Developing Android Social Media Application - GabChat

Akhil Varghese
3061075

Submitted in partial fulfilment for the degree of

Master of Science in Computing

Griffith College Dublin
June, 2022

Under the supervision of Barry Denby

Acknowledgements

Before I begin the dissertation, I'd want to express my gratitude to all who have assisted in some way in the creation of this document.

- **Professor Barry Denby:** Since I began my thesis, he has provided me with a wealth of advice, direction, knowledge, and support. Without his guidance, I would not be capable of creating this piece of research. His extensive understanding and wealth of experience have always motivated me throughout my thesis.
- **Family and Friends:** There are far too many individuals to mention here, but they all contributed in some manner, whether it was by small pieces of advice or support.
- **Griffith College:** For providing this opportunity to learn and apply.

Table of Contents

List of Figures

Abstract

Our project is social media application that helps to chat, phone calls, video calls, and screen share, which is built on the SendBird framework. The effective use of online social media platforms has created a huge impact on today’s modern world communication. Rapid changes in content sharing methodologies have paved way to new application development which tends to make life easier for people in terms of communication and in transferring messages on a day-to-day basis. Chatting apps usage have increased rapidly over the past decade which has broken all type of barriers with respect to communication around the globe. Irrespective of one’s geography these applications have become handy in transferring contents within seconds to any part of the world at ease. The technology has been around for quite a long time, but the acceptance of this drive was quite recent. Platforms like firebase which are used in real-time systems allows the development of applications very easy and quick and it has also reduced the monotonous use of other database systems in real-time. Along with it, the SendBird framework provides videos with a great deal of reliance and security for storing media content. The indigenous Android framework has given numerous opportunities to build applications with all possible features that can be accommodated on a mobile application.

Chapter 1. Introduction

1.1 Problem Statement

Social media applications are one of the most widely utilised communication and collaboration tools in businesses. They need to communicate with employees, sometimes they need to communicate with customers to solve their issues and customers need to share their screens to solve some issues. For example, if a customer uses a payment application and has some issues. To solve this issue, technical people may want to see the customer's phone screen.

For these usages, we need to install separate applications. Such as screen share apps, chatting apps, video call apps etc. Also, most companies use WhatsApp and Facebook groups to communicate with customers and employees.[28] Most people are using social media applications such as WhatsApp, Facebook, and Instagram to communicate with friends and families, and they use these applications to get relaxation in their professional life.[28] To solve these problems, we intend to construct a social media application which contains messaging, video calls, audio calls, and screen sharing with minimum security.

1.2 Motivation

In all sectors of life, communication plays an important role. The evolution of internet technologies had benefited people to access the web easily. People have been communicating with each other through various applications or mediums for a long time. Using social media applications people can communicate with others in different ways. They can send messages, share files, talk to each other through video calls and audio calls etc. Social media applications encourage people to react the social issues and they will encourage youngsters to communicate with society. Moreover, there are some negative aspects of social media. The main issue is everyone shares all the details on social media. Hence the data was not secure on social media.[29]

The core issue with all these applications is, that the data belong or is accessible to giant corporations. To address the data security and privacy of application users, we intend to deliver an alternative product that could support most of the latest features like Chat, Audio calls, Video calls, Screen sharing, Screenshot detection, and Encrypted messaging to enhance security.

1.3 Goals and Objectives

The main goal of this project is to develop a social media application. It unlocks features to the users to share messages and place video and audio calls respectively and also allows the user to share their screens. Security is the main factor of every social media. this application has minimum security. Every user's message was encrypted using the AES algorithm and it sends to SendBird to sever. Hence no one can understand users' messages without decryption. Moreover, this project contained screenshot detection. When anyone took a screenshot of a chat or video call, another end user will get a notification message. By doing this project we are trying to implement a secured social media application.

1.4 Overview of Development

The overall development involves several steps. Mainly I am using SendBird SDK for the development of this project.

There are several Modules on this application:

Authentication: Firebase authentication was used for the authentication.

Notification: Firebase real-time messaging and firebase notifications were used to get the notification. Mainly users will get the notifications,

1. Make voice calls and video call
2. Share the user’s phone screen
3. When someone took a screenshot of a chat or video call.

Chat: The user can chat with the selected people. It is implemented using Sendbird chat SDK.

