

**ENCHAT: SECURED & REALTIME MESSAGING ANDROID APPLICATION**

**BY**

**S. M. MOHIBBULLAH**

**ID: 191-15-12834**

**MD OMUR FARUK REDOY**

**ID: 191-15-13023**

**AND**

**MOHAIMENUL ISLAM MUBIN**

**ID: 191-15-12820**

This Report is presented in Partial Fulfillment of the Requirements for the  
Degree of Bachelor of Science in Computer Science and Engineering

Supervised By

**Mr. Raja Tariqul Hasan Tusher**

Assistant Professor

Department of CSE

Daffodil International University

Co-Supervised By

**Md. Jueal Mia**

Assistant Professor

Department of CSE

Daffodil International University



**DAFFODIL INTERNATIONAL UNIVERSITY**

**DHAKA, BANGLADESH**

**JANUARY 2023**

## **APPROVAL**

This Project/internship titled “**EnChat: Secured & Realtime Messaging Android Application**”, submitted by S.M. Mohibullah, ID No.: 191-15-12834, Md Omur Faruk Redoy, ID No.: 191-15-13023 and Mohaimenul Islam Mubin, ID No.: 191-15-12820 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfilment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on *28-01-2023*.

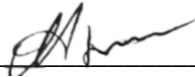
### **BOARD OF EXAMINERS**

**Chairman**

\_\_\_\_\_  
**Dr. Touhid Bhuiyan**

**Professor and Head**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University



**Internal Examiner**

\_\_\_\_\_  
**Nazmun Nessa Moon**

**Associate Professor**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University

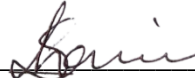


**Internal Examiner**

\_\_\_\_\_  
**Zakia Sultana**

**Senior Lecturer**

Department of Computer Science and Engineering  
Faculty of Science & Information Technology  
Daffodil International University



**External Examiner**

\_\_\_\_\_  
**Dr. Shamim H Ripon**

**Professor**

Department of Computer Science and Engineering  
East West University

## DECLARATION

We hereby declare that this project has been done by us under the supervision of Mr. Raja Tariqul Hasan Tusher, Assistant Professor, Department of CSE Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

### Supervised By



---

**Mr. Raja Tariqul Hasan Tusher**  
Assistant Professor  
Department of CSE  
Daffodil International University

### Co-Supervised By



---

**Md. Jueal Mia**  
Assistant Professor  
Department of CSE  
Daffodil International University

### Submitted by:



---

**S.M Mohibbullah**  
ID: 191-15-12834  
Department of CSE  
Daffodil International University



---

**Md Omur Faruk Redoy**  
ID: 191-15-13023  
Department of CSE  
Daffodil International University



---

**Mohaimenul Islam Mubin**  
ID: 191-15-12820  
Department of CSE  
Daffodil International University

## ACKNOWLEDGEMENT

We express our sincere thanks and gratitude to the almighty creator for his infinite mercy due to which we have been able to successfully complete our final year project on time.

The true spirit of reaching a goal is via excellence and good discipline. We could not have completed our work without the participation, support, and assistance of numerous individuals.

We are truly grateful and deeply indebted to **Mr. Raja Tariqul Hasan Tusher, Assistant Professor, CSE** Daffodil International University, Dhaka. His support and suggestions helped us to complete the work properly. This project was made possible by his endless endurance, academic leadership, constant encouragement, constant and energetic supervision, constructive criticism, invaluable suggestions, and numerous reviews and revisions at every stage.

We would like to express our heartfelt gratitude to **Professor Dr. Touhid Bhuiyan, Head, Department of CSE**, for his kind assistance in completing our project, and we would also like to acknowledge with gratitude the critical role of Daffodil International University (DIU) staff, who granted me access to all types of library materials and equipment to gain knowledge and clear out our understandings. We must thank the other supervisors and lecturers for their assistance in clarifying our concept and instilling in us the need of carefully completing the project report while maintaining high knowledge and quality.

We would like to thank everyone of our Daffodil International University classmates who participated in this discussion while completing their course work. Finally, we must express our gratitude for our parents' continuous support and patience.

## **ABSTRACT**

Humans have made a lot of progress in terms of science and technology. With the advancement of science and technology, many improvements have been made in the communication system. Nowadays, messaging apps are one of the most important means of communication. Using messaging apps has made people's communication easier. We have built our app to facilitate human communication and to provide various facilities for communication. People will benefit from using our app to communicate with each other easily. This app built by us is completely free and ad-free. In this app, users can open accounts and communicate with each other using their mobile numbers. In addition to messaging without interruption, users can make calls as needed. Using it, users can open groups for group-based communication, and many people can communicate simultaneously. Users can share photos, videos, and documents with the same app as they need. Security is very important when it comes to communicating through messaging apps. We have worked on the security of the users in the app. We have provided a secret chat option in this app through which users can communicate with each other while maintaining privacy. We have worked to protect users from shoulder surfing issues. Through shoulder surfing, others can see users' personal information and use the information to harm users. We came up with an idea in the field of messaging to solve this problem. While messaging using our mobile app, the user must tap on the message sent by the user to decrypt the message before viewing the message. After some time after decrypting the message, it will automatically be encrypted again. As a result of this process, users can avoid shoulder surfing. Moreover, users can report any problem to the authorities through feedback.

## TABLE OF CONTENTS

<b>CONTENTS</b>	<b>PAGE</b>
Approval	i
Declaration	ii
Acknowledgements	iii
Abstract	iv
<b>CHAPTER</b>	
<b>CHAPTER 1: INTRODUCTION</b>	<b>01-03</b>
1.1 Introduction	01
1.2 Motivation	01
1.3 Objectives	01
1.4 Expected Outcomes	01
1.5 Project Management and Finance	02
1.6 Report Layout	03
<b>CHAPTER 2: BACKGROUND</b>	<b>04-06</b>
2.1 Preliminaries/Terminologies	04
2.2 Relative Works	04
2.3 Comparative Analysis	05
2.4 Scope of the Problem	05
2.5 Challenges	06

<b>CHAPTER 3: REQUIREMENT SPECIFICATION</b>	<b>07-11</b>
3.1 Business Process Modelling	07
3.2 Requirement Collection and Analysis	08
3.3 Use Case Modelling and Description	09
3.4 Logical Data Model	10
3.5 Design Requirement	11
<b>CHAPTER 4: DESIGN SPECIFICATION</b>	<b>13-22</b>
4.1 Front-end Design	13
4.2 Back-end Design	13
4.3 Interaction Design and User Experience (UX)	14
4.4 Implementation Requirements	22
<b>CHAPTER 5: IMPLEMENTATION AND TESTING</b>	<b>23-34</b>
5.1 Implementation of Database	23
5.2 Implementation of Front-end Design	25
5.3 Testing Implementation	33
5.4 Test Results and Reports	34
<b>CHAPTER 6: IMPACT ON SOCIETY, ENVIRONMENT AND SUSTAINABILITY</b>	<b>36-37</b>
6.1 Impact on Society	36
6.2 Impact on Environment	36
6.3 Ethical Aspects	36
6.4 Sustainability Plan	37

## **CHAPTER 7: CONCLUSION AND FUTURE SCOPE**

7.1 Discussion and Conclusion	38
7.2 Limitations	39
7.3 Scope for Future Developments	39
<b>APPENDIX</b>	<b>40-44</b>
<b>REFERENCES</b>	<b>45</b>
<b>PLAGIARISM</b>	<b>46</b>



## LIST OF FIGURES

<b>FIGURS</b>	<b>PAGE NO.</b>
Figure 1.5: Gantt Chart According to the table 1.5	03
Figure 3.1: Business Process Modelling	07
Figure 3.2 Security Threat of Existing social media	08
Figure 3.3: Use Case Modelling and Description	09
Figure 3.4: Logical Data Model	10
Figure 3.5: Design Requirement	11
Figure 4.1: Splash Screen	14
Figure 4.2: Login Screen	15
Figure 4.3: Home Page & Conversation page	16
Figure 4.4: Call List	17
Figure 4.5: Create New Message	18
Figure 4.6: Create New Group	19
Figure 4.7: Profile Menu	20
Figure 4.8: Secure Chat	21
Figure 4.9: Secure Conversation Interface	22
Figure 5.1.1: User & User Authentication in Firebase	24
Figure 5.1.2: Chat & Secure Chat Database (encrypted)	25
Figure 5.2.1: Login Screen Design	26
Figure 5.2.2: Homepage Screen Design	27
Figure 5.2.3: Chat List Screen Design	28

Figure 5.2.4: Call List Screen Design	29
Figure 5.2.5: Create New Message Screen Design	30
Figure 5.2.6: Create Group Screen Design	31
Figure 5.2.7: Secure Chat Screen Design	32
Figure 5.3.1: Messaging Testing	33
Figure 5.4.1: Feedback Testing Result	34
Figure A1: Login back-end code	39
Figure A2: Homepage back-end code	40
Figure A3: Chat option back-end code	40
Figure A4: Secure Message back-end code	41
Figure A5: Connection with Database	42
Figure B1: Realtime Database of Chatting	43
Figure C1: Exit Option	44

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

“EnChat” is an Android-based full-featured chatting software, whose main goal is secure communication for users. Which users can use very easily and safely. This will add a new level of security to the users.

### 1.2 Motivation

Most internet users use chatting apps to communicate, where security is a big issue. The main objective of the App is to create a secure communication system for users by solving security problems.

### 1.3 Objectives

It is a real time messaging platform based on AES algorithm for encrypted messaging. By using this app, users can freely communicate with each other. It is safe and secure, so users don't have to worry about their data being stolen. By using this app, users can enjoy maximum security in sharing information with each other. They can use this app in an easy and reliable way. There is no possibility of users stealing or leaking any information. Even if someone tries to hack somehow, it will not be possible to decode it without the user's private key. Users can easily send messages to any part of the world without any changes by this app.

### 1.4 Expected Outcomes

- a. Security will be much stronger for those who are concerned about their personal safety in the internet world.
- b. It will be possible to secure communication so that users can freely communicate with each other with security.

## 1.5 Project Management and Finance

We can learn more about project management and finance from the table and Gantt chart below.

**TABLE 1.5: Project Management and Finance**

<b>Task Name</b>	<b>Date</b>	<b>Day to Complete</b>
1. Splash Screen	15 Dec 2021	6
2.Home	22 Dec 2021	15
3.Call List	11 Jan 2022	10
4.Create New Message	23 Jan 2022	14
5.Conversation Page	10 Feb 2022	16
6.Secure Chat	2 March 2022	63
7.Private Key Generate	10 May 2022	17
8. Implement Message Encryption	5 June 2022	15
9. Secure Chat Conversation Page	25 June 2022	9
10. Create Group	11 July 2022	7
11. Profile	20 July 2022	13
12.Feedback Page	7 Aug 2022	25
13.Firebase	2 November 2022	34

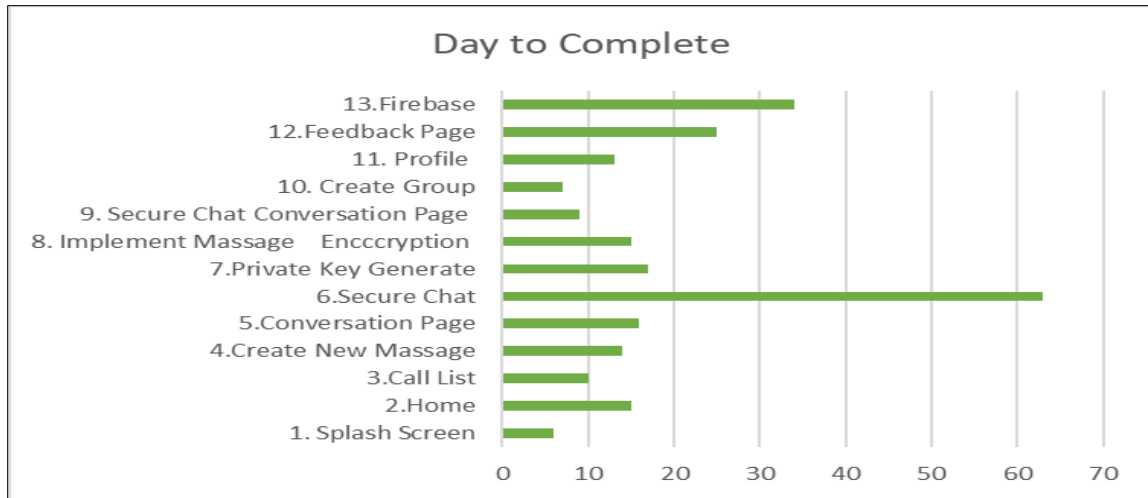


Figure 1.5: Gantt Chart According to the table 1.5

## 1.6 Report Layout

Report layout describes a summary of all the chapters. A brief summary of all chapters is given below:

- **Chapter 1:** Describes an introduction of the EnChat, Motivation, Objectives, Expected Outline, and the Report layout.
- **Chapter 2:** Describes the background, the related works, Comparative Analysis, Scope of the problems, and Challenges of the EnChat.
- **Chapter 3:** Describes the Business Process Modelling of EnChat, Requirement Collection and Analysis, Use Case Modelling and Description, Logical Data Model, and Design Requirement.
- **Chapter 4:** Describes the Front-End design of EnChat, Back-End design, 4.3 Interaction Design and User Experience (UX), and Implementation Requirements.
- **Chapter 5:** Describes the Implementation of Database Implementation of Front-end Design, Testing Implementation, Test Results and Reports of EnChat.
- **Chapter 6:** Describes the Sustainability of our app, Impact on Society, Impact on Environment, Ethical Aspects, and Sustainability Plan.
- **Chapter 7:** Describes the Conclusion, Limitations and Future Scope of our app.

