!!.imageView.setOnClickListener {
            val intent = Intent()
            intent.action = Intent.ACTION_GET_CONTENT
            intent.type = "image/*"
            startActivityForResult(intent, 45)
        }
        binding!!.continueBtn02.setOnClickListener {
            val name: String = binding!!.nameBox.text.toString()
            if (name.isEmpty()) {
                binding!!.nameBox.setError("Please type a name")
            }
            dialog!!.show()
            if (selectedImage != null) {
                val reference = storage!!.reference.child("Profile")
                    .child(auth!!.uid!!)
                reference.putFile(selectedImage!!).addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        reference.downloadUrl.addOnCompleteListener { uri ->
                            val imageUrl = uri.toString()
                            val uid = auth!!.uid
                            val phone = auth!!.currentUser!!.phoneNumber
                            val name: String = binding!!.nameBox.text.toString()
                            val user = User(uid, name, phone, imageUrl)
```



```

        database!!.reference
            .child("users")
            .child(uid!!)
            .setValue(user)
            .addOnCompleteListener {
                dialog!!.dismiss()
                val intent =
                    Intent(this@SetupProfileActivity,
                        MainActivity::class.java)
                startActivity(intent)
                finish()
            }
        }
    }
}
else {
    val uid = auth!!.uid
    val phone = auth!!.currentUser!!.phoneNumber
    val name: String = binding!!.nameBox.text.toString()
    val user = User(uid, name, phone, "No Image")
    database!!.reference
        .child("users")
        .child(uid!!)
        .setValue(user)
        .addOnCanceledListener {
            dialog!!.dismiss()
            val intent =
                Intent(this@SetupProfileActivity, MainActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
}
}
}
}
}
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (data != null) {
        if (data.data != null) {
            val uri = data.data!!.filePath
            val storage = FirebaseStorage.getInstance()
            val time = Date().time
            val reference = storage.reference
                .child("Profile")
                .child(time.toString() + "")
            reference.putFile(uri!!).addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    reference.downloadUrl.addOnCompleteListener { uri ->
                        val filePath = uri.toString()
                        val obj = HashMap<String, Any>()
                        obj["image"] = filePath
                        database!!.reference
                            .child("users")

```

```
        .child(FirebaseAuth.getInstance().uid!!)
        .updateChildren(obj).addOnSuccessListener { }
    }
}
binding!!.imageView.setImageURI(data.data)
selectedImage = data.data
}
}
}
```

**e. VerificationActivity.kt**

```
package com.example.spillthetea
import android.app.ProgressDialog
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.example.spillthetea.databinding.ActivitySetupProfileBinding
import com.example.spillthetea.model.User
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.storage.FirebaseStorage
import java.util.Date
class SetupProfileActivity : AppCompatActivity() {
    var binding:ActivitySetupProfileBinding? = null
    var auth:FirebaseAuth? = null
    var database:FirebaseDatabase? = null
    var storage:FirebaseStorage? = null
    var selectedImage:Uri? = null
    var dialog:ProgressDialog? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySetupProfileBinding.inflate(layoutInflater)
        setContentView(binding!!.root)
        dialog!!.setMessage("Updating Profile...")
        dialog!!.setCancelable(false)
        database = FirebaseDatabase.getInstance()
        storage = FirebaseStorage.getInstance()
        auth = FirebaseAuth.getInstance()
        supportActionBar?.hide()
        binding!!.imageView.setOnClickListener {
            val intent = Intent()
            intent.action = Intent.ACTION_GET_CONTENT
            intent.type = "image/*"
            startActivityForResult(intent, 45)
        }
        binding!!.continueBtn02.setOnClickListener {
            val name: String = binding!!.nameBox.text.toString()
            if (name.isEmpty()) {
                binding!!.nameBox.setError("Please type a name")
            }
            dialog!!.show()
            if (selectedImage != null) {
                val reference = storage!!.reference.child("Profile")
                    .child(auth!!.uid!!)
                reference.putFile(selectedImage!!).addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        reference.downloadUrl.addOnCompleteListener { uri ->
                            val imageUrl = uri.toString()
                            val uid = auth!!.uid
                            val phone = auth!!.currentUser!!.phoneNumber
                            val name: String = binding!!.nameBox.text.toString()
                            val user = User(uid, name, phone, imageUrl)
```