Audio Call: Everyone can make a phone call using this application. It is implemented using SendBird call SDK.

Video Call: Everyone can make a video call using this application. It is implemented using SendBird call SDK.

Screen Share: While making a Video call users can share their screen with other people.

Screenshot Detection: if anyone took a screenshot of a chat or video call at that time another end user will get a notification.

These are the main modules of this project. Other details will explain in the upcoming chapters.

1.5 Document Structure

The main content of the document starts with the second chapter which contains a review of the literature in the field of social media applications, and the technologies employed to build this application and its work. The methodologies and design of the application structure is described in the third chapter of the document. The system design and specifications, including any hardware and software that were used, are detailed in the next section. This project's implementation details are detailed in Chapter 5. Details of my project's functional prototype, as well as my testing and evaluation method, are described in Chapter Six. We also go over the outcomes, including any necessary changes to the overall system specs and implementation. Conclusions and future work are elaborated in the last chapter

Chapter 2. Background

2.1 Literature Review

In 1996, the Israeli company Mirabilis created ICQ (I Seek You), which was the first widely used online messenger [19]. In 1961, MIT's Computation Centre developed the Compatible Time-Sharing System (CTSS), which allowed up to 30 people to login in at the same time and send text messages. The CB Simulator, created by CompuServe in 1980, is widely regarded as the first dedicated online chat service; users had to pay for a subscription to participate.[19]

Q-Link (Quantum Link) is an online service for Commodore 64 and 128 computers that allows multi-person conversation, file sharing and email via modem connection. Later in 1991, Quantum Link was renamed America Online (AOL). In the mid-1990s, it was the most popular Internet service provider in the United States. In 1997, AOL Instant Messenger (AIM) was established. At that time some chatbots are established like, StudyBuddy, smarterchild etc. This helps to have instructions for users while playing games. AIM had 52 per cent of the instant messaging market in 2006. It has a huge competition with Google Talk, Yahoo Chats and Skype also AIM was struggling to monetize and fell out of users favour.

In the 2000s, BlackBerry Messenger was developed, and popularity increased, and it promised a better future for mobile communication.[19] At that time SMS was king and, in some countries, the price of SMS was so huge. At that time, social media applications like Facebook, WeChat, and WhatsApp etc were established and people communicate using these applications.

Video conferencing is another important feature of social media. The first video two-way video conference was established in the 1930s. Compared with the audio transmission, technologically Video transmission was so difficult because of the cameras. The first tv cameras entered the market in the 1920s. On 1927 April 7th, the first Tv real-time communication was established by the AT&T Bell telephone. It telecast a moving image of Hoover, who was a commerce secretary, from the white house of New York over a 200-mile distance. Later in the 1930s, they developed two-way video communication between two AT & T officers, and they can see each other. Later, Bell Telephone developed a two-way video communication system in 1959. But it sent one frame every two seconds, and it has stable and clear visuals. Then, on

1964, April 20th, AT&T presented Picturephone Mode 1 at the world fair in New York. The second version of Picturephone Mode 11 was developed in 1969 with 30 frames per second broadcast speed. But it failed. CLI T1 was the first group video conferencing system launched by Compression Labs in 1980. It needs huge space, Its hardware filled nearly an entire room and cost a whopping $250,000 upfront and each call has 1000 dollar per hour.

In 1984, PictureTel Corp. was founded by MIT students. It was the first video codec with faster data transfers. PictureTel was chosen by AT&T for an international video conference in 1989. It enabled two-way, real-time voice and full-motion video connections between PictureTel and AT&T's Paris office. PictureTel joined IBM's multimedia business partner program in 1991 and began developing a PC-based video conferencing solution. The first webcam was invented in 1991 by Cambridge University students. It pulled photos three times per minute and delivered a 129*129-pixel grayscale picture at one frame per second.

In the early 1990s, a Cornell undergraduate created the CU-SeeMe program. This was the world’s first desktop video conferencing system. It was launched in 1992 for Macintosh and in 1994 for Windows. Connectix is a company which came out with QuickCam, as its first commercial webcam, in 1994. It only comprised of a 320x240 pixel resolution, and it had a grayscale colour depth of 16 shares at 60 frames per second, or 15 frames per second if 256 shades of grey were selected. The QuickCam was purchased by Logitech in 1998[20].

