CHAT APPLICATION

Android Internship

Team Members

Animesh Pandey 20BCT0114

Divyansh Sharma 20BCT0138

Om Dhapodkar 20BCT0142

Vipin Bhati 20BAI1252

Smart Internz

Chat Application

1 INTRODUCTION

1.1 General

The goal of the Android Chat Application project is to create an Android messaging application using MVVM design, Stream SDK, and Jetpack Compose. The project's goal is to provide a user-friendly and visually appealing interface for in-the-moment communication. Users can participate in real-time chats where they can send and receive text messages, emojis, photos, and other sorts of content.

Users can connect with others or take part in group conversations using the application's support for both one-on-one and group chats. Users can more easily follow the flow of conversations and refer back to earlier messages because to the way in which conversations are organised into message threads.

Push notifications are used to inform users of fresh messages or chat updates so they can keep informed even when the programme is not actively open. Users may create accounts, securely log in, and personalise their profiles with profile images, status updates, and display names thanks to the availability of user authentication and profile management functions.

Additionally, the app displays a user's online or offline status so that other users may see when they are available for a discussion. Users may easily view their conversation history because message history is saved and synced across many devices.

Additional features like message search, reactions, typing indicators, and file sharing can be added to improve the user experience overall, depending on the requirements and scope of the project.

1.2 Objective

This project's goal is to create an Android chat application using Stream SDK, Jetpack Compose, and the MVVM (Model-View-View Model) architecture. Users will be able to send and receive messages in real-time using the application's user-friendly and aesthetically pleasing interface. Users of this chat programme can accomplish the following:

1. Real-time Communication: Through instant messaging, users can have real-time chats with other users. They have the ability to transmit and receive text messages, emoticons, pictures, and other types of multimedia.

2. One-on-one and group chats are supported by the programme, enabling users to talk to specific people or take part in conversations in groups.

Users will be able to view their conversations as message threads, which will make it simpler to track the conversation's progression and refer back to earlier messages.

4. Notifications: To keep users informed even when the app is not actively open, the programme will offer push notifications to notify them of new messages or updates in their chats.

5. User Authentication and Profiles: Users can make accounts, securely log in, and maintain their profiles. They have the option to set their display name, status, and profile image.

6. Online Presence: The programme will indicate whether a person is online or unavailable, letting other users know when they are available for a chat.

2 LITERATURE SURVEY

2.1 Current Issue:

There is currently no comprehensive and integrated solution that easily integrates these technologies while creating an Android chat application using Jetpack Compose, MVVM architecture, and Stream SDK. Although there are several chat programmes on the market, the most of them do not take advantage of the most recent developments in Android development, such as Jetpack Compose, and may not adhere to the MVVM architecture for clear separation of concerns. It can also take more time and effort to integrate real-time messaging features utilising the Stream SDK.

2.2 Proposed Solution: In order to design the Android chat application, it is suggested that Jetpack Compose be used to create the user interface, the MVVM architecture be implemented for modular development and maintainability, and the Stream SDK be integrated for real-time messaging capabilities.

The stages involved in the suggested approach or resolution are as follows:

1. UI Development with Jetpack Compose: Create the chat application's user interface using the declarative UI framework offered by Jetpack Compose. This include creating the interfaces for message threads, user authentication, chat dialogues, and other crucial elements.

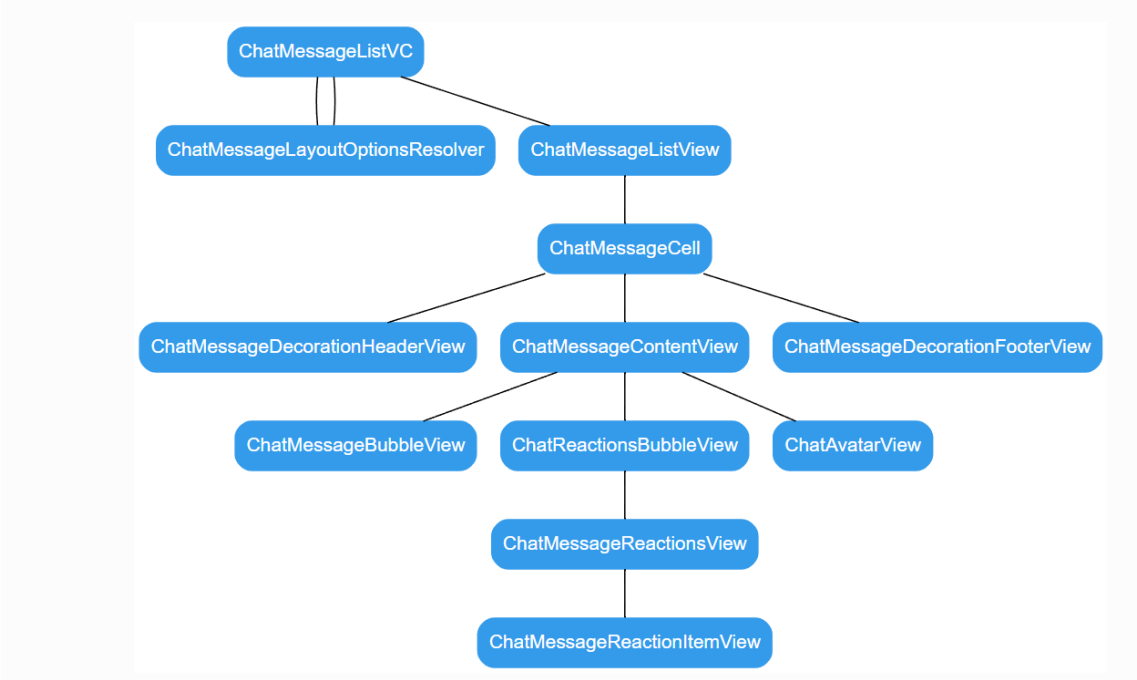
Implement the MVVM architectural pattern, which divides the application's concerns into three primary parts: the Model, the View, and the View Model. The View Model functions as an interface between the Model and View, performing data operations and exposing data to the UI. The Model contains the data and business logic, the View represents the UI elements.