## **CHAPTER 2**

### **BACKGROUNDS**

#### **2.1 Preliminaries/Terminologies**

The inclusion of technologies from today. Every day, it gets better. As information sharing has improved in the current era, the risk of data breach is increasing day by day. Currently, all types of information, data, government documents are exchanged through the internet. Due to the development of technology, communication system is also advancing day by day. In today's world people prefer internet as a medium of easy communication. As a result, they exchange all their information differently through the app. People always look for a secure platform to share this information. Nowadays, hackers use various platforms to breach people's information. Cyber security experts are constantly inventing new security methods to prevent them.

With this mind, we are thinking of creating a mobile application that will help for people all over the world, especially security concern people, government, Army, or National security agencies etc. The most important thing for a messaging app is not viewing one person message to another. In this app user get a private key and a public key. Here, Phone number is public key for every user, but the private key is generated by the app or user can enter it manually under secure chat option. After getting messages from a contact, the messages always in encrypt form. After viewing messages, it will be encrypted form again for the security purpose. If anyone does not want this system, he/she also can change it to regular chat.

#### **2.2 Related Works**

Different types of chatting apps related to our mobile applications are used worldwide. Among the various apps used worldwide, the most notable are WhatsApp, Messenger, Telegram, Instagram, etc. Because it enables users to communicate between users in real time through text, voice, and file sharing, the app is popular with users of mobile and traditional computing devices (ex: personal computers and laptops).

### **2.3 Comparative Analysis**

A popular and frequently used app named 'Messenger' is related to our app. Our software is linked to Messenger's main security feature named 'Go to secret conversation'. Facebook Messenger allows you to exchange information, images, videos, documents, stories, and days. Messenger takes up a lot of space on the smartphone because it has so many functions. Messenger occasionally crashes on phones with limited RAM. Messenger's Go to Secret conversion offers a message disappearing option. As a result, useful messages are frequently erased automatically. There is no way to access the message afterwards. They also feature end-to-end encryption, however the message on the user interface or conversation area is not encrypted. As a result, the user is vulnerable to a shoulder surfing attack.

In our application, however, information, photos, videos, and documents can be transmitted. Our app takes up very little space on the phone. As a result, our app will work on any smartphone. We also offer two alternatives for communication: regular chat and secure chat, similar to messenger. Our app's secure chat feature enables encrypted message exchange and reverts to encrypted mode after viewing the message. As a result, communication is secure, and no solder surfing attacks are possible. This communication can later be accessed using a private key which produced automatically by the app or manually by the user.

### **2.4 Scope of the Problems**

In today's era, as cybercrime is increasing, people's fear of data theft is also increasing. It is often heard that various private information and conversations of the people are sold on dark web. These increases the risk of human life as well as undermines the privacy of people.

### **2.5 Challenges**

Challenges include anything that potentially ominously affect our system from the outside, such as deliver chain issues, changes in market demands, or a shortage of recruitment. It's important to anticipate difficulties and make progress toward them before we become victims of them and experience a stop in our growth.

Nowadays many tech companies have released their own messaging applications in the market. They are monopolizing their influence on this side. As the number of their users is increasing day by day, the surveillance of hackers towards these tech giants is also increasing. For example, 50 million Facebook accounts were hacked in 2018. It was one of the company's biggest data breaches. In this way, Google and Twitter are constantly becoming the target of hackers.

Consequently, facing out against companies like Google, Facebook, WhatsApp and Telegram is a huge task for us. Even though our whole structure is based on systems that allow for the sale or sharing of applications, Copyright declaration issues are also present.

Another challenge we have is that we must run our application at low internet speed. Because people in remote areas of our country do not get proper internet service. 3G internet service is not fully available to them yet. We may also take measures to ensure that no one can use our application for unethical activities.



# CHAPTER 3

## REQUIREMENT SPECIFICATION

### 3.1 Business Process Modelling

Figure 3.1 displays the fundamental business process model. This is a flowchart model that displays the processes that can be performed to represent the business processing model from beginning to end. Through visual representation, we may understand the specific process. If we understand the entire process through code, it will be quite difficult; nevertheless, anyone can easily understand the concept of the process using flowchart. In the case of our project, the user will first go to the home screen when they open the application.

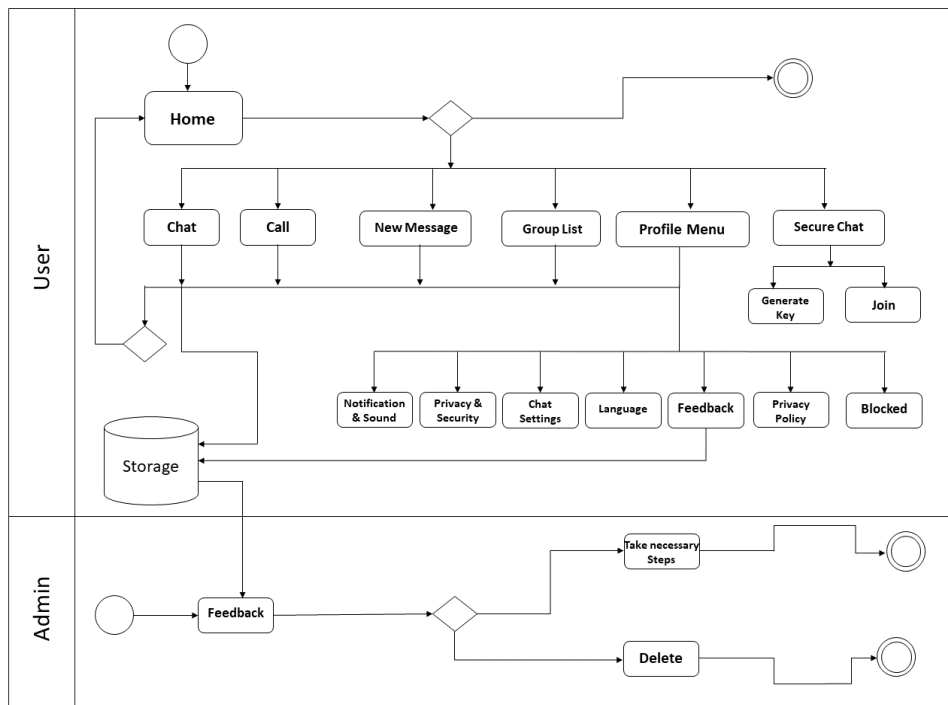


Figure 3.1: Business Process Modelling

The user can access the menu bar, search icon, call, new message, group, and profile options from the home screen in addition to viewing the user's entire conversation history.

The user has a variety of options on the home screen, allowing them to navigate between pages with ease. The user will be taken to the next activity if they select the call list, then user go to create new message page by clicking call next icon which is mentioned by ‘Plus’ sign. Every choice from the bottom will work in this way. On the conversation list, the Secure chat is in the upper right corner. Notifications & Sound, Privacy & Security, Chat settings, Feedback, Privacy Policy and Blocked options can be accessed by clicking the Profile icon. Users can immediately contact the administrator through the feedback option to express their thoughts and make suggestions. This will enable us to improve the app's usability in the future.

### 3.2 Requirement Collection and Analysis

There are several talking applications accessible via which we may communicate messages, videos, information, and so on. However, cybercrime is becoming more widespread by the day. Everyone who uses the internet to communicate information now need a secure real-time messaging program. As a result, several well-known firms, such as Meta (Messenger & WhatsApp), Telegram, provide various secure features, but they have certain flaws. Cyber criminals somehow found a technique to steal someone's information. We considered every aspect of a messaging app before developing this app.

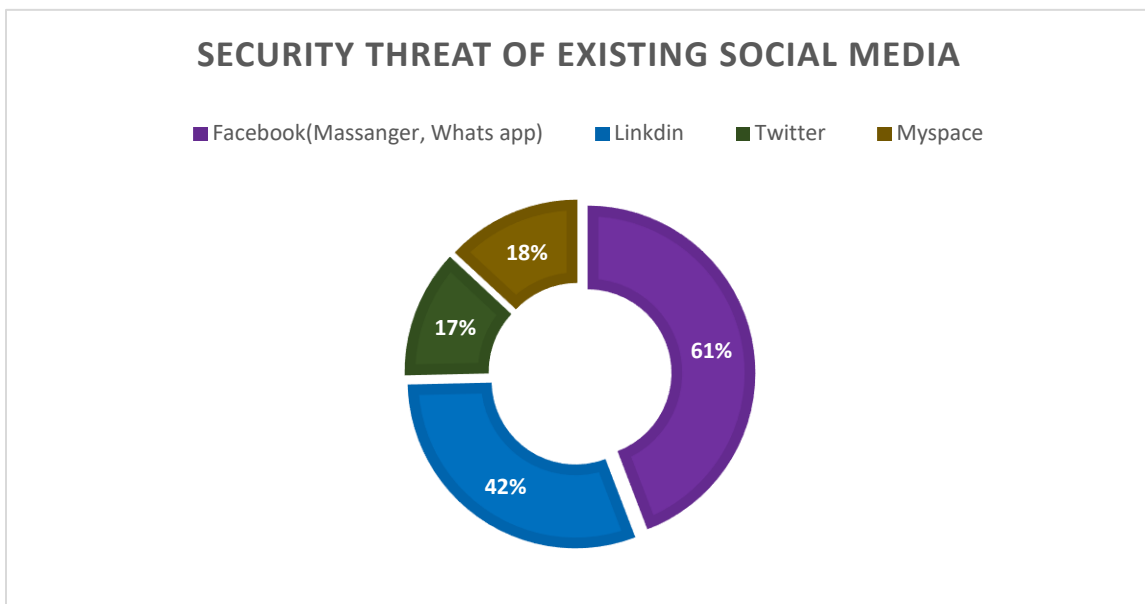


Figure 3.2: Security Threat of Existing social media

We implement security while sending a message to be seen, and we additionally encrypt the message after seeing it. A private key is generated automatically by the app or manually by the user and shared with another user in order to begin a conversation.

We utilized an encrypted messaging mechanism here. We will also provide basic chat options, such as text message, audio calls, group creation, and so on, similar to a standard chatting program.

### 3.3. Use Case Modelling and Description

We have built our app by analyzing the data from the internet on the problems faced by security concern people. If we look at our use case diagram, then here are two actors. One actor is user, and another actor is admin. Here, the primary actor is the user. Here, user can view chat, calls, Create new message, group list, and profile option. User can also view Secure chat option on the top right corner. By clicking on Lock icon which is secure chat icon, a new interface up and show generate or enter private key and join button. By clicking profile option, Notifications & Sound, Privacy & Security, Chat settings, Feedback, Privacy Policy and Blocked option can be accessed. User can give feedback about problems and opinion through text message.



Figure 3.3: Use Case Modelling and Description

Admin can view feedbacks and take necessary steps or delete feedbacks.

**Use Case:** Home

**Actor:** User, Admin

**Primary Actor:** User

**Description:** user can view chat, calls, group list, profile, and new message and secure chat option. User can also view menu option on the top left corner.

**Pre-condition:** Users must have the app installed on their Android phones.

**Post-condition:** Before using the app and send message, the user needs to connect to the internet.

### 3.4 Logical Data Model

A diagram of entity-relationships is shown, providing a logical representation of the data model. It outlines the system's design, the nature of each entity, how they interact, and how logically dependent they are on one another. It will assist us in determining the components we require for our project and the relationships we can create between those components.