After the innovation of smart phone’s, video calling had dramatic changes. The Kyocera Visual Phone VP-210, introduced in Japan in 1999, was the first smartphone with a front-facing camera. Back during 2003, a number of mobiles, including the Sony Ericsson Z1010, came into use featuring front cameras. But when with the release of the iPhone 4 and FaceTime during 2010, smartphone video conferencing gave huge openings in a big way. We could switch from a voice call to two-way video communication with the touch of a button. It was meant to support Wi-Fi connections at the beginning, but Apple deliberately added 3rdGeneration and 4thGeneration/LTE to its connection.[20]

In August 2003, three Estonian engineers launched Skype. Skype was purchased by eBay in 2005 and Microsoft brought eBay in 2011. It began as a messenger but has since evolved to include video. Two ex-Yahoo developers found an instant messaging application WhatsApp in 2009. It wasn’t until 2016 that it added video chat, two years after it was purchased by Facebook.

From the last decade onwards social media platforms are globally famous. After finding the smartphones social media platforms are part of human life. It provides so easy lifestyles to people. Originally, ‘Classmates’ was the first social media application, founded by Randy Conrad’s in 1995 December. Initially, it was used for school affiliations, and later it adds features like a user-profiles friends list. Later, this company was rebranded as Memory Lane [17]. After that in 1996 May, Andrew Weinreich founded ‘Six Degrees’, it is usually recognized to have been the first social network site by the majority of people. This site also contains users' profiles, friends lists and etc. At that time this site had millions of users and later 20’s it was sold to YouthStream Media Network.
In 2000, James Hong and Jim Young developed a chat and date application named ‘Are you Hot or Not?’. It allows users to rate the photos, uploaded by other people. Its domain was hotorNot.com and is currently owned by hot or not limited. This site makes a significant influence to make a social media site creators such as Facebook and YouTube [18]. In October 2001, the first business professional social network ‘Ryze’ was developed by Adrian Scott [17]. It is also known as the father of LinkedIn. It also had more than 5 lack customers in over 200 countries. Later in the 2000s lots of social media applications are established such as Friendster, LinkedIn, Orkut, Facebook, WeChat etc. Nowadays, social media applications are so famous all over the world. People are sharing files, communicating with others, sharing their thoughts etc.

2.2 Related Work

Currently, there are lots of social media applications are available in the market, which have features such as messaging, audio calling, video calling etc. Among them, we selected a few applications which currently famous on the market.

Facebook

On 2003 October 28th, Harvard University students, Mark Zuckerberg and his college friends Andrew McCollum, Eduardo Saverin, Chris Hughes, and Dustin Moskovitz were found Facebook. Initially named as Face Mash. Later, on February 4, 2004, changed its name to Facebook. At first, the site was only intended for the students at Harvard University. In 2005, it dropped the “The” from its name and was simply named Facebook and by 2006 it became available for public users by the beginning of 2006, it had over 6 million users globally. Facebook acquired Instagram company, a photo-sharing service on 2012 April 12, and on February 19, 2014, Facebook acquired WhatsApp, an instant social messenger service. By the time, 2021 October 28, Facebook was renamed Meta “to reflect its focus on building the metaverse”.[22] There is no denying that Facebook is the most popular site when it comes to Social Networking, as of the first quarter of 2022, there were over 2.93 billion users.[23]

Twitter

In March 2006, Jack Dorsey, Biz Stone, Evan Williams, and Noah Glass established the microblogging service Twitter. Initially, it allowed only 140-character tweets(posts) by a user to its followers. The users will be able to reply, mention, and retweet along with hashtags. Statistics on registered subscribers are not published by Twitter. and hence daily user estimates vary. In February 2009, a blog entry by the site “Compete.com”, Twitter was listed as the third most popular social media platform with 6 million unique monthly visitors. As per the research published in April 2014, it has been found that almost 44% of Twitter users have never used their accounts for any tweeting or making posts.[25] Barack Obama, Justin Bieber, Katy Perry, Rihanna and Cristiano Ronaldo are the top 5 followed accounts. Twitter gathered a lot of steam in April 2022, when business tycoon and the founder of SpaceX and Tesla, Mr Elon Musk offered to purchase the American microblogging site for a buyout of 44 billion USD. By April 2022, there are as many as 465.1 million users on Twitter as per the website DataReportal making it the 14th most active social media site. [24]