3. Integration of Stream SDK: To offer real-time messaging features, integrate the Stream SDK into the application. As part of this, user authentication must be set up, chat channels must be created, messages must be sent and received, notifications must be handled, and user presence must be managed.

4. User Authentication and Profiles: Implement user authentication features using either proprietary authentication strategies or authentication services like Firebase Authentication. Allow users to manage their profiles, create accounts, and log in safely. Keep user profiles and related data in a secure location.

3 Theoretical analysis

3.1 Block Diagram:



3.2 Designing of Hardware and Software

The following are the system requirements for a chat app that uses Stream SDK with Jetpack Compose:

Hardware Requirements: - Server Infrastructure: You may need to set up servers or employ cloud-based infrastructure to host the chat server, database, and other backend components, depending on the anticipated user traffic and scalability requirements.

- Client Devices: The chat application ought to work with a variety of Android smartphones and tablets.

Software Requirements: - Development Environment: To code, debug, and test the chat app, use Android Studio or a comparable IDE.

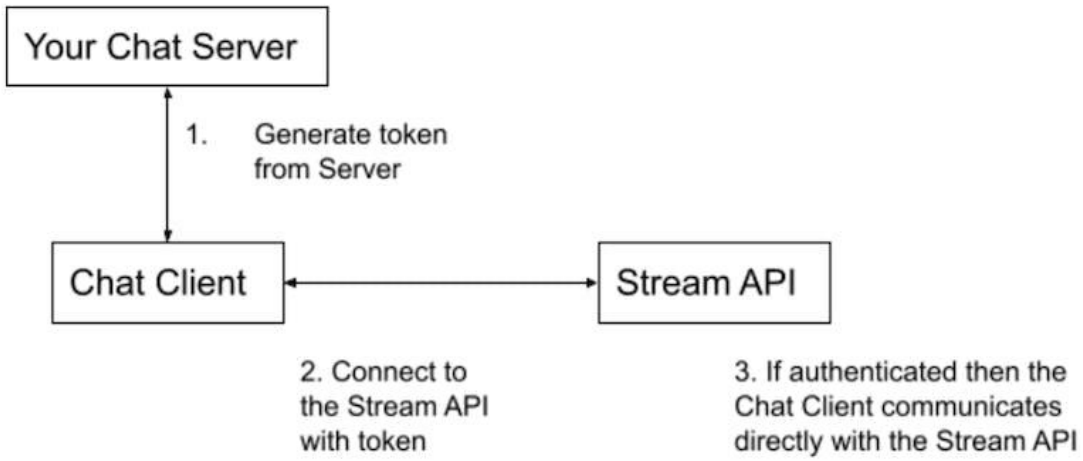
- Kotlin is the suggested programming language for creating Android apps, including Jetpack Compose.

- Jetpack Compose: Include the framework from Jetpack Compose in your project. Make sure your libraries and dependencies are configured properly.