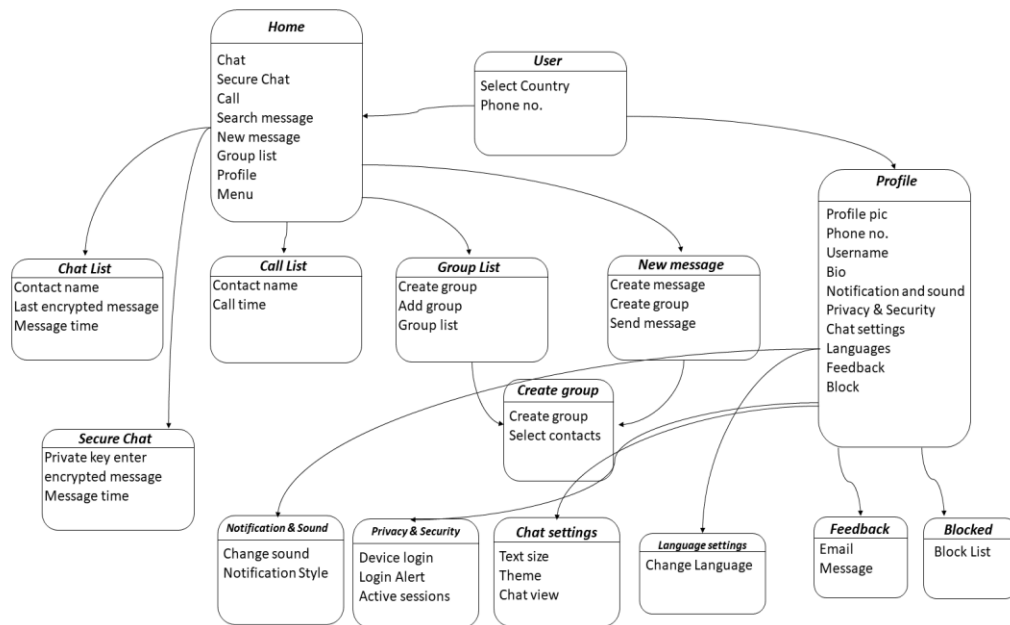


Figure 3.4: Logical Data Model

Multiple tables, including Home, Chat, Call, New Message, Group List, Profile, and Secure chat, are present in the database. From the home page, the user can access all entities. The entity labeled home includes a Secure chat entity in the upper right corner, and the secure chat has a new interface where user enter or generate private key manually. After enter key two user join e secret conversation. Email and message attributes are present in the feedback entity. Only the admin can view the feedback messages, act accordingly, and delete the feedback from the database.

### 3.5 Design Requirements

Users can easily send messages, invite others, keep their data protected, and feel comfortable using it since we gather and analyze the references we need. We developed it with that in mind, regardless of where it is.

Home Page: We utilized a chat list view here, as well as a nav bar at the bottom of the homepage. We retain five options in the navbar. Like chat list, call list, create new message, create new group, and last is profile menu. In the upper right corner, we placed a lock icon which indicates secure chat option.

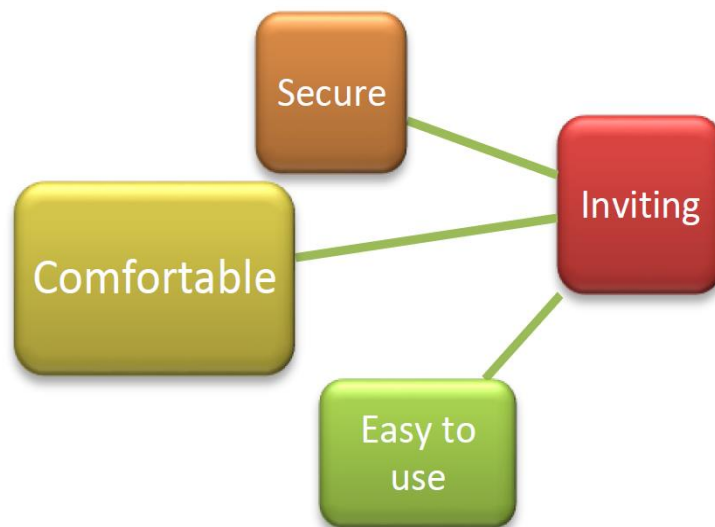


Figure 3.5: Design Requirement

Call list: Here, the user may find another person from their contact list or search for them by name. These contacts are shown as a list to the user. Users may summon their preferred user with a single press.

Create a new Message: By tapping this button, users may select one individual or search for them by name and select from their contacts. After selecting, they proceed to our main conversation page and send a message.

Create New Group: Here, the user may create a group with their friends and family from their contacts.

Profile Menu: From here, users may change their name, username, bio, and access many other features such as Settings, Notification & Sound, Chat settings, Language, and Privacy policies, and so on.

## **CHAPTER 4**

### **DESIGN SPECIFICATION**

#### **4.1 Front-end Design**

Android Studio, Visual Studio, Droid Script, and more IDEs are available for building Android apps. Visual Studio Code was used in our project. Because Visual Studio Code is the best IDE for developing Android apps. Using VS Code as an Android programming environment has several advantages. Here, we may make use of a variety of extensions that are useful for writing and organizing code. With syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and other features, VS Code allows us to be productive quickly. This software application allows for front-give-up and back-give-up improvement in a simple environment.

#### **4.2 Back-end Design**

For developing the back-end part of our project we have used Dart as a programming language and Flutter as a framework. Flutter is a mobile app development platform created by Google. It enables developers to build online, desktop, and cross-platform applications for Android and iOS devices. Flutter makes use of the Dart reactive programming language, which makes development faster and easier than traditional approaches.

Actually, the primary goal of Android expansion software became to create a platform software environment that could operate on any device. Flutter is a mobile app SDK that enables the development of high-quality native iOS and Android apps. That is why we picked Dart as the programming language for our app.

#### **4.3 Interaction Design and User Experience (UX)**

We have designed the interface of our project using Flutter. Flutter is a UI software development kit developed by Google and is open source. It is utilized to create cross-platform apps. We used Dart programming language to perform our work in the flutter. Dart is an object-oriented language that uses a syntax similar to C. It supports programming constructs like classes and interfaces.

Below are some screenshots for ease of understanding.

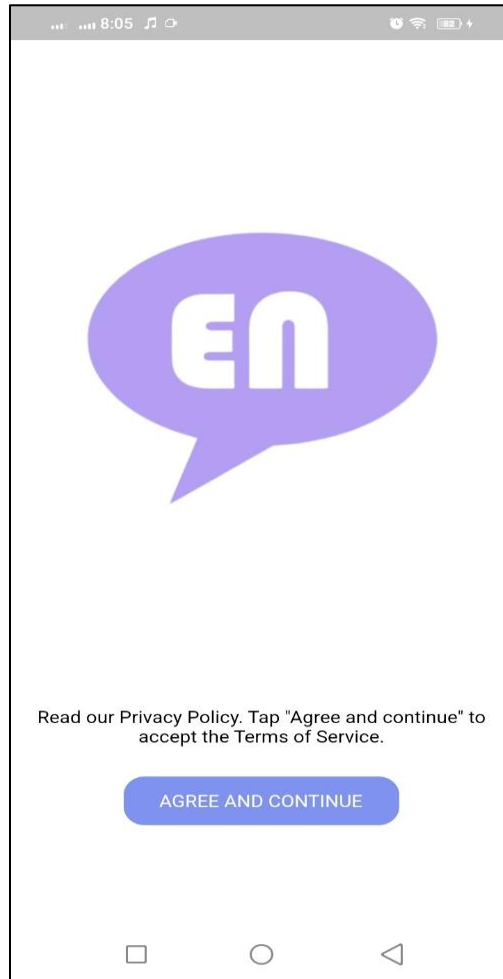


Figure 4.1: Splash Screen

After Privacy policy and terms and conditions are very important things. First of all, users will know about our terms and privacy policy. Users need to agree with our terms and privacy policy to create an account. After agreeing with the terms and conditions and privacy policy, users need to send their number using the app to create their account.



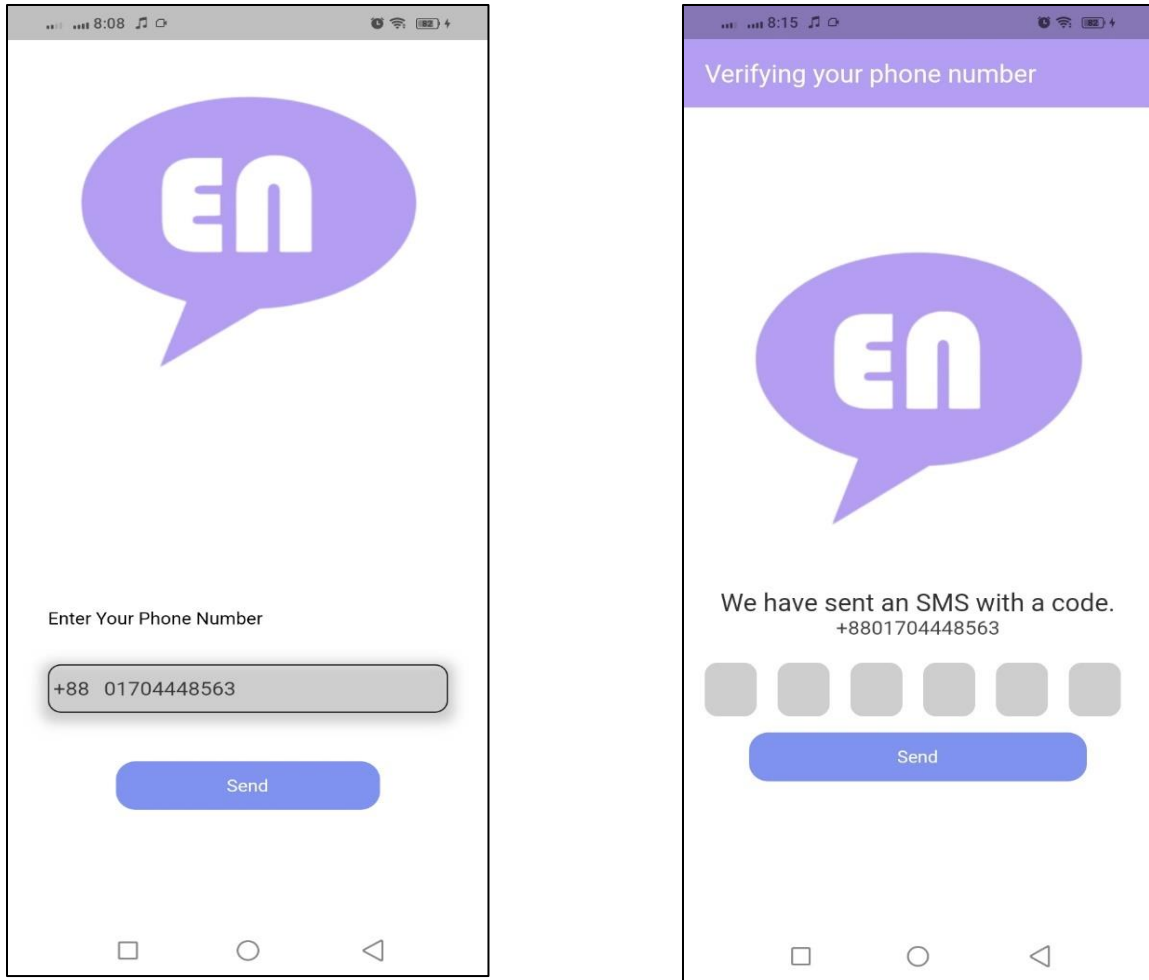


Figure 4.2: Login Screen

After users send their number, an OTP code will be sent to their personal mobile phone through a message. Users can verify their account creation through the OTP code sent to them. After verifying the account using the OTP code, the user can enter the home page.

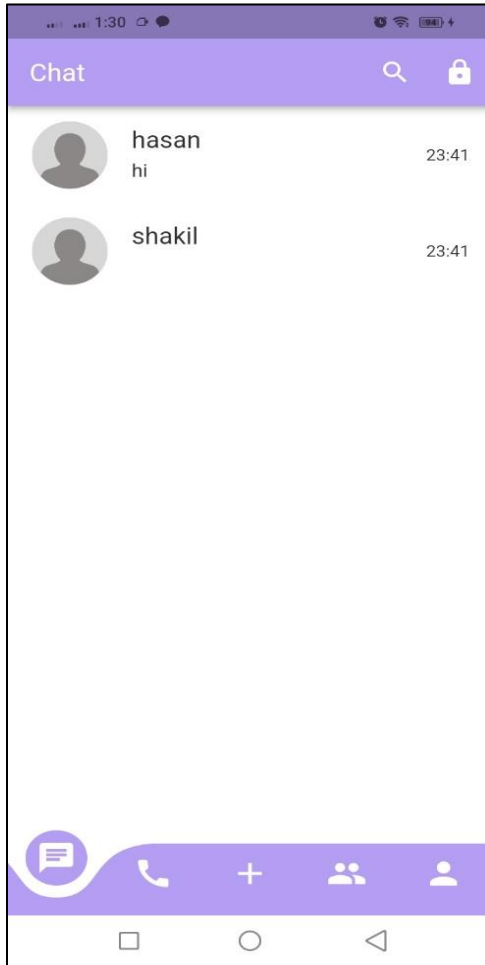


Figure 4.3: Home Page & Conversation page

By clicking Call icon from homepage user will be able to see call list and a search button on the top of call list page. Here user can call easily by find or select contact. User also call another user from their chat conversation page.