Instagram

In 2010, Instagram was founded by Mike Krieger and Kevin Systrom, Instagram was initially released as a photo and video-sharing social networking service. Posts are shared either publicly or for limited followers. The main features of Instagram are, that users can share pictures and videos, and users can like and comment on these shared photos and videos, and users can follow each other, they can see the shared content and tag other users and share locations. Recently they added reels features. So everyone can upload and watch reels. According to the Verge, Instagram introduced an Android version of its application on April 3, 2012, and it was downloaded over one million times in less than 24 hours.[26] On April 12, 2012, Facebook acquired Instagram. As of May 2022, Instagram has roughly 1.21 * billion users worldwide [27]. In 2010, Instagram came in second place for "Best Mobile App" By the survey of TechCrunch Crunchies in January 2011. In September 2011, Instagram was named "Best Locally Made App" at the SF Weekly Web Awards. Instagram was selected the "App of the Year" for 2011 by Apple Inc. in December 2011. Instagram was also awarded No. 1 by Mashable in 2015 on its list of "The 100 best iPhone apps of all time," As per the report from May 2022 in Wikipedia's list of most-followed Instagram users, it was listed that, Cristiano Ronaldo, a Portuguese professional footballer, is the most followed person on Instagram, with over 446 million followers.

Snapchat

Snapchat is a social media application that allows users to communicate via time-sensitive messages or snaps, one that disappears after viewing. It is mostly used by teens to share their everyday activities with friends. One reason why kids love Snapchat is that it offers a lot more innovative stuff like games, news and entertainment, quizzes, photo filters, and photo and video editing tools.[32]

The idea of Snapchat was first developed by Reggie Brown in 2011 and was launched in the same year by Evan Spiegel and Bobby Murphy, the students of Stanford University students, under the name of Picaboo on the iOS platform only. [33] It was later launched on the Android platform as well in 2012.

Any user above the age of 13 can sign up and use Snapchat. Some of the features it offers are Snapstreaks, Snap Map, Snap Story, Snap Code, Snapcash, Snap store and a variety of lenses and filters. A Snapstreak is established when two users send snaps back and forth within a 24-hour period for three days in a row. It allows the addition of a few more emojis and the Snapchat score. Kids often end up consuming their time to increase their scores. Snap Map displays users' real-time location on the application that can be viewed only by Snapchat friends. There are options available to turn the map off or use it in a Ghost mode.[32] As per the sources of Business of Apps data, Snapchat's revenue increased by 63% in 2021 to $4.1 billion, with $1.5 billion generated in the fourth quarter. And the Annual percentage revenue growth increased in 2021. [34]

WhatsApp

WhatsApp is a simple, beautiful ad-free interface application that allows users to communicate via text messages, voice and video calls, group calls, images, and record audio notes. Fortunately, WhatsApp is considered to be most known social media application, for mobile communications with over two billion active people using it on a daily basis. [35] It was invented by ex-Yahoo employees, Brian Acton, and Jan Koum in 2009. Facebook brought all the rights of WhatsApp for $19 billion in 2014 (Forbes). It is said to be widely used in over 180 countries today. [36]

Statista states that prominent users of WhatsApp greater than that of Facebook or Messenger (1.3 billion). One of the most obvious reasons for its popularity is its user-friendliness and high accuracy of end-to-end encryption, clear cut user data and chat privacy. A report by Statista clearly shows how WhatsApp is pouncing up over all other messaging applications. WhatsApp is the most downloaded application in 2017 and 2019. [37]. Although, any change in its privacy policy can trigger a change in its growth.

Chapter 3. Methodology

3.1 Application platform

Mobile platform applications: these are used for mobile operating systems such as Android, IOS etc. [5].

We developed this application for mobile platforms, especially Android. According to a survey in 2022 70% of people in the world, the population were using smartphones [6]. Moreover, in a global survey in 2021 found that 72.2% of people were using android applications and 26.99 per cent of people were using IOS applications [7]. This was the main reason we chose to construct an android application. Moreover. Android applications are so user friendly, and people can easily understand the process of android applications [41].

3.2 Technologies used

For this project, we used Android Studio as IDE (Integrated development environment). Nowadays there are several IDEs are available on the market.