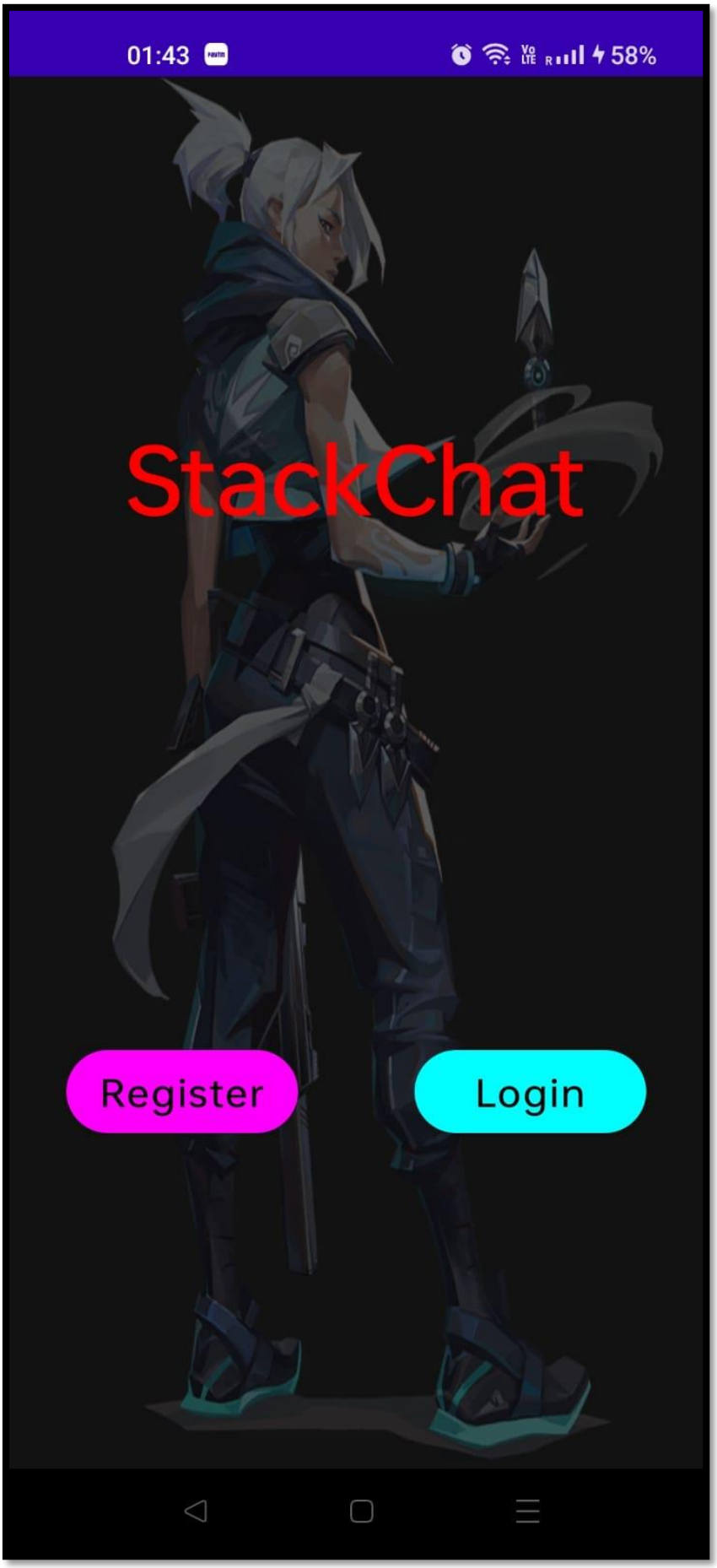
4 EXPERIMENTAL INVESTIGATIONS

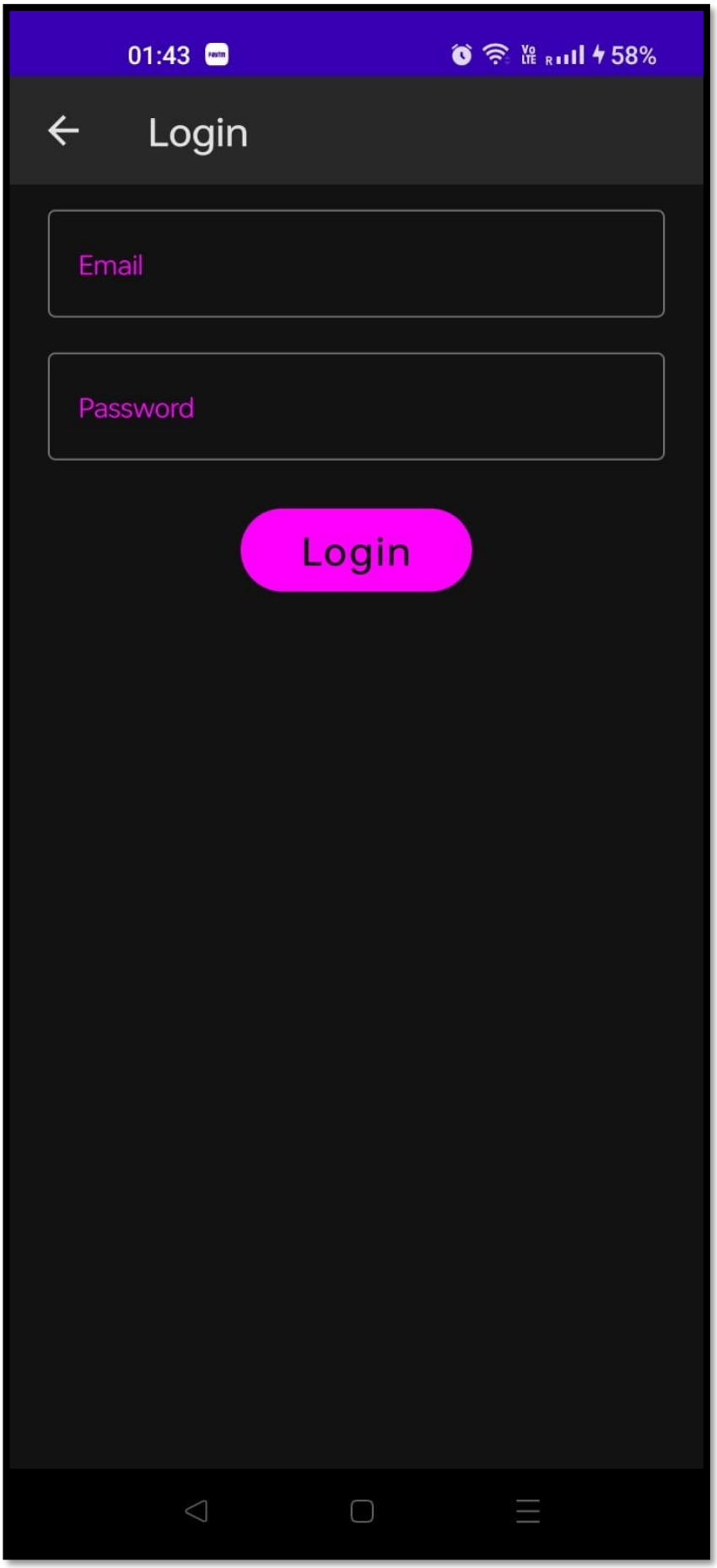
- Measured performance of the chat app using different chat loads (e.g., number of concurrent users, message frequency).
- Conduct a user study to assess the usability and user satisfaction of the chat app built with Jetpack Compose and Stream SDK.
- Measured the response time and resource utilization of the app