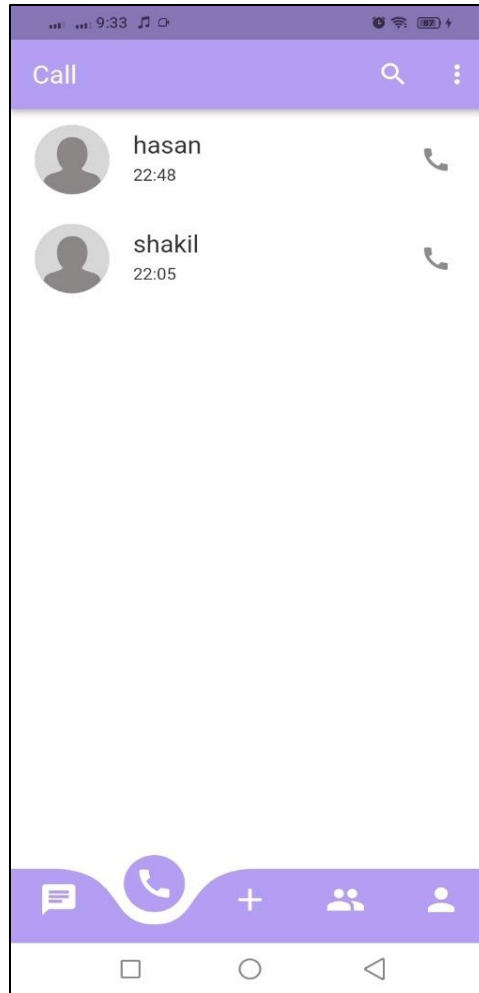


Figure 4.4: Call List

From call list user can go Create new message icon which is indicates by 'Plus' sign. Here user can select a contact to create a new message. Then it takes the user to conversation page.

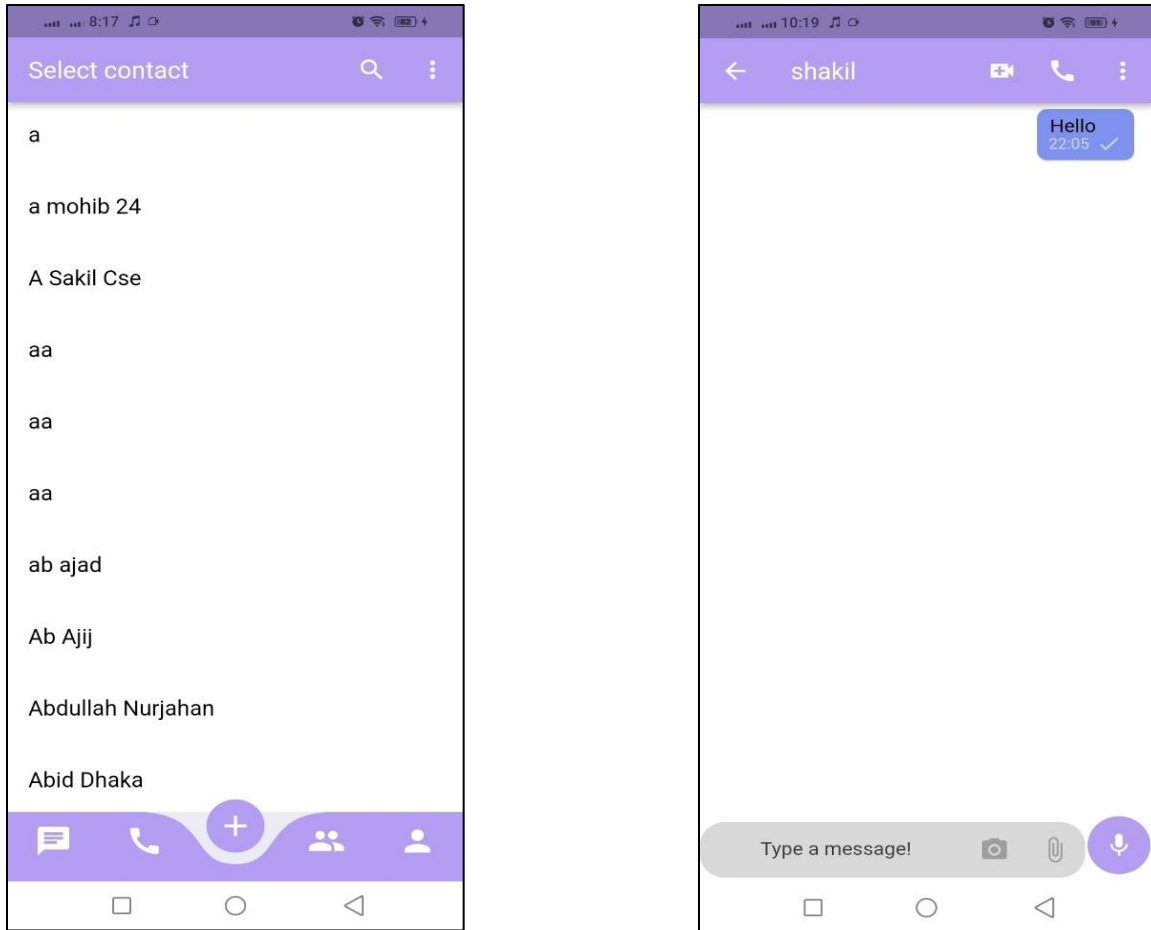


Figure 4.5: Create new message

Users may choose the group symbol to create a new group with relatives or friends from their contact list. They also have the option of naming their group and selecting all desired connections.

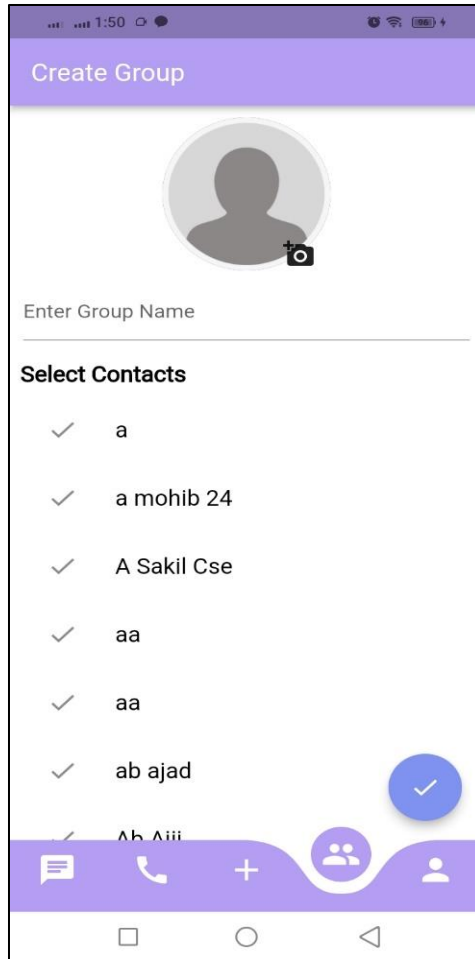


Figure 4.6: Create new group

The profile menu icon allows the user to edit or enter settings. After clicking the profile menu button, the user is shown with such options like username and bio. They also can change the information on this page. They may also change the notification and sound settings, the privacy and security settings, the chat settings, the language setting, the privacy policy, and the block settings. They can also send feedback to us using Feedback option from this page.

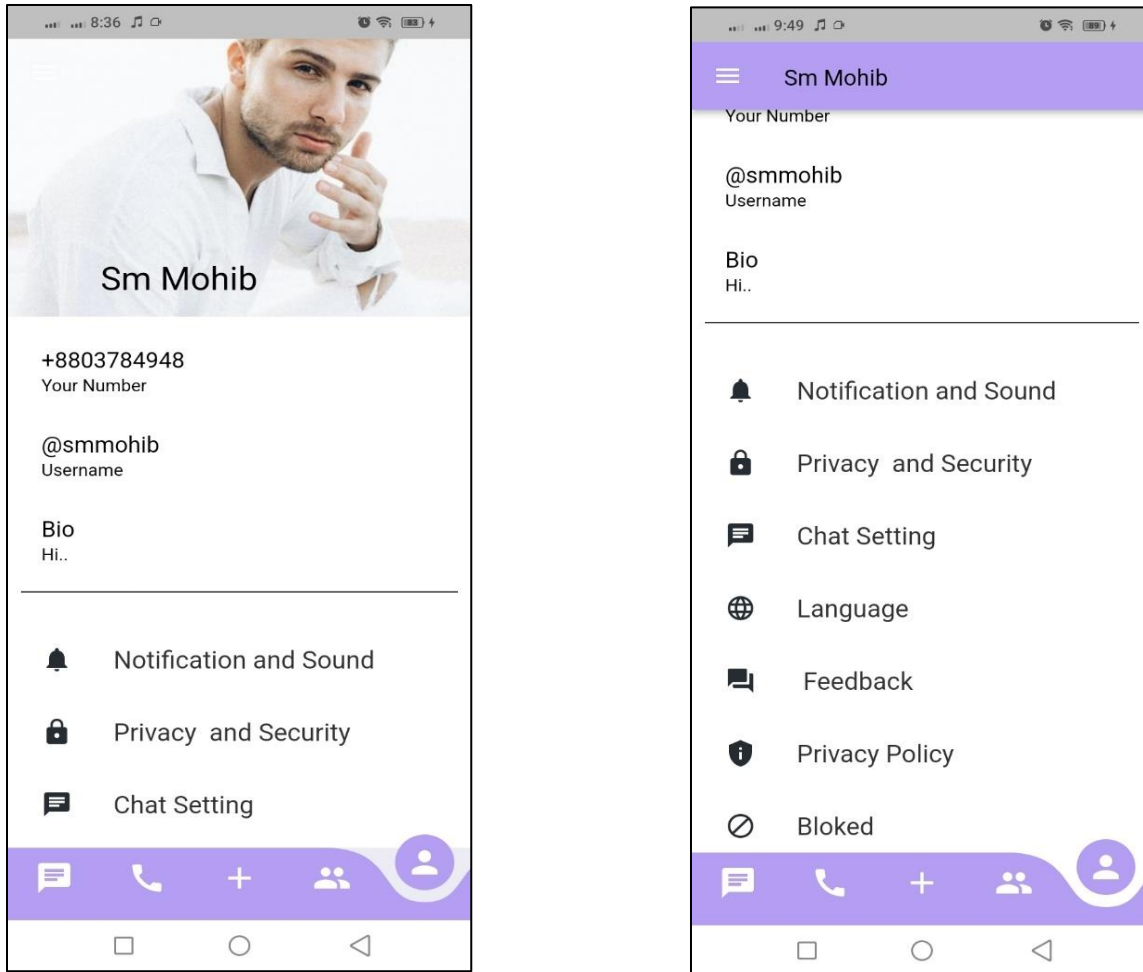


Figure 4.7: Profile Menu

When a user wants to access the secure chat option, they must tap the secure chat button in the right upper corner of the homepage to access the secure chat interface. In this interface, the user can manually enter or generate a private key. After clicking the join button, the user is sent to a conversation page where they can securely message others.