The main available IDEs are,

Eclipse: The Eclipse is the commonly used open-source IDE used for developing applications, according to the PYPL Index, with the most recent version having mostly downloads application, with more than 1.8 million active devices and users from June 2017. Using this we can develop java, PHP, android, c++ applications [9].

IntelliJ IDEA: IntelliJ IDEA is a "capable and ergonomic IDE for JVM," according to the IntelliJ IDEA site. It forms the framework for Android Studio and includes various capabilities for Android development. This IDE has also received multiple honors, and it is ranked sixth in the PYPL ranking, just below NetBeans. It is available in two versions: a public Community version and a purchased Ultimate version.[10]

NetBeans: NetBeans was created by a Prague university student and acquired by Sun Microsystems in 1999, then Oracle in 2010. NetBeans, like Eclipse, is largely recognized as a Java IDE, although it also helps inn the implementation of different programming languages. It is considered to be the ideal IDE for Java 8 and has a sizable user base. Support for Node.js, Maven, PHP, and C/C++ has been improved in the current version. Extra support for Android development is provided through plug-ins. NetBeans is the fifth most popular IDE according to the PYPL Index.[11]

Visual Studio: Visual Studio, which is Microsoft's flagship IDE, isn't usually the first choice for Android development; however, that has improved in recent times, especially with the help of Xamarin. Enterprise for big organizations, Expert for workgroups, Visual Studio for Mac, and the public Community version are all available in Visual Studio. It also includes Visual Studio Code, an open-source code editor. Visual Studio integrates with Xamarin for mobile development.[12]

Android Studio: Google's official Android IDE is Android Studio; it is built on IntelliJ IDEA and includes capabilities tailored to Android. It took the role of the older Eclipse-based Android Development Tools (ADT) acts as the common platform's which is used as an IDE. Android Studio is the prominent IDE in the world, as per PYPL Index's latest current top IDE ranking, with a 9.87 per cent share. It is the default IDE available for Android developers, and it interacts seamlessly with the Google Cloud Platform as a Google service.[13]

Currently, most developers are using android studio for the android application development. That’s the major reason we chose Android Studio for the development of this application

We can develop android applications using Java and Kotlin language.

Java: Initially, java was the Android official language to develop android applications. Java is the most popular language used to develop web applications, windows applications android applications, etc. It is also the most commonly used language. Typically, all of the other android applications in the Play Store and in other Android stores are developed using Java, which is also Google's prominently recommended programming language. In addition, Java has a fantastic online resource for giving details in the event of any scenarios.

Kotlin: JetBrains created Kotlin, an open-source statically typed programming language. It's object-oriented and allows for functional programming. Currently, Kotlin is the official language of android.

To develop this project, we choose Kotlin. Because it’s a new language and it’s so easy to understand compared with java, also Kotlin allows the user to use java knowledge in the existing project. It automatically converts Java to Kotlin.

Kotlin is null safe, which eradicates errors that are frequently occurred by absent types or parameters when not initialized correctly. It’s very important to make sure that you do not use a type as the default null value.[15]

There are several SDKs that are available to integrate chat or messaging modules in android applications. The main popular are Twilio, get App, devto, chatsdk, SendBird etc. [30] Also, there are several SDK’s currently available in the market to develop lop audio and video call. they are pplivo, twillo sinch, SendBird etc. [31]

Among these SDKs we preferred the SendBird SDK framework to integrate chat and call integration. SendBird was developed in 2013. SendBird is sever used to integrate chat, video, and audio calls.

SendBird Calls for voice and video simply work, allowing users to interact via the app although they would in person. Users can have access to all the correct experiences they've come to anticipate when chat, audio, and video are combined. Moreover, it’s a fast-growing framework. After a few years maybe they will provide a lot of features in this project. It will easily help us to integrate with the current system. It is a faster time-to-market is made possible by a simple Chat SDKs, and a completely chat system just on backend. It provides delivery proofs, mobile messaging, moderation tools, and analytics, also we can create advanced chat and communications services.

SendBird Chat: Chat SDK helps the user to create a chat application with a modern messaging experience. we can easily incorporate real-time chat into a client app using SendBird Chat SDK for Android. With the client-side solution, we can quickly set up and customize the chat. SendBird provides trustworthy infrastructure management services for your conversation within the app on the server-side.