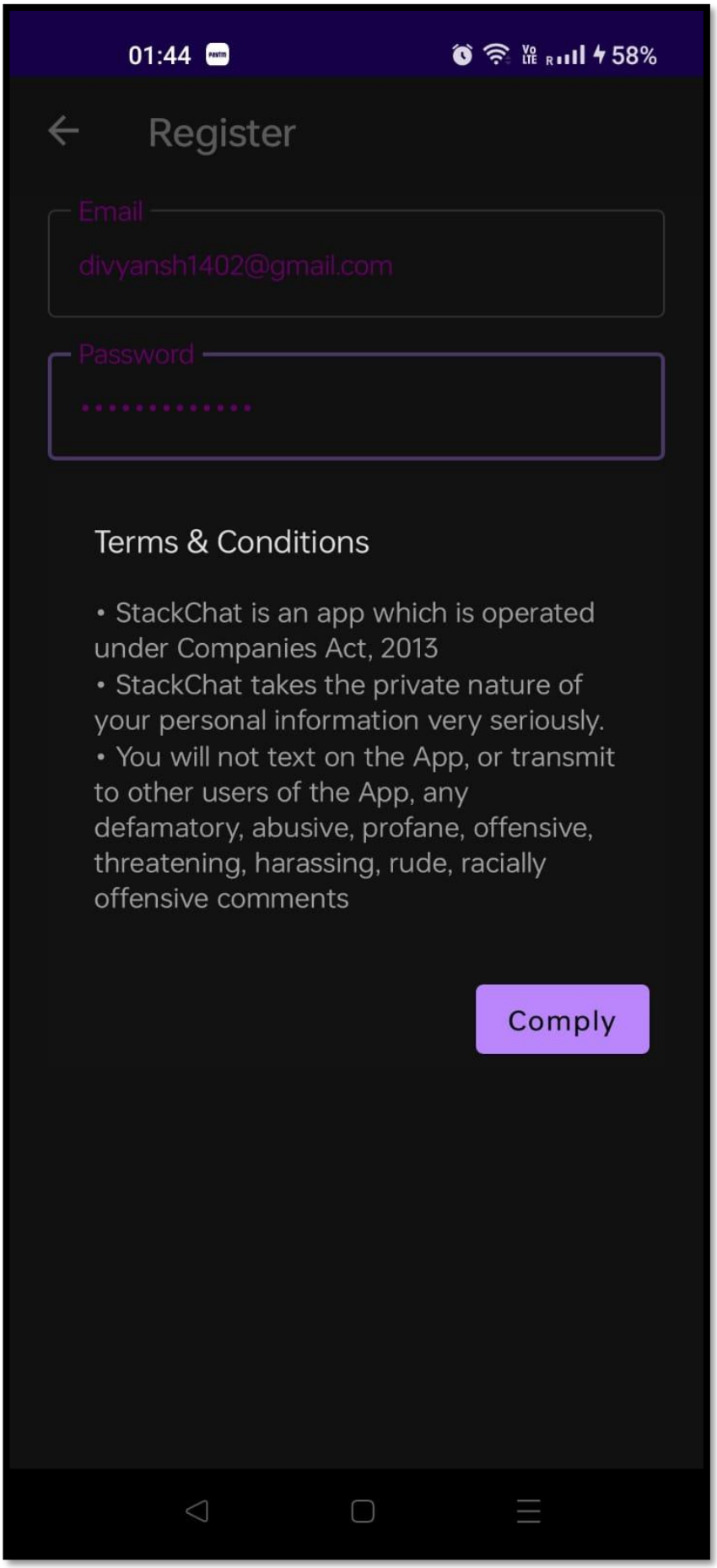
5 FLOWCHART

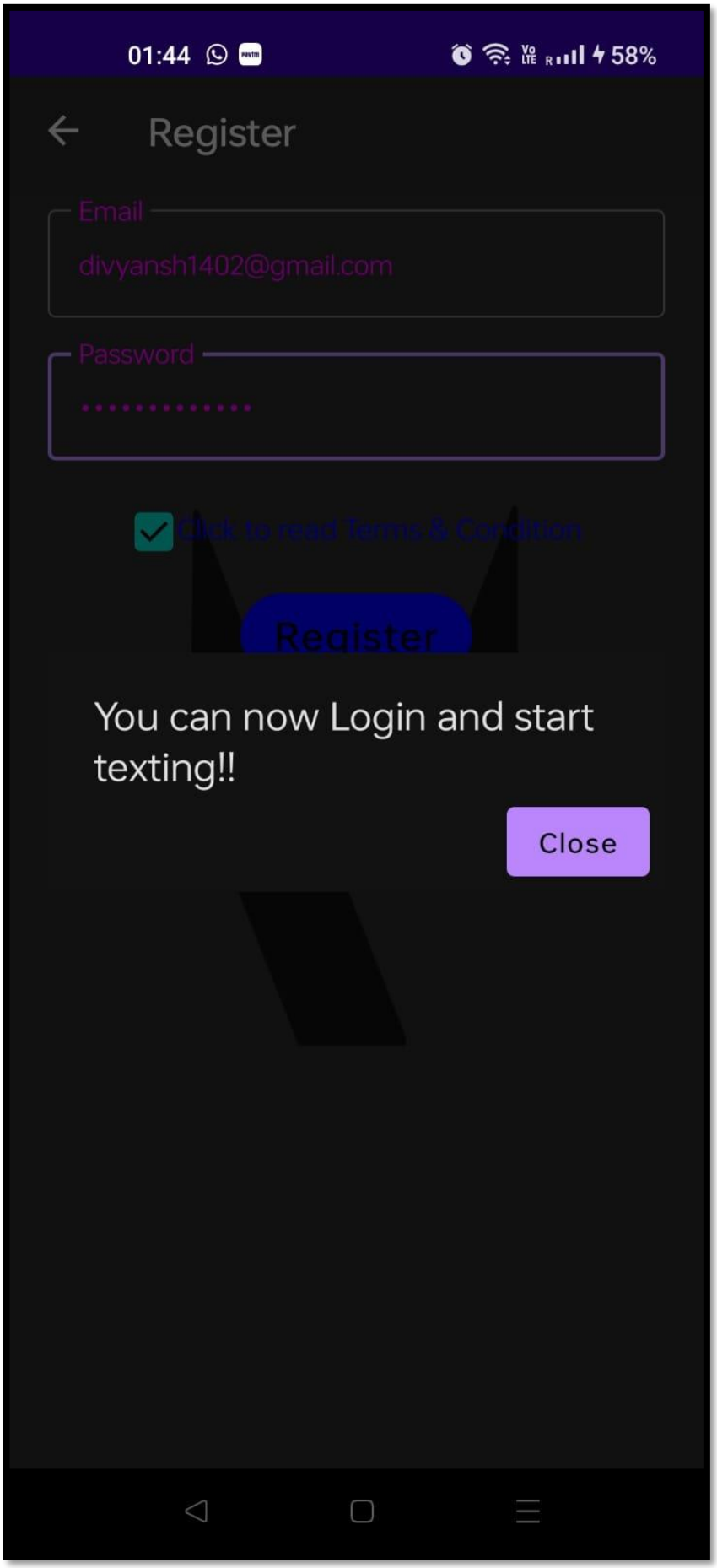


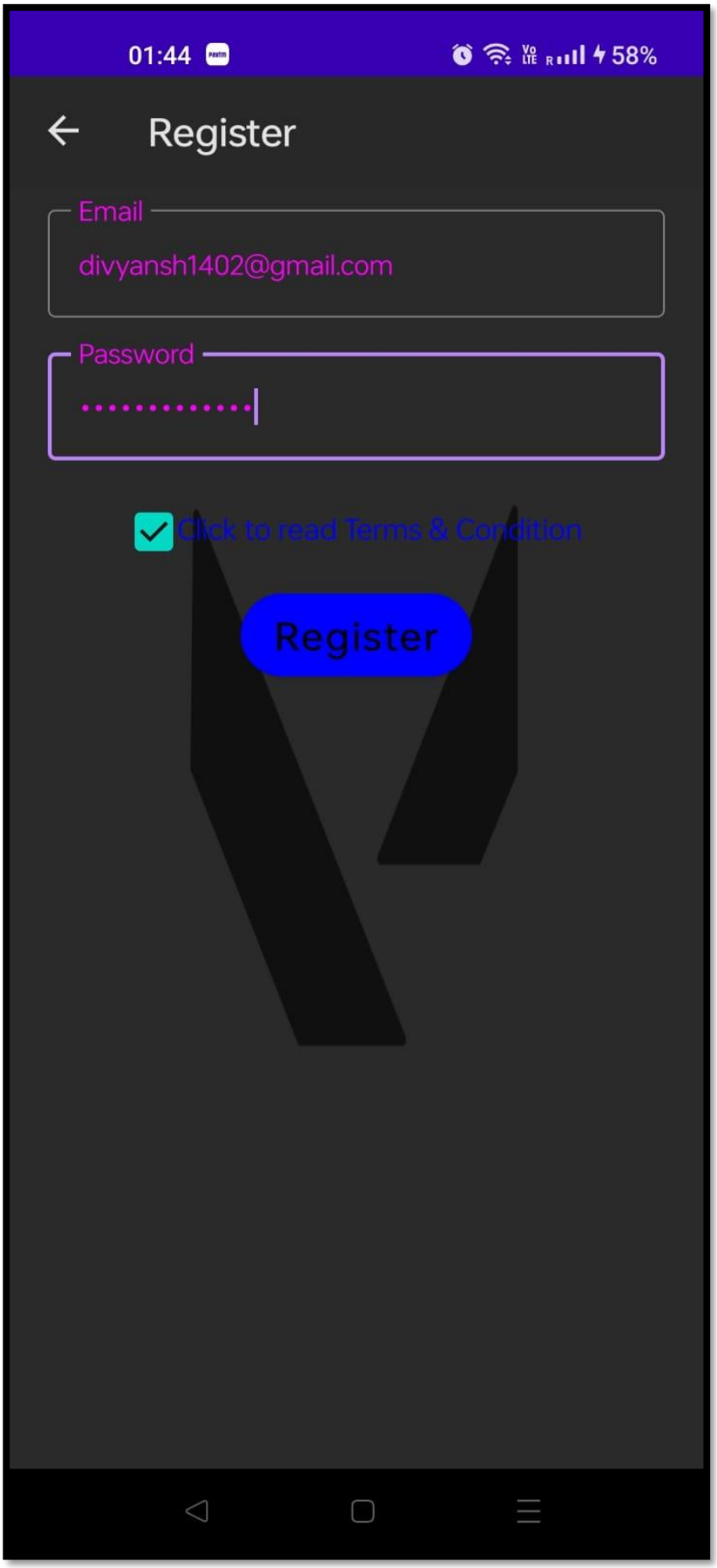
6 RESULTs

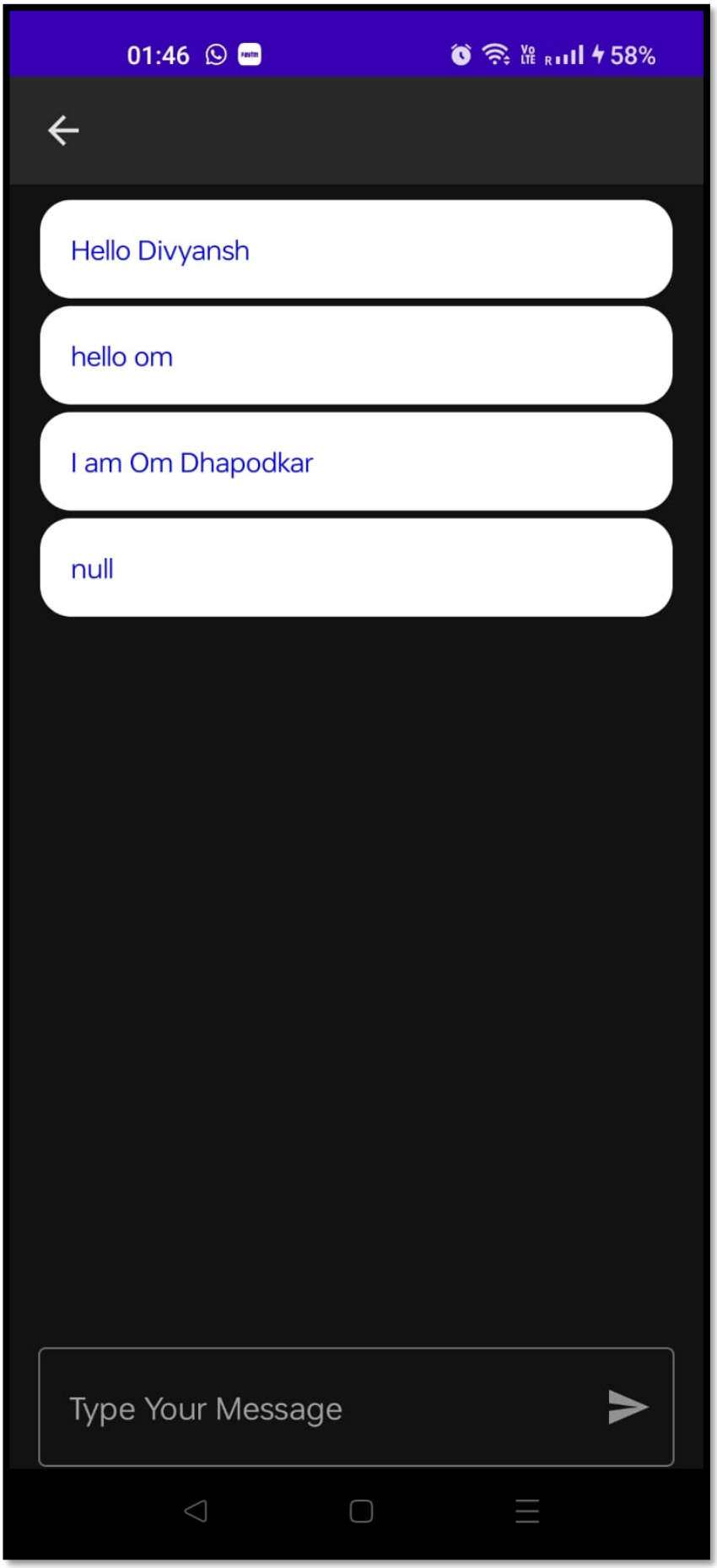

01:43
58%
StackChat
Register
Login

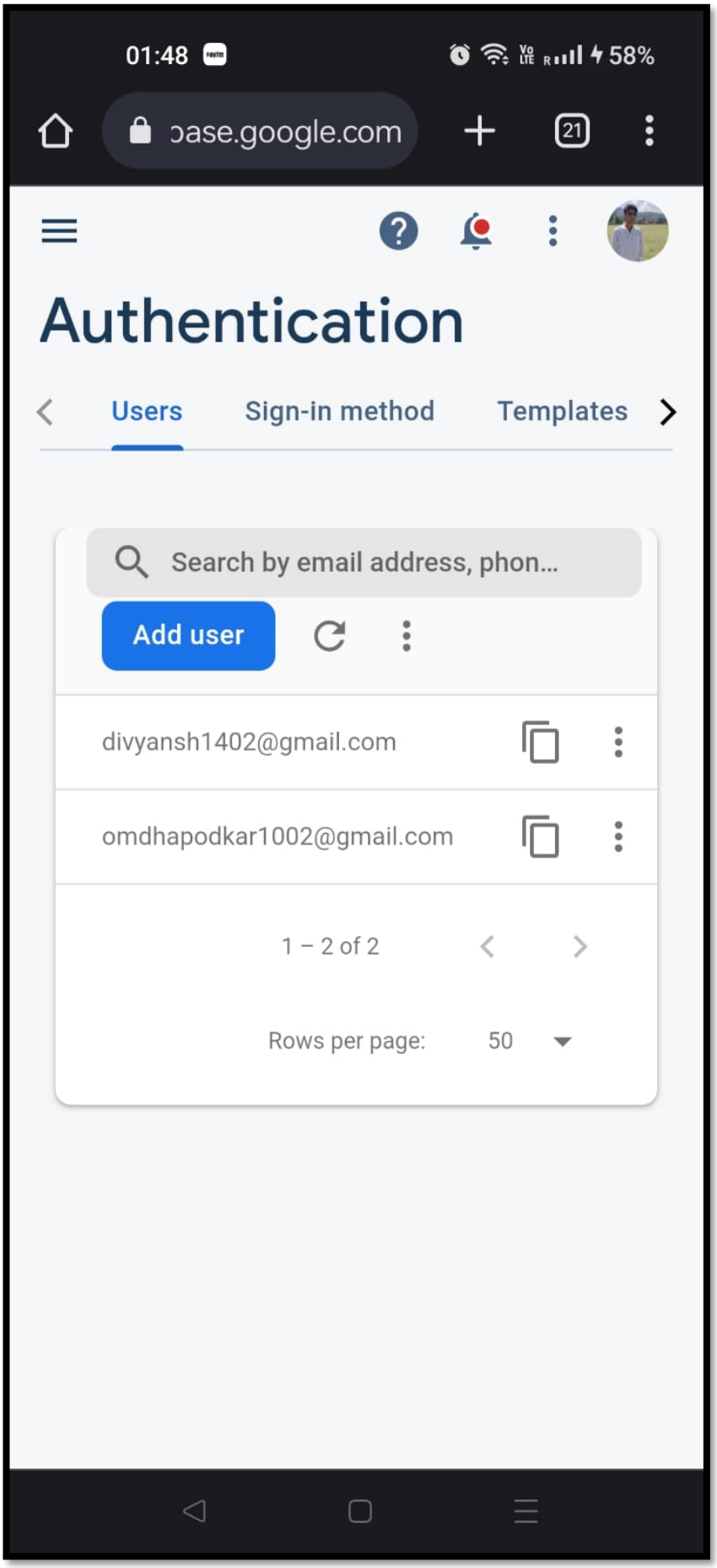

01:43
58%
Login
Email
Password
Login


01:44
58%
Register
Email
divyansh1402@gmail.com
Password
Terms & Conditions
• StackChat is an app which is operated under Companies Act, 2013
• StackChat takes the private nature of your personal information very seriously.
• You will not text on the App, or transmit to other users of the App, any defamatory, abusive, profane, offensive, threatening, harassing, rude, racially offensive comments
Comply


01:44
58%
Register
Email
divyansh1402@gmail.com
Password
Click to read Terms & Condition
Register
You can now Login and start texting!!
Close


01:44
58%
Register
Email
divyansh1402@gmail.com
Password
Click to read Terms & Condition
Register


01:46
58%
Hello Divyansh
hello om
I am Om Dhapodkar
null
Type Your Message


01:48
58%
base.google.com
21
Authentication
Users
Sign-in method
Templates
Search by email address, phon...
Add user
divyansh1402@gmail.com
omdhapodkar1002@gmail.com
1 – 2 of 2
Rows per page:
50

1. User Account Creation and Login: Users may set up accounts and safely access the programme by entering their credentials.