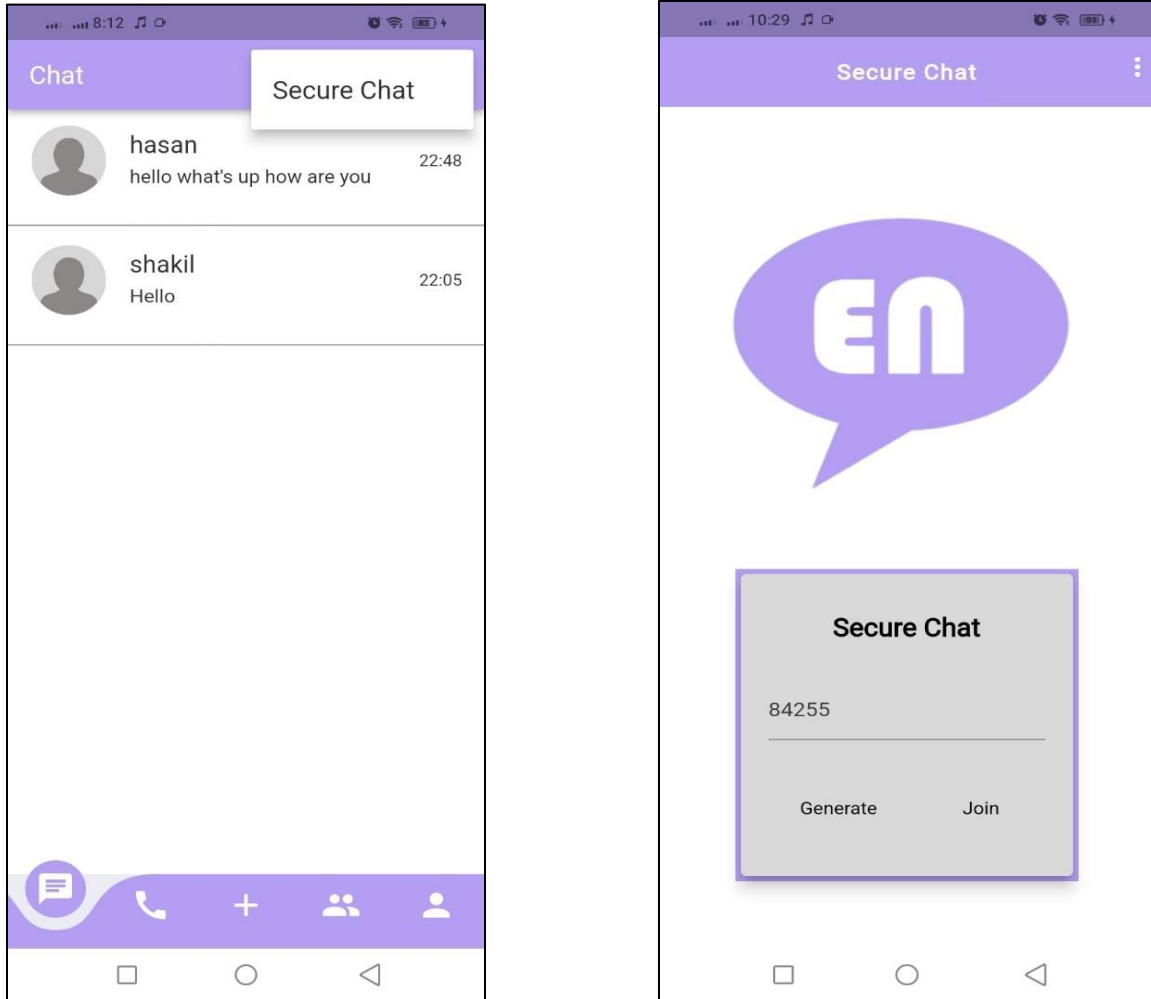


Figure 4.8: Secure chat

After entering the secure conversation page, the user can send a message in encrypted form. When the receiver taps on the message, it is displayed as normal text. It will automatically revert to encrypted form after two seconds.

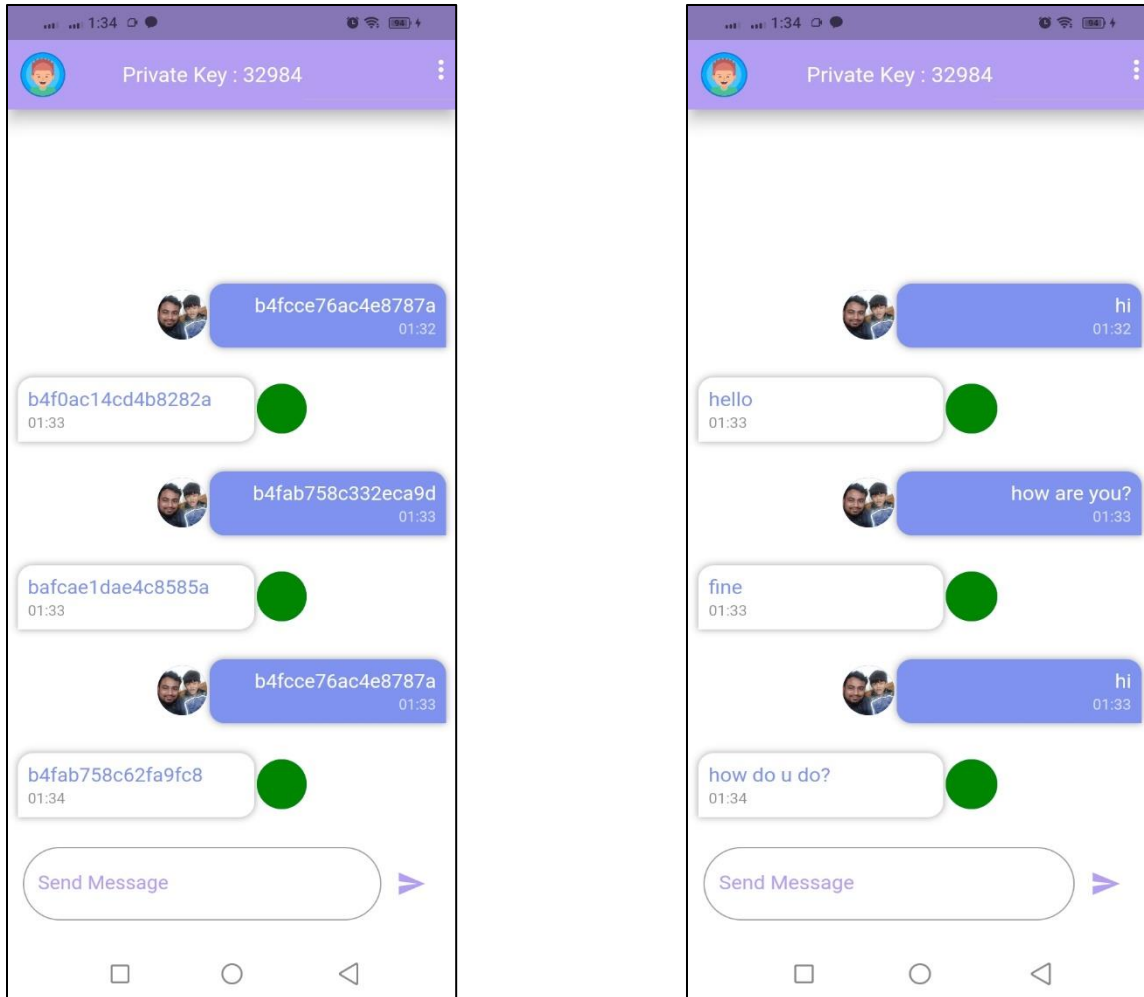


Figure 4.9: Secure conversation interface

#### 4.4 Implementation Requirements

Several steps were taken to complete this project. It is essential to outline what was necessary to carry out this project. This project requires the Flutter and Dart programming languages. We worked with the flutter and Dart languages in Visual Studio Code. The project made use of certain real-time databases. When dealing with a real-time database, we must be familiar with Firebase. We also store our app data and real-time communications using firebase. We need to understand real-time communications as well as encrypted messaging systems. We also need a thorough understanding of the encryption process and how to apply it.



## CHAPTER 5

### IMPLEMENTATION AND TESTING

#### 5.1 Implementation of Database

A database is a structured collection of information or records that is often kept electronically in a computer system. Database security and record safety are strictly enforced. A secure database is the core of an automated system. In our project, we use Firebase database below room database to keep entries.

Firebase is a Back-end software program. It is a real time database. It is a NoSQL database hosted in the cloud that allows us to store and sync data between our users in real time. It is a Google-developed real time database for interactive internet applications, Android, and iOS apps. As a result, we can state that it is a Google back-end tool that provides first-rate services such as app crash resolution, analytics recording, and monitoring. A database in Firebase is kept in a single file, which separates it from other database engines.

The primary reason for using Firebase is that we can utilize various Google or third-party plug-ins or packages such as: `firebase_auth`, `cloud_firestore`, `permission_handler`, `network_image`, `Cached_video_player` (for media play in app). These plug-ins and packages are available for free here. It also expands the functionality of our app. Because our database is a real-time database platform, users may quickly contact other users who are using our app through their contacts. The database table is stored as a hierarchical approach right here, which makes it very simple to gain right of entry to the primary key by contacting the basic data.

In firebase, our secure chat option is very adequately protected. When a user sends a secure message to another, it is encrypted and remains encrypted in our database. This message is only accessible if the private key is accurate. When a user detects or encounters an issue, he can notify us using the feedback option. This option can be found in the profile menu. They must enter their email address to send feedback messages. Following that, our

administrative staff will investigate and resolve the issue. We can simply provide an explanation for the information in the following database snapshot.

The screenshot shows the Google Cloud console interface for a Firebase database. The breadcrumb path is 'users > AYT3kJhQJlgzv...'. The left sidebar shows the 'myrsa-app' project with collections 'chat', 'users', and 'users2'. The 'users' collection is selected, showing a list of documents with their keys. The document key 'AYT3kJhQJlgzvQSow2z3w77P4m32' is expanded to show its fields:

- groupId
- isOnline: true
- name: "habib"
- phoneNumber: "+8801234567890"
- profilePic: "https://png.pngitem.com/pimgs/s/649-6490124\_katie-notopoulos-katienotopoulos-i-write-about-tech-round.png"
- uid: "AYT3kJhQJlgzvQSow2z3w77P4m32"

The screenshot shows the 'Authentication' console in the Google Cloud interface. The 'Users' tab is active, displaying a table of registered users. The table has the following columns: Identifier, Providers, Created, Signed in, and User UID.

Identifier	Providers	Created ↓	Signed in	User UID
lili@gmail.com	✉	16 Dec 2022	16 Dec 2022	Zbd7CPduQJan9pYC5i80la2F0Bm2
lallga@gmail.com	✉	16 Dec 2022	16 Dec 2022	2MetpEFVDXT9K9N3qjCCNT4Gsf22
kabil@gmail.com	✉	16 Dec 2022	16 Dec 2022	S8seMLOi9SWHEmaDcpH6ulxQV...
jakir@gmail.com	✉	16 Dec 2022	16 Dec 2022	YK7GfZ112TSzs4oYwlfnsNDjLWD3
bilal@gmail.com	✉	16 Dec 2022	16 Dec 2022	RrP2VXkY07c4G0eN0A6Mo1FuEC...
kible@gmail.com	✉	16 Dec 2022	16 Dec 2022	ziMp5uapP8QBgdK8nPXA3go51p...
mohibs@gmail.com	✉	15 Dec 2022	15 Dec 2022	eThr5aq2RGQycHS4TnQ6CjkvfeU2