SendBird Call: Call SDK helps users to create a call application with modern features such as phone calls and video calls. Also, it allows for the creation of high-quality audio and video conversations for web and mobile apps that include real-time interactivity for increased engagement. SendBird Calls for Android offers one-to-one audio and video calls between users within your SendBird-integrated app. This development kit can help to set up, configure, and implement voice and video calling in your Android app.

SQL and NoSQL database storages are currently available on the market. The SQL programming language is used to communicate with relational databases. My SQL, MS SQL Server, and Oracle Database are famous databases in the market. NoSQL is a non-relational database management system that does not employ SQL. Firebase Realtime Database, MongoDB, and CouchDB are the popular NoSQL database examples. In this project, we used the Firebase Real-time NoSQL database. It is a cloud-based database, that stores data on the base of JSON. Moreover, Firebase is Google’s official NoSQL database.

Firebase Authentication on Android

Firebase authentication is a very simple approach, which is used to monitor, check and authenticate the users of the application. It not only guarantees that consumers have a straightforward login procedure, but it also ensures that developers have a basic flow that is available in practically all applications. Firebase provides the authentication using email and password, and phone authentication. Also, users can use firebase to log in to social media applications sign in such as Google, Facebook, Twitter, and GitHub.[4]   To authenticate users, all we have to do is make sure the user's login authentication credentials are valid and then send them to the Firebase Authentication SDK. These credentials can be email addresses and passwords, phone numbers, or just about any token obtained from web services providers such as Facebook, Google, Twitter, and GitHub. After passing the credentials, Firebase will cross-check all the valid credentials and returns a response that tells you if the authentication is successful or not. By enabling the phone services on the Firebase Authentication, the user will get an SMS message to his mobile phone. Using this message user can sign into that application. An OTP is provided in the SMS and is used to sign in the user into the application. A user's identification is required by most apps. Knowing a user's details will help an app store user data securely in the cloud and give a conditioned and tailored experience across all the user's devices. To authenticate users to this application, Firebase Authentication provides backend services and easy-to-use Inbuilt libraries [4]. Compared with other login methods, phone authentication has less secure. Because people can easily track phone messages. OTP messages can easily be transmitted to another user and Multiple users can log in to the same device.

Firebase Realtime Database: By using the Realtime Database user can store their user details in cloud storage. Also, it helps to control real-time notifications. It is a real-time messaging application. If any data is changed, it will automatically be reflected on the client application.

Firebase Storage: It is a file storage SDK. It can store files, images, audio, videos and other data files. This SDK is used to connect firebase to store the user’s profile pictures.

Chapter 4. System Design and Specifications

This Chapter describes the process of System Design and Specifications for a chat application named "GabChat". This is a real-time platform that allows the user to have a conversation between members. This chapter details the most relevant parts of the application development, decisions taken and algorithms.
This application contains mainly 5 modules,

- Authentication
- Chat
- Call
- Screen share
- Screenshot detection

Authentication: This GabChat will be authenticated with a firebase phone validator and store the details in the database. Then the user gets into the project.
Chat: The user can chat with other members. This message will be encrypted, once it delivers on another end then it will be decrypted
Call: Users can make an audio, or video call through this application.
Screen share: In between the video call user can share the screen with other end people.
Screenshot detection: in between the chat and video call, it will send a notification to another end person.

4.1 Data Flow Diagram

Level 0

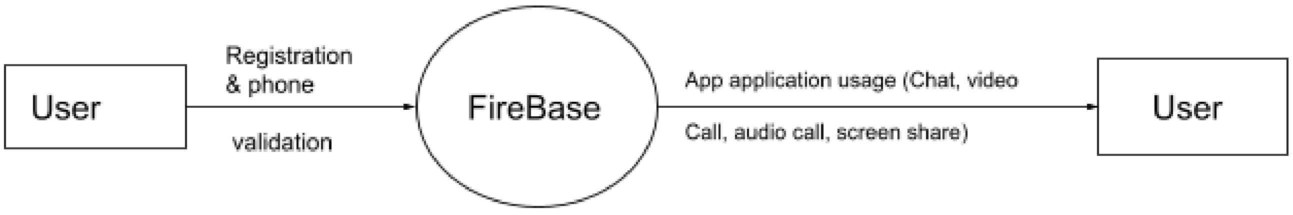

User
Registration & phone validation
FireBase
App application usage (Chat, video Call, audio call, screen share)
User