```

        database!!.reference
            .child("users")
            .child(uid!!)
            .setValue(user)
            .addOnCompleteListener {
                dialog!!.dismiss()
                val intent =
                    Intent(this@SetupProfileActivity,
                        MainActivity::class.java)
                startActivity(intent)
                finish()
            }
        }
    }
}
else {
    val uid = auth!!.uid
    val phone = auth!!.currentUser!!.phoneNumber
    val name: String = binding!!.nameBox.text.toString()
    val user = User(uid, name, phone, "No Image")
    database!!.reference
        .child("users")
        .child(uid!!)
        .setValue(user)
        .addOnCanceledListener {
            dialog!!.dismiss()
            val intent =
                Intent(this@SetupProfileActivity, MainActivity::class.java)
            startActivity(intent)
            finish()
        }
    }
}
}
}
}
}
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (data != null) {
        if (data.data != null) {
            val uri = data.data!!.filePath
            val storage = FirebaseStorage.getInstance()
            val time = Date().time
            val reference = storage.reference
                .child("Profile")
                .child(time.toString() + "")
            reference.putFile(uri!!).addOnCompleteListener { task ->
                if (task.isSuccessful) {
                    reference.downloadUrl.addOnCompleteListener { uri ->
                        val filePath = uri.toString()
                        val obj = HashMap<String, Any>()
                        obj["image"] = filePath
                        database!!.reference
                            .child("users")

```

```
        .child(FirebaseAuth.getInstance().uid!!)
        .updateChildren(obj).addOnSuccessListener { }
    }
}
binding!!.imageView.setImageURI(data.data)
selectedImage = data.data
}
}
}
```

**f. ChatAdapter.kt**

```
package com.example.spillthetea.adapter
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import de.hdodenhof.circleimageview.CircleImageView
class ChatAdapter(private val context: Context, private val chatList: ArrayList<Chat>) :
    RecyclerView.Adapter<ChatAdapter.ViewHolder>() {
    private val MESSAGE_TYPE_LEFT = 0
    private val MESSAGE_TYPE_RIGHT = 1
    var firebaseUser: FirebaseUser? = null
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        if (viewType == MESSAGE_TYPE_RIGHT) {
            val view =
                LayoutInflater.from(parent.context).inflate(R.layout.item_right, parent, false)
            return ViewHolder(view)
        } else {
            val view =
                LayoutInflater.from(parent.context).inflate(R.layout.item_left, parent, false)
            return ViewHolder(view)
        }
    }
    override fun getItemCount(): Int {
        return chatList.size
    }
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val chat = chatList[position]
        holder.txtUserName.text = chat.message
        //Glide.with(context).load(user.profileImage).placeholder(R.drawable.profile_image).into(holder.imgUser)
    }
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val txtUserName: TextView = view.findViewById(R.id.tvMessage)
        val imgUser: CircleImageView = view.findViewById(R.id.userImage)
    }
    override fun getItemViewType(position: Int): Int {
        firebaseUser = FirebaseAuth.getInstance().currentUser
        if (chatList[position].senderId == firebaseUser!!.uid) {
            return MESSAGE_TYPE_RIGHT
        } else {
            return MESSAGE_TYPE_LEFT
        }
    }
}
```

**g. UserAdapter.kt**

```
package com.example.spillthetea.adapter
import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.FirebaseUser
import de.hdodenhof.circleimageview.CircleImageView

class ChatAdapter(private val context: Context, private val chatList: ArrayList<Chat>) :
    RecyclerView.Adapter<ChatAdapter.ViewHolder>() {
    private val MESSAGE_TYPE_LEFT = 0
    private val MESSAGE_TYPE_RIGHT = 1
    var firebaseUser: FirebaseUser? = null
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        if (viewType == MESSAGE_TYPE_RIGHT) {
            val view =
                LayoutInflater.from(parent.context).inflate(R.layout.item_right, parent, false)
            return ViewHolder(view)
        } else {
            val view =
                LayoutInflater.from(parent.context).inflate(R.layout.item_left, parent, false)
            return ViewHolder(view)
        }
    }
    override fun getItemCount(): Int {
        return chatList.size
    }
    override fun onBindViewHolder(holder: ViewHolder, position: Int) {
        val chat = chatList[position]
        holder.txtUserName.text = chat.message
        //Glide.with(context).load(user.profileImage).placeholder(R.drawable.profile_image).into(holder.imgUser)
    }
    class ViewHolder(view: View) : RecyclerView.ViewHolder(view) {
        val txtUserName: TextView = view.findViewById(R.id.tvMessage)
        val imgUser: CircleImageView = view.findViewById(R.id.userImage)
    }
    override fun getItemViewType(position: Int): Int {
        firebaseUser = FirebaseAuth.getInstance().currentUser
        if (chatList[position].senderId == firebaseUser!!.uid) {
            return MESSAGE_TYPE_RIGHT
        } else {
            return MESSAGE_TYPE_LEFT
        }
    }
}
```

## **GitHub & Project Demo Link**

**GitHub Link:** <https://github.com/smartinternz02/SI-GuidedProject-591101-1697212905>

**Project Demo Link:** [https://drive.google.com/file/d/1It\\_fCNfMOvV7QPZpp85IAFg8lbs4nID/view?usp=sharing](https://drive.google.com/file/d/1It_fCNfMOvV7QPZpp85IAFg8lbs4nID/view?usp=sharing)