Figure 5.1.1: User & User authentication in Firebase

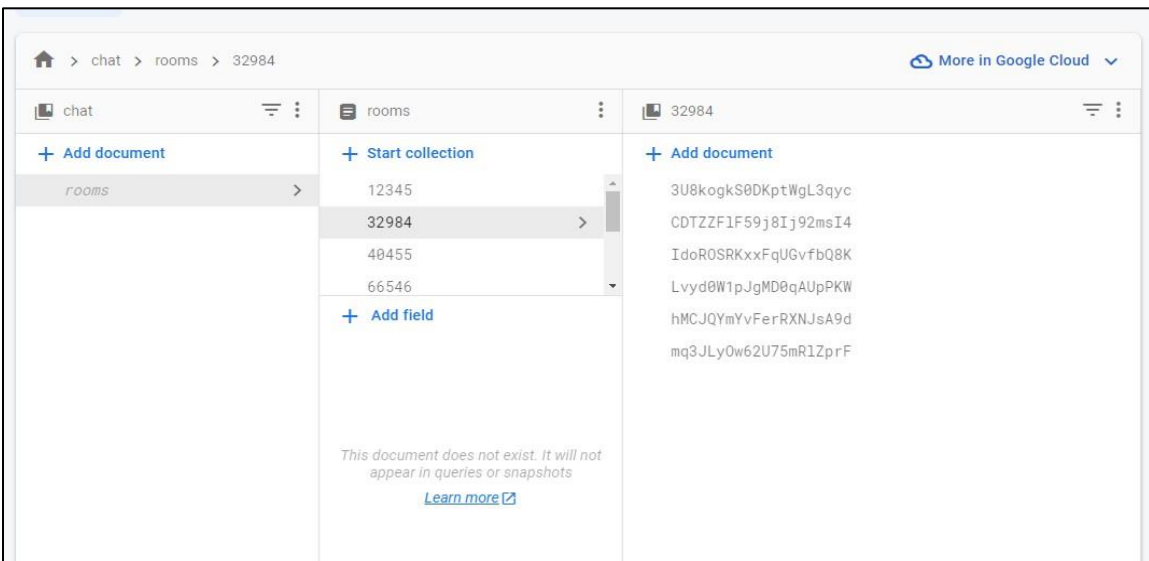
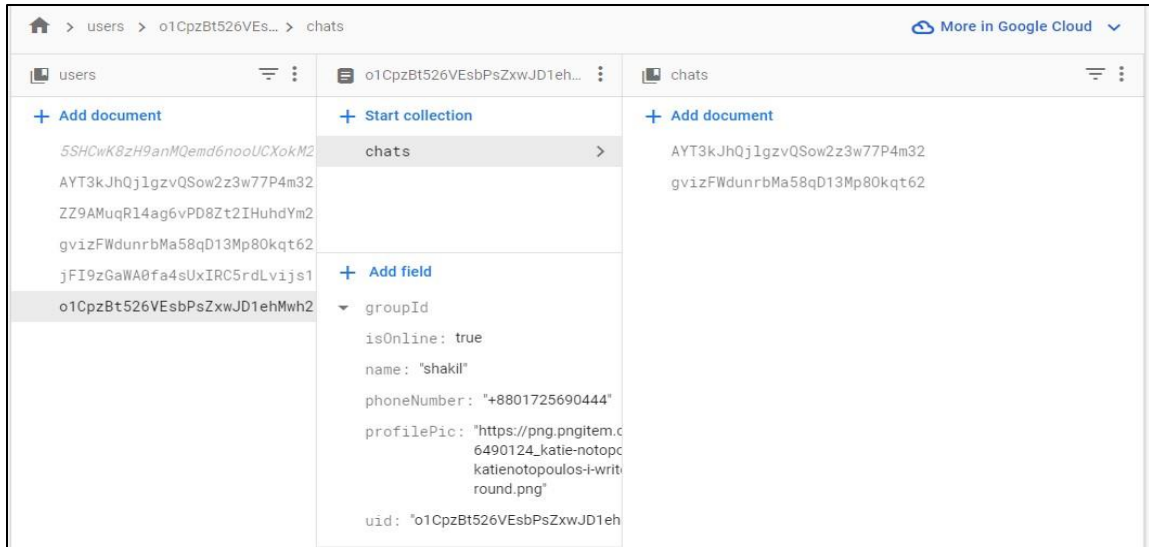


Figure 5.1.2: Chat & Secure Chat Database (encrypted)

## 5.2 Implementation of Front-end Design

The term "front-end design" refers to the graphical user interface. The front-end of a mobile app is everything the user sees, including the app's design. Front-end design interacts directly with the user. This is a synonym for "User Experience" or "UX".

In our project, we use flutter as a framework. We write flutter programming to provide a decent and simple "user interface" to our users. Our app design back-end code is mentioned below to help you understand it better.

## Flutter Code:

### Login Screen

After accepting the terms and conditions, the app will redirect the user to the phone number and OTP page.

```
59     body: SingleChildScrollView(  
60       child: Padding(  
61         padding: const EdgeInsets.all(18.0),  
62         child: Column(  
63           crossAxisAlignment: CrossAxisAlignment.center,  
64           children: [  
65             const Text('We need to verify your phone number.'),  
66             const SizedBox(height: 10),  
67             TextButton(  
68               onPressed: pickCountry,  
69               child: const Text('Pick Country'),  
70             ), // TextButton  
71             const SizedBox(height: 5),  
72             Row(  
73               children: [  
74                 if (country != null) Text('+${country!.phoneCode}'),  
75                 const SizedBox(width: 10),  
76                 SizedBox(  
77                   width: size.width * 0.7,  
78                   child: TextField(  
79                     style: text16(),  
80                     controller: phoneController,  
81                     decoration: const InputDecoration(  
82                       hintText: 'phone number',  
83                     ), // InputDecoration  
84                   ), // TextField  
85                 ), // SizedBox  
86             ],
```

```
28     appBar: AppBar(  
29       title: const Text('Verifying your number'),  
30       elevation: 0,  
31       backgroundColor: backgroundColor,  
32     ), // AppBar  
33     body: Center(  
34       child: Column(  
35         children: [  
36           const SizedBox(height: 20),  
37           const Text('We have sent an SMS with a code.'),  
38           SizedBox(  
39             width: size.width * 0.5,  
40             child: TextField(  
41               style: text16(),  
42               textAlign: TextAlign.center,  
43               decoration: const InputDecoration(  
44                 hintText: '- - - - -',  
45                 hintStyle: TextStyle(  
46                   fontSize: 30,  
47                 ), // TextStyle  
48               ), // InputDecoration  
49               keyboardType: TextInputType.number,  
50               onChanged: (val) {  
51                 if (val.length == 6) {  
52                   verifyOTP(ref, context, val.trim());  
53                 }  
54               },  
55             ), // TextField  
56           ), // SizedBox
```

Figure 5.2.1: Login Screen Design

## Homepage

Our homepage has seven options, including a Chat list in a list view and others indicated by various icons.

```
24 class _MobileLayoutScreenState extends ConsumerState<MobileLayoutScreen> {
25   int _pageIndex = 0;
26   GlobalKey _bottomNavigationKey = GlobalKey();
27
28   List pages = [
29     MyRoute(
30       iconData: Icons.chat,
31       page: ContactsList(),
32     ), // MyRoute
33     MyRoute(
34       iconData: Icons.call,
35       page: CallList(),
36     ), // MyRoute
37     MyRoute(
38       iconData: Icons.add,
39       page: SelectContactsScreen(),
40     ), // MyRoute
41     MyRoute(
42       iconData: Icons.people,
43       page: CreateGroupScreen(),
44     ), // MyRoute
45     MyRoute(
46       iconData: Icons.person,
47       page: ProfilePage(),
48     ), // MyRoute
49   ];
50
```

```
55     bottomNavigationBar: CurvedNavigationBar(
56       key: _bottomNavigationKey,
57       index: 0,
58       height: 60.0,
59       items: pages
60         .map(
61           (p) => Icon(
62             p.iconData,
63             size: 30,
64             color: Colors.white,
65           ), // Icon
66         )
67         .toList(),
68       // color: Color(0xff4367b1),
69       color: appBarColor,
70       buttonBackgroundColor: appBarColor,
71       //buttonBackgroundColor: Color(0xff4367b1),
72       backgroundColor: whiteColor,
73       animationCurve: Curves.easeInOut,
74       animationDuration: Duration(milliseconds: 500),
75       onTap: (index) {
76         setState(
77           () {
78             _pageIndex = index;
79           },
80         );
81       },
82     ), // CurvedNavigationBar
83     backgroundColor: Colors.white,
84     body: pages[_pageIndex].page,
85   ); // Scaffold
```

Figure 5.2.2: Homepage Screen Design

## Chat List Page

Here user can view all conversation list and see who is online through the green light.

```
118     return Scaffold(  
119       backgroundColor: whiteColor,  
120       appBar: AppBar(  
121         backgroundColor: appBarColor,  
122         title: StreamBuilder<UserModel>(  
123           stream: ref.read(authControllerProvider).userDataById(uid),  
124           builder: (context, snapshot) {  
125             if (snapshot.connectionState == ConnectionState.waiting) {  
126               return const Loader();  
127             }  
128             return Column(  
129               crossAxisAlignment: CrossAxisAlignment.start,  
130               children: [  
131                 Text(name),  
132                 Text(  
133                   snapshot.data!.isOnline ? 'Online' : 'Offline',  
134                   style: const TextStyle(  
135                     fontSize: 13,  
136                     fontWeight: FontWeight.normal,  
137                   ), // TextStyle  
138                 ), // Text  
139               ],  
140             ); // Column  
141           }), // StreamBuilder  
142       centerTitle: false,  
143       actions: [  
144         IconButton(  
145           onPressed: (() {}),  
146           icon: const Icon(Icons.video_call),  
147         ), // IconButton
```

Figure 5.2.3: Chat List Screen Design



## Call List

Here user can see call list with time.

```
69     body: Padding(  
70       padding: const EdgeInsets.only(top: 5),  
71       child: SingleChildScrollView(  
72         child: Column(  
73           children: [  
74             StreamBuilder<List<Group>>(  
75               stream: ref.watch(chatControllerProvider).chatGroups(),  
76               builder: (context, snapshot) {  
77                 if (snapshot.connectionState == ConnectionState.waiting) {  
78                   return const Loader();  
79                 }  
80  
81                 return ListView.builder(  
82                   shrinkWrap: true,  
83                   itemCount: snapshot.data!.length,  
84                   itemBuilder: (context, index) {  
85                     var groupData = snapshot.data![index];  
86  
87                     return Column(  
88                       children: [  
89                         InkWell(  
90                           onTap: () {  
91                             Navigator.pushNamed(  
92                               context,  
93                               MobileChatScreen.routeName,  
94                               arguments: {  
95                                 'name': groupData.name,  
96                                 'uid': groupData.groupId,  
97                                 'isGroupChat': true,  
98                                 'profilePic': groupData.groupPic,  
99                               },  
100                          ),  
101                        ],  
102                      ),  
103                    ],  
104                  ),  
105                ),  
106              ],  
107            ),  
108          ),  
109        ),  
110      ),  
111    ),
```

Figure 5.2.4: Call List Screen Design

## Create New Message

From this screen, the user can select a preferred contact and compose a new message to begin a conversation.

```
42     body: ref.watch(getContactsProvider).when(  
43       data: (contactList) => ListView.builder(  
44         itemCount: contactList.length,  
45         itemBuilder: (context, index) {  
46           final contact = contactList[index];  
47           return InkWell(  
48             onTap: () => selectContact(ref, contact, context),  
49             child: Padding(  
50               padding: const EdgeInsets.only(bottom: 6.0),  
51               child: ListTile(  
52                 title: Text(  
53                   contact.displayName,  
54                   style:  
55                     const TextStyle(fontSize: 18, color: textColor),  
56                 ), // Text  
57                 leading: contact.photo == null  
58                   ? null  
59                   : CircleAvatar(  
60                     backgroundImage: MemoryImage(contact.photo!),  
61                     radius: 30,  
62                   ), // CircleAvatar  
63               ), // ListTile  
64             ), // Padding  
65           ); // InkWell  
66         }), // ListView.builder  
67       error: (err, trace) => ErrorScreen(error: err.toString()),  
68       loading: () => const Loader(),  
69     ),  
70   ); // Scaffold  
71 }
```

Figure 5.2.5: Create New Message Screen Design



## Create a Group

```
45 @override
46 Widget build(BuildContext context) {
47   return Scaffold(
48     appBar: AppBar(
49       title: const Text('Create Group'),
50     ), // AppBar
51     body: Center(
52       child: Column(
53         children: [
54           const SizedBox(height: 10),
55           Stack(
56             children: [
57               image == null
58                 ? const CircleAvatar(
59                   backgroundImage: NetworkImage(
60                     'https://png.pngitem.com/pimgs/s/649-6490124_katie-not
61                   ), // NetworkImage
62                   radius: 64,
63                 ) // CircleAvatar
64               : CircleAvatar(
65                   backgroundImage: FileImage(
66                     image!,
67                   ), // FileImage
68                   radius: 64,
69                 ), // CircleAvatar
70             Positioned(
71               bottom: -10,
72               left: 80,
73               child: IconButton(
74                 onPressed: () {},
75                 // onPressed: selectImage,
```