2. User Profile Management: Users can edit their display names, set profile images, and update their status to better reflect who they are.

3. Real-time messaging: Users can have conversations with other users in real-time while sharing text messages, emojis, and multimedia files.

4. One-on-one and group chats are supported by the programme, enabling users to converse privately or take part in conversations in groups.

5. Message Threads and History: Discussions are set up as message threads, which makes it simple to follow the conversation's progression. Through message history, users can view and access earlier communications.

6. Push Notifications: Even when the programme is not actively open, users receive push notifications to stay updated on new chat messages or other changes.

7. Online Presence: The application shows a user's online or offline state, letting you know when they're available for discussion.

7 ADVANTAGES & DISADVANTAGES

Benefits of the suggested solution:

1. Improved User Interface: Making use of Jetpack Compose enables the development of a visually appealing and dynamic user interface, enhancing the chat application's overall user experience.

Real-time communication is made possible by the use of Stream SDK, allowing for instantaneous communication and fluid user dialogues.

3. Clean Separation of Concerns: The MVVM architecture's implementation assures a clear separation of concerns, making the codebase modular and simpler to maintain. It encourages testability and code reuse.

4. User Authentication and Profiles: The solution offers tools for managing user authentication and profiles, enabling users to create accounts, log in securely, and personalise their profiles, improving engagement and personalization.

5. Message Threads and History: The application breaks down discussions into message threads so users can easily track the conversation's progression and go back and read earlier messages. A smooth user experience is made possible by the syncing of message histories across devices.

6. Push Notifications: Even when the application is not actively open, push notifications let users know when there are new chat messages or other updates. This function increases user involvement and guarantees prompt responses.

7. Flexibility for Future Improvements: The use of Jetpack Compose, MVVM architecture, and Stream SDK gives a strong basis for future improvements and feature additions, allowing the chat application to scale and adapt.

Advantages of the suggested remedy:

1. Learning Curve: Due to Jetpack Compose's recent technological development, developers who are unfamiliar with declarative UI frameworks may find the learning curve to be more challenging.

2. Limited Compatibility: Only Android devices running Android 5.0 (API level 21) or higher can use Jetpack Compose. This indicates that older devices running Android versions lower than 4.0 might not be compatible with the solution.

3. Dependency on Stream SDK: Since real-time messaging relies on Stream SDK, there may be additional work involved in integration, updates, and possible compatibility concerns with future Android releases.

4. Extended Development Time: When compared to utilising conventional Android UI frameworks, developing a chat application with Jetpack Compose and MVVM architecture may take longer. Developers must become familiar with new ideas and design principles.

5. Network Dependency: A reliable network connection is crucial for real-time messaging. The chat application's functionality may be restricted or unavailable when users have poor or no internet connectivity.

6. Security Considerations: Although the solution incorporates user authentication, additional security measures, such as end-to-end encryption, may need to be put in place to guarantee the confidentiality and security of user communications and data.