Figure 5.2.6: Create Group Screen Design

## Secure Chat

A safe and secure conversation using a private key can be started here. Only the right private key is permitted to see messages. When you tap the secure icon, a private key enter page appears.

```
15 ~ class JoinRoom extends StatefulWidget {
16   @override
17   _JoinRoomState createState() => _JoinRoomState();
18 }
19
20 class _JoinRoomState extends State<JoinRoom> {
21   final _formKey = GlobalKey<FormState>();
22
23   var randomId = TextEditingController();
24   var roomId;
25   late int randomRoomID;
26   Color background = Color(0xff125589);
27   Color button = Color(0xffe0e0e0);
28
29   void generateRoomId() {
30     // ignore: unnecessary_new
31     Random random = new Random();
32     setState(() {
33       randomId.text = random.nextInt(99999).toString();
34     });
35   }
36
37   void logout() async {
38     SharedPreferences pref = await SharedPreferences.getInstance();
39     pref.remove('token');
40     FirebaseAuth.instance.signOut();
41     print('logout');
42     Navigator.push(
43       context,
44       MaterialPageRoute(builder: (context) => AuthScreen()),
45     );
46   }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Figure 5.2.7: Secure Chat Screen Design

### 5.3 Testing Implementation

Software testing is an essential procedure that evaluates a specific software program to find bugs or faults and ensure that it satisfies the client's expectations. Furthermore, the software testing process assures that a produced program is free of faults and mistakes, as well as that quality requirements are met. It evaluates the usability, reusability, performance, reliability, scalability, and portability of a software program.

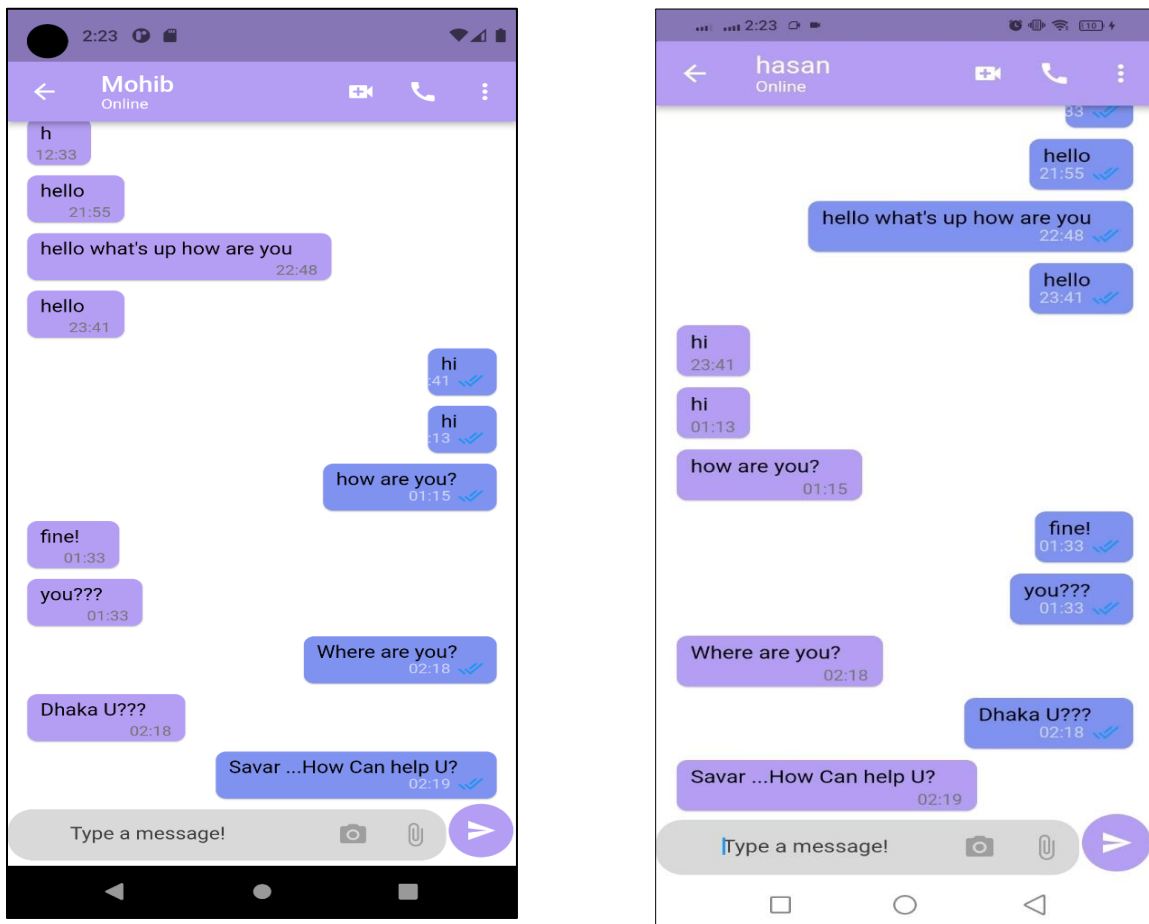


Figure 5.3.1: Messaging Testing

In our project, some features need to be run perfectly and flawlessly. One of the key features of our app is the ability to send and receive messages. Our major emphasis feature is secure messaging. We must keep track of whether these functions are functioning properly. The internet connection is essential in our application. Our app is a messaging app that operates in real time. This implies that in order to send or receive a message, we

must have an internet connection. Firebase is a real-time database that is available over the internet that we utilize for our data sharing. If a user's internet connection is off, any message he sends will not reach the database, and the person to whom he sent the message will not get it.

Because of any faults, all exceptions must be carried out during testing. All test and error messages must be fully demonstrated. Some testing results are given below.

### 5.4 Test Results and Reports

The procedure of implementation and all test findings determine how successful this project will be. This app has a nice interface. This app has a nice interface and is easy to use. Having occasional issues with language settings but everything else is working perfectly. From exchanging messages to the rest of the important tasks are running smoothly. Several screenshots are shared through which we will understand the test results better. The user can enter their name, email address, and a feedback message here. When the user presses the send button, the message is saved in our database under the feedback area. If the user leaves any of these three blanks, the message will not be sent to the database, and a warning will be sent to the user.

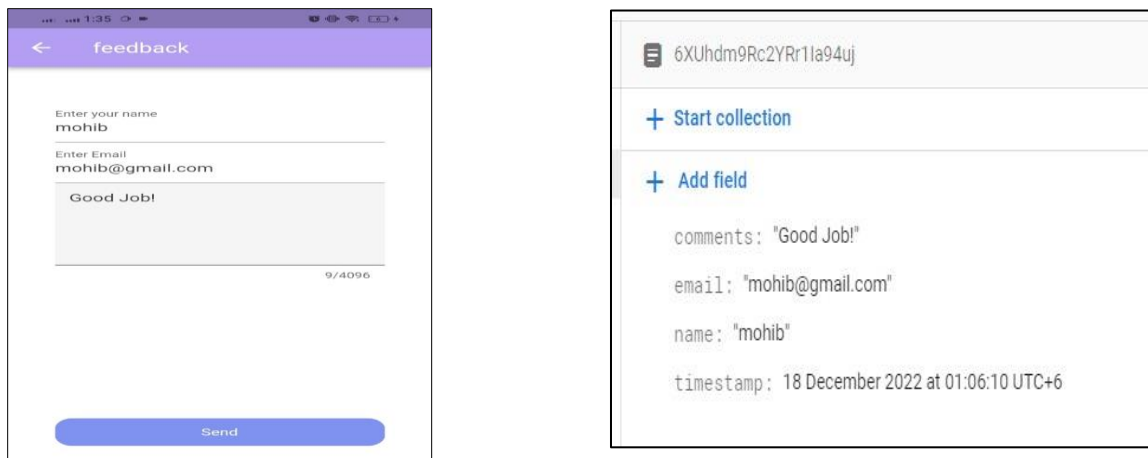


Figure 5.4.1: Feedback Testing Result

Figure 5.4.1 shows that our feedback option is operational. It is a very helpful function for both users and administrators since users may quickly provide suggestions or report straight to administrators. Admin act based on user suggestions or reports.

## CHAPTER 6

### IMPACT ON SOCIETY, ENVIRONMENT, SUSTAINABILITY

#### 6.1 Impact on Society

EnChat is an android application that will act as a secure communication medium for people. It will contribute in various ways to our life. The achievements and contributions of the APP are listed below,

- 1.This application can make human communication safe and easy.
- 2.Communication through the "EN" chat is completely free. So, there will be no cost to the users to use the app.
- 3.It makes messaging smooth with its excellent navigation system.

#### 6.2 Impact on Environment

Environmental impact is always important. Our app has no direct environmental impact.

#### 6.3 Ethical Aspects

It is a thorough report that outlines the objectives of the project, including its technical, linguistic, and financial components.

Our top priority is to protect user privacy. Providing complete privacy to users on the Internet is important because it gives users control over their identity and personal information.

If a user faces a problem through the app, they can give feedback about the problem. In the case of nudity, violence, terrorism, harassment, and other harmful messages, the user can give feedback against them and if any user is involved in any such activity, his/her account will be cancelled, and the user must accept it. User data is never shared with third parties through this app.

#### 6.4 Sustainability Plan

The modern era is dominated by technology and rife with rivalry. A new app was made today, and after some time, a different app with some additional features would appear, lowering the worth of the first app. We have some ambitious ideas for this competition to make our app viable.

This mobile app is accessible to users without charge, so there is no cost to users to download the app. Being ad-free makes it much more comfortable to use. There is no subscription fee to use the app. The app has a beautiful and easy-to-use interface that will be fun for the users. It is user friendly so people of any age can easily use it. The app is designed with the idea that users can easily communicate with each other while keeping their privacy intact. In case of any problem, users can give feedback to the authority, and the problem will be solved. Measures have been taken to avoid shoulder surfing issues so that no one else can monitor your messages and learn your information.

## CHAPTER 7

### CONCLUSION AND FUTURE PLAN

#### 7.1 Discussion and Conclusion

Since the introduction of Android, the market for mobile software has seen significant growth in the developed world. The fact that Android is an open-source operating system is the main driving force behind this. Nevertheless, there are numerous Android messaging apps accessible. However, the Internet's security is deteriorating day by day. As a result, everyone is constantly concerned about message theft in conversation. In addition, several techniques, such as malware, phishing, MITM, DDoS attacks, SQL injection, zero-day exploits, social engineering, etc., are being used to hack into people's social media accounts. Although there are already several strategies used to stop them, this software will surely contribute to a significant change in this industry.

This program will undoubtedly be beneficial to individuals all around the world. It is specifically created for people who are concerned about their security. Those who are less concerned about security, on the other hand, can benefit from this application. Nowadays, different enticing advertisements or lures steal people's social media credentials. They then enter into their account and use their private communication, private photos, and videos to harm one or more people. If such a messaging app can be made for them, they will be protected from such events. Currently, there is an important aspect of cyber security that many ordinary citizens in our country do not properly understand. As a result, they are continually confronted with such issues. This program is based on Message Encryption System. To view messages in conversation, a private key is required, which is not usually stored in this app. If the user prefers, he can use regular chat option instead of using secure chat. As a result, if someone else wants to view his conversations, he won't be able to do so unless he has the private key. This app also has the ability to make regular calls and send messages.

Overall, this application is really simple to use. For individuals who can't remember their private key, the regular chat option comes in useful. It also secures. Because the app is for messaging, it will work on any internet-connected smart phone. Users who utilize this application will no longer have to worry about their conversations being stolen or exposed.

## **7.2 Limitations**

- To use EnChat user must need an Android phone.
- To send messages, users must have at least 3G internet access.
- Firebase free database limit 1 GB.
- Only 100 users can connect simultaneously.

## **7.3 Scope for Further Developments**

We are presently preserving all the basic messaging features within the app. As a security feature, it currently offers message encryption and decryption. This option can be upgraded and made more secure in the future. We may also work on interface to make it extra user friendly. In addition, the app's user interface enhancement and different functionality, such as encrypted folder sharing and private key with biometric identity, can be added. It will also be available to iOS users. Moreover, depending to government clearance, a more secure version of it can be opened for use by the country's security forces.



## APPENDIX

### Appendix-A Backend Code

This element is an expand a part of chapter 4.2: Back-end design. Here we've got proven the real Dart code of a particular task.

#### Login

```
31 void signInWithPhone(BuildContext context, String phoneNumber) {
32     authRepository.signInWithPhone(context, phoneNumber);
33 }
34
35 void verifyOTP(BuildContext context, String verificationId, String userOTP) {
36     authRepository.verifyOTP(
37         context: context,
38         verificationId: verificationId,
39         userOTP: userOTP,
40     );
41 }
42
43 void saveUserDataToFirebase(
44     BuildContext context, String name, File? profilePic) {
45     authRepository.saveUserDataToFirebase(
46         name: name,
47         profilePic: profilePic,
48         ref: ref,
49         context: context,
50     );
51 }
52
53 Stream<UserModel> userDataById(String userId) {
54     return authRepository.userData(userId);
55 }
56
57 void setUserState(bool isOnline) {
58     authRepository.setUserState(isOnline);
59 }
60 }
61
```

Figure A1: Login back-end code

## Homepage

```
34 class MyApp extends ConsumerWidget {
35   const MyApp({Key? key}) : super(key: key);
36
37   @override
38   Widget build(BuildContext context, WidgetRef ref) {
39     return MaterialApp(
40       debugShowCheckedModeBanner: false,
41       // title: 'UI',
42       // ignore: duplicate_ignore
43       theme: ThemeData.light().copyWith(
44         scaffoldBackgroundColor: whiteColor,
45         appBarTheme: const AppBarTheme(
46           color: appBarColor,
47         ), // AppBarTheme
48
49       textTheme: TextTheme(
50         bodyText1: TextStyle(color: blackColor),
51         bodyText2: TextStyle(color: blackColor),
52       ).apply( // TextTheme
53         bodyColor: Color.fromARGB(255, 0, 0, 0),
54         displayColor: Colors.blue,
55       ),
```

Figure A2: Homepage back-end code

## Chat

```
45 void sendTextMessage(
46   BuildContext context,
47   String text,
48   String recieverUserId,
49   bool isGroupChat,
50 ) {
51   final messageReply = ref.read(messageReplyProvider);
52   ref.read(userDataAuthProvider).whenData(
53     (value) => chatRepository.sendTextMessage(
54       context: context,
55       text: text,
56       recieverUserId: recieverUserId,
57       senderUser: value!,
58       messageReply: messageReply,
59       isGroupChat: isGroupChat,
60     ),
61   );
62   ref.read(messageReplyProvider.state).update((state) => null);
63 }
64
65 void sendFileMessage(
66   BuildContext context,
67   File file,
68   String recieverUserId,
69   MessageEnum messageEnum,
70   bool isGroupChat,
71 ) {
72   final messageReply = ref.read(messageReplyProvider);
73   ref.read(userDataAuthProvider).whenData(
74     (value) => chatRepository.sendFileMessage(
75       context: context,
76       file: file,
```

Figure A3: Chat option back-end code

## Secure Message

```
138     color: greyColor,
139     elevation: 10,
140     child: Column(
141       mainAxisAlignment: MainAxisAlignment.spaceEvenly,
142       children: [
143         Text(
144           'Secure Chat',
145           style:
146             TextStyle(fontSize: 20, fontWeight: FontWeight.bold),
147         ), // Text
148         Padding(
149           padding: const EdgeInsets.symmetric(horizontal: 20),
150           child: Form(
151             key: _formKey,
152             child: TextFormField(
153               style: text16(),
154               controller: randomId,
155               key: ValueKey('Enter Private Key'),
156               validator: (value) {
157                 if (value!.length < 5 || value.length > 5) {
158                   return 'Please enter 5 digit room ID';
159                 }
160                 return null;
161               },
162               onSave: (value) {
163                 roomId = value;
164               },
165               onChanged: (value) {
166                 roomId = value;
167               },
168               keyboardType: TextInputType.number,
169               decoration:
```

```
31   @override
32   Widget build(BuildContext context) {
33     return StreamBuilder(
34       stream: FirebaseFirestore.instance
35         .collection('chat/rooms/' + roomId.toString())
36         .orderBy('time', descending: true)
37         .snapshots(),
38       builder: (ctx, AsyncSnapshot<QuerySnapshot> streamSnapshot) {
39         if (streamSnapshot.connectionState == ConnectionState.waiting) {
40           return Center(
41             child: CircularProgressIndicator(), // You, now * Uncommitted
42           ); // Center
43         }
44         final documents = streamSnapshot.data?.docs;
45         var currId = FirebaseAuth.instance.currentUser?.uid;
46         return documents != null
47           ? ListView.builder(
48             reverse: true,
49             itemCount: streamSnapshot.data?.docs.length,
50             itemBuilder: (ctx, index) => MessageBubble(
51               documents[index]['text'],
52               documents[index]['userId'] == currId,
53               documents[index]['userImage'],
54               documents[index]['time'].toDate(),
55               key: ValueKey(documents[index].id)) // MessageBubble //
56           : Center(child: Text('No Messages in this room'));
57       }); // StreamBuilder
58   }
59 }
```

Figure A4: Secure Message back-end code

## Firestore Database Connection

```
17 class DefaultFirestoreOptions {
18   static FirestoreOptions get currentPlatform {
19     if (kIsWeb) {
20       return web;
21     }
22     switch (defaultTargetPlatform) {
23       case TargetPlatform.android:
24         return android;
25       case TargetPlatform.iOS:
26         return ios;
27       case TargetPlatform.macOS:
28         return macos;
29       case TargetPlatform.windows:
30         throw UnsupportedError(
31           'DefaultFirestoreOptions have not been configured for windows - '
32           'you can reconfigure this by running the FlutterFire CLI again.',
33         );
34       case TargetPlatform.linux:
35         throw UnsupportedError(
36           'DefaultFirestoreOptions have not been configured for linux - '
37           'you can reconfigure this by running the FlutterFire CLI again.',
38         );
39       default:
40         throw UnsupportedError(
41           'DefaultFirestoreOptions are not supported for this platform.',
42         );
43     }
44   }
45
46   static const FirestoreOptions web = FirestoreOptions(
47     apiKey: 'AIzaSyDxQEjeAmUItWUH6-QN-jXSWl1_vNUM_ik',
48     appId: '1:79654548800:web:d5ece465a655594036f6a5',
```

Figure A5: Connection with Database

## Appendix-B Database Design

This section is an expand a part of chapter 5.1: Implementation of Database. Here we have given screenshot of firebase.

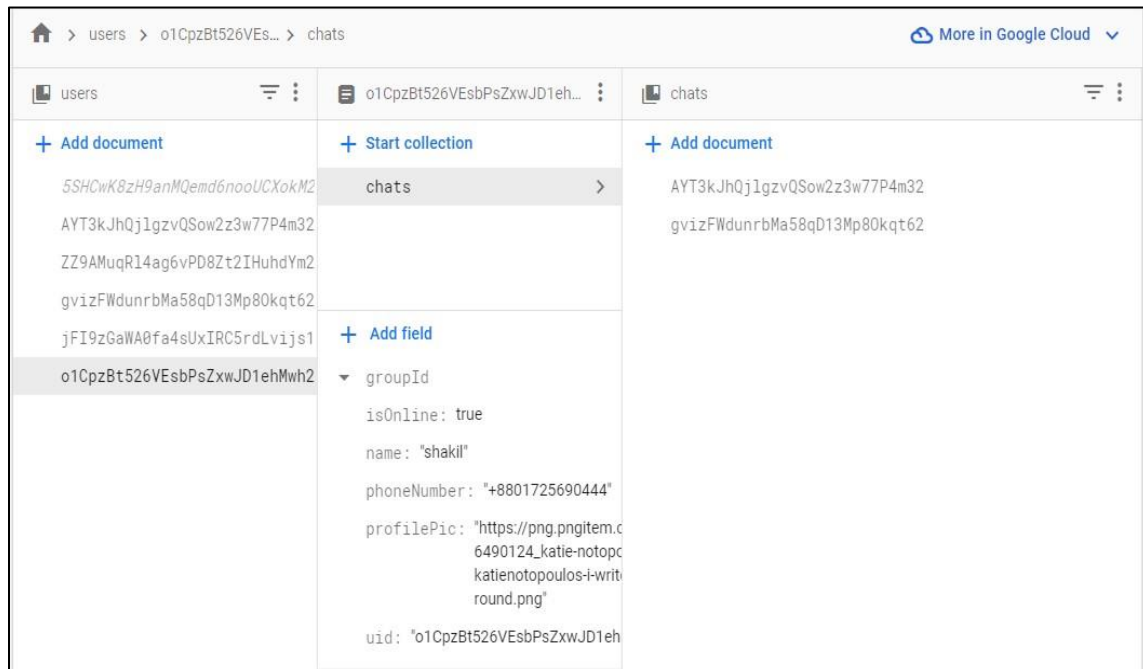


Figure B1: Realtime Database of Chatting

## Appendix-C GUI Design

This section is an expand a part of chapter 4.3: Interaction Design and User Interface. We've shown a few major GUI features from our project. Here, we've also included a few additional interfaces for our project's sub-activities.

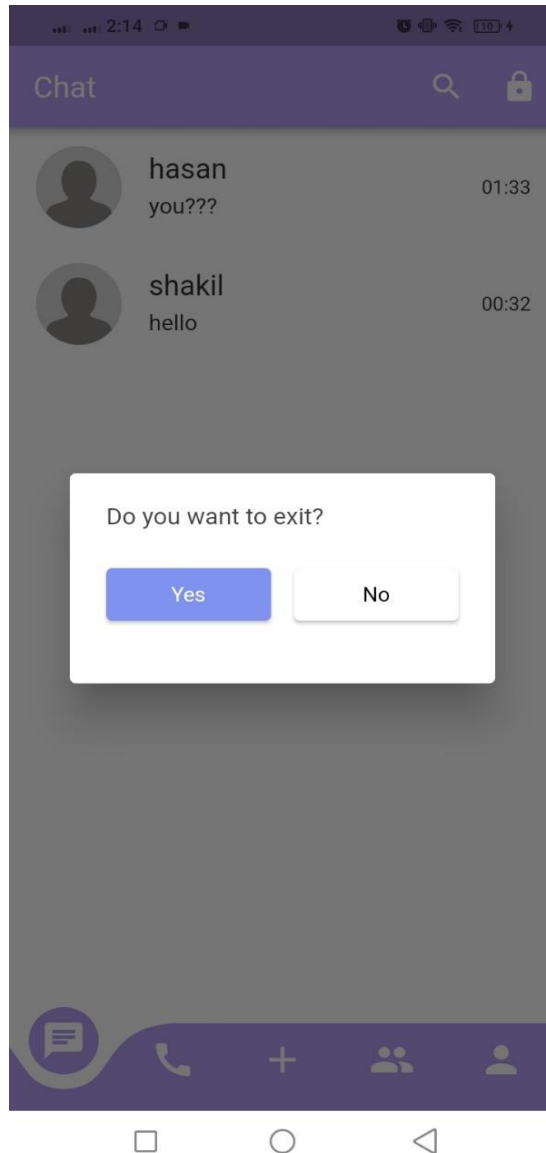


Figure C1: Exit Option

The exit option is shown in Figure C. After enjoying the app's features, users may return to the main page. They may be accessed from the main page by hitting the back button. A pop-up message will appear if you click the back button. There are two possibilities, one of which is yes, and the other is no. By pressing yes, the user exits the app; by clicking on it, the user returns to the home screen.



## REFERENCE

- [1] Flutter documentation <<<https://docs.flutter.dev/>>>, last accessed on 19/12/2021 at 08.00 PM.
- [2] Flutter Mapp <<<https://www.youtube.com/@FlutterMapp>>>, last accessed on 16/02/2022 at 07.30 PM.
- [3] Udemy <<<https://www.udemy.com/>>>, last accessed 25/12/2021
- [4] Dart documentation <<<https://dart.dev/guides>>>, last accessed on 03/1/2022 at 6.00 PM.
- [5] Github <<<https://github.com/>>>, last accessed on 27/03/2022 at 08.35 PM.
- [6] Developer documentation for Firebase <<<https://firebase.google.com/docs>>>, last accessed 05/04/2022 at 09.50 PM.
- [7] Eiband, M., Khamis, M., Von Zezschwitz, E., Hussmann, H. and Alt F., “Understanding shoulder surfing in the wild: Stories from users and observers” pp. 4254-4265, May 2007. [ CHI Conference on Human Factors in Computing Systems].
- [8] Design and material collection <<<https://m3.material.io/develop/flutter>>>, last accessed on 16/04/2022 at 08.27 PM.
- [9] Encryption and decryption in flutter <<<https://genuinecoder.com/encryption-and-decryption-in-flutter>>>, last accessed on 22/05/2022 at 06.35 PM.
- [10] Flutter Cryptography <<<https://fluttergems.dev/cryptography-security-permissions>>>, last accessed on 25/05/2022 at 011.00 PM.

## Plagiarism Check:

Our project report plagiarism less than 30%.

Similarity Index	Similarity by Source
22%	Internet Sources: 21% Publications: 2% Student Papers: 11%

exclude quoted   exclude bibliography   exclude small matches   mode:

quickview (classic) report   print   refresh   download

---

11% match (Internet from 21-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

3% match (student papers from 15-May-2022)  
[Submitted to Daffodil International University on 2022-05-15](#) ✕

---

1% match (Internet from 20-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

1% match (Internet from 21-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

1% match (student papers from 12-Jan-2021)  
[Submitted to Daffodil International University on 2021-01-12](#) ✕

---

1% match (student papers from 18-Dec-2020)  
[Submitted to Southern Cross University on 2020-12-18](#) ✕

---

<1% match (Internet from 26-Oct-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

<1% match (Internet from 21-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

<1% match (Internet from 21-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

<1% match (Internet from 21-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

---

<1% match (Internet from 19-Nov-2022)  
<http://dspace.daffodilvarsity.edu.bd:8080> ✕

[https://www.turnitin.com/newreport\\_classic.asp?lang=en\\_us&oid=1987575625&ft=1&bypass\\_cv=1](https://www.turnitin.com/newreport_classic.asp?lang=en_us&oid=1987575625&ft=1&bypass_cv=1)