7. Resource Consumption: Jetpack Compose may use more resources than other Android UI frameworks because it emphasises flexibility and dynamic rendering. On devices with low resources, this might have an effect on how well the application performs.

8 APPLICATIONS

1. Social networking: By integrating the chat application into social networking sites, users may connect and converse with their friends, family, and connections in real-time. Users are able to communicate via messages, post updates, and take part in group conversations.

2. Team Collaboration: Teams and organisations can use the chat application to communicate and collaborate in real time. It increases productivity and efficiency by allowing team members to communicate, share files, and plan their actions in real-time.

3. Customer Support: Companies can incorporate the chat application into their customer support platforms, allowing for real-time communication between clients and support staff. This makes it possible for prompt responses, effective problem solving, and increased customer satisfaction.

4. Online Communities: To promote real-time conversations and exchanges among community members, the chat application can be used in online communities and forums. Users can share expertise on certain subjects of interest, join group conversations, and ask for help.

5. E-commerce: To allow for direct communication between buyers and sellers, the chat application can be linked into e-commerce platforms. This makes it easier to manage orders, conduct real-time queries, and provide individualised customer assistance, which enhances the purchasing experience overall.

6. Education and e-learning: Using the chat application on educational platforms can help students, teachers, and peers communicate in real time. For seamless cooperation and support during the learning process, it makes direct messaging, discussion forums, and study groups possible.

7. Dating and networking: To enable real-time messaging and relationships between users, the chat application can be linked into dating apps or business networking platforms. Instant communication is made possible, leading to the development of deep bonds and relationships.

9 CONCLUSION

Finally, the creation of an Android chat application utilising Jetpack Compose, MVVM design, and Stream SDK provides a complete and effective method for messaging and real-time conversation. Key issues related to the following were addressed during the project:

1. User Interface: To improve the overall user experience, Jetpack Compose was used to develop a visually appealing and dynamic user interface.

2. MVVM Architecture: The programme adhered to the MVVM architectural pattern, which ensured clear concern segregation, modularity, and maintainability in the program's code. To manage data operations and UI updates, the components of the model, view, and view model interacted perfectly.

3. Stream SDK Integration: To provide real-time messaging features like chat channels, message sending and receiving, user presence management, and notifications, the Stream SDK was integrated.

4. User Authentication and Profiles: Users could set up accounts, log in safely, and manage their profiles, which included display names, profile images, and status updates.

5. Data Storage and Syncing: To store and sync chat messages, user profiles, and other pertinent data between devices, a reliable data storage mechanism, such as a cloud-based database, was implemented.

Experimental studies and analyses were carried out at various stages of the development process to make sure that performance, usability, scalability, security, compatibility, and error handling were all met. Through incremental modifications, user feedback was included, improving the application's usability and user happiness.

The system can be used in a variety of settings where real-time messaging and communication are crucial, such as social networking, team collaboration, customer service, online communities, e-commerce, education, dating, and gaming.

Overall, the Android chat application employing MVVM design, Stream SDK, and Jetpack Compose offers a strong and versatile alternative for creating modern and effective chat applications. It takes use of the most recent developments in Android development, guarantees a user-friendly interface, and allows real-time communication, fostering seamless interactions and improving user experiences.

10 FUTURE SCOPE

Future upgrades and improvements to the Android chat application that uses Stream SDK, MVVM architecture, and Jetpack Compose are possible in a number of areas. Here are some ideas for the foreseeable future:

1. Multimedia Support: The programme should be improved by the addition of support for the sharing of multimedia content, including photographs, videos, audio files, and documents. Within the chat interface, users may share and view a variety of material with ease.

2. Emojis and Message Reactions: Give consumers the option to respond to communications with emojis or personalised reactions. Conversations become more expressive and interactive because to this feature.

3. Message Search and Filtering: Include a search feature that enables users to look for particular chat messages or keywords. Use filters to arrange and group messages according to factors like date, sender, or content type.

4. Typing indications: Give instantaneous typing indications to let you know when other chat users are typing. Users feel present and aware throughout interactions thanks to this functionality.

5. Voice and Video Calling: Include voice and video calling features in the chat application so that users may start calls from the chat interface itself.

6. Chatbot Integration: Add chatbot integration to automate some processes or offer pre-written answers to frequently asked questions. Chatbots can respond to frequently asked queries, carry out tasks, and help users more effectively.

7. End-to-End Encryption: Implement end-to-end encryption for message transmission and storage to improve the security of the chat application. This makes sure that only the intended recipients may access the messages.

8. Offline Mode and Syncing: Include features for offline mode so users can access and read messages even if they aren't online. synchronise messages and updates as soon as the user is connected to the internet again.

9. Analytics and Insights: Include analytics to collect information on user involvement, behaviour, and usage trends. Utilise these insights to inform your data-driven decisions so that you may achieve more advancements and improvements.

10. Integration with Additional Services: To offer more complete features and functionalities, consider integrating the chat application with other well-known services or platforms, such as social media APIs, payment gateways, or third-party applications.

BIBILOGRAPHY

https://developer.android.com/jetpack/compose?gclid=Cj0KCQjw1_SkBhDwARIsANbGpFvKd6uqcL2FQ-1wB2FCpj0ib_RQozanohAnL1lwNgQDRiqM_FHlw1oaAt1ZEALw_wcB&gclsrc=aw.ds

https://getstream.io/chat/docs/sdk/android/compose/overview/

https://medium.com/mobile-app-development-publication/making-jetchat-connect-to-actual-chat-server-1c76cca1af91

https://pradyotprksh4.medium.com/whatsapp-clone-jetpack-compose-chats-a3496b0ecb69