Figure 4.1: level 0 data flow charts

Level 1

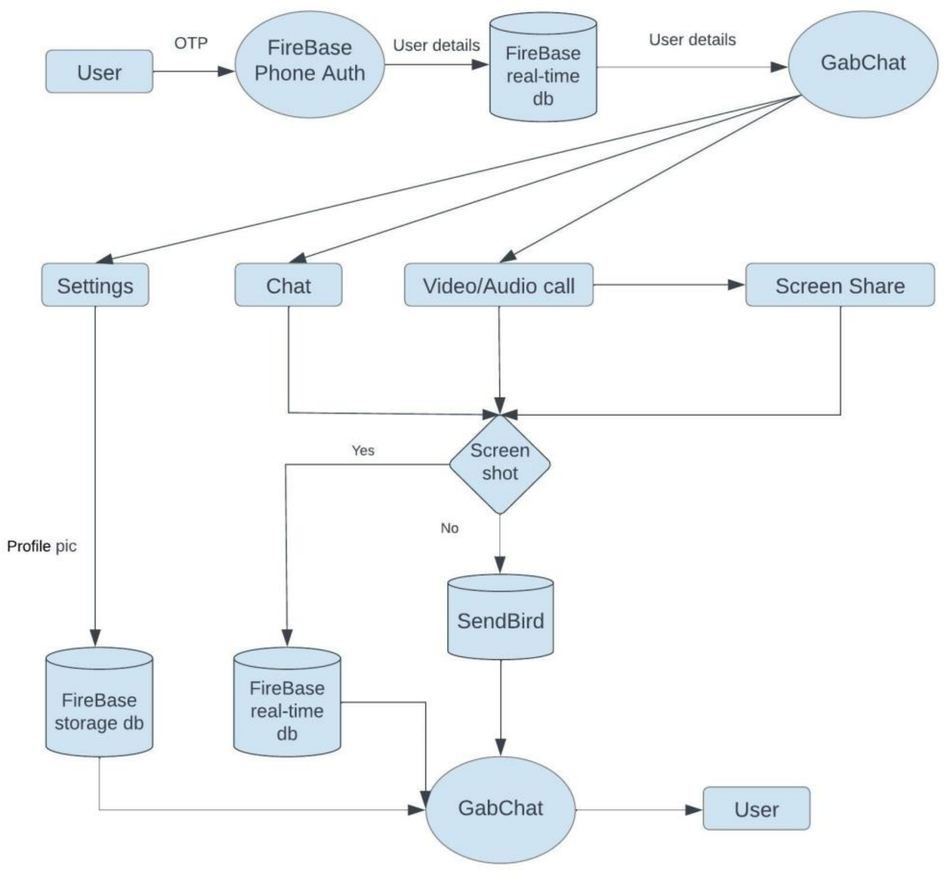


Figure 4.1: level 1 data flow charts

Once the user opens the application it will show a login page with the phone number. Users need to provide a phone number to log in to this system. First, will generate the OTP through firebase authentication, and then firebase will verify the message. Firebase phone authentication was authentication and checking the user details present in the firebase database. If the user is already present, it will get the user details and shows the details on the registration page then the user needs to provide the password to log in to this system. If the password is correct then only the user can get into the system, if the user is not available, the user needs to provide the user details and password. Then these details will be stored in the database and the user can get into the application GabChat.

Once the user gets into the GabChat first, it will authenticate the user with SendBird, Then the system will create a firebase token for this user, and this token will be stored in the SendBird server. This token helps to redirect the call to the correct user. Using this application users can communicate with others and make video/ audio phone calls.

When the user chats with a person, then first it will encrypt the message in a chat module. Then the system will check whether the user will take any screenshot of the chat. If the user took any screenshots, it would send a message to firebase, and firebase generates notifications, it was implemented using firebase's real-time application. So once any changes happen, data changes will automatically reflect on the system. When the data get into GabChat it will call a notification generation service. and another user will get the notification. If the user didn’t take any screenshots, then it will send the message to the SendBird server, it will redirect the message to the actual user and redirect to GabChat and the user can see the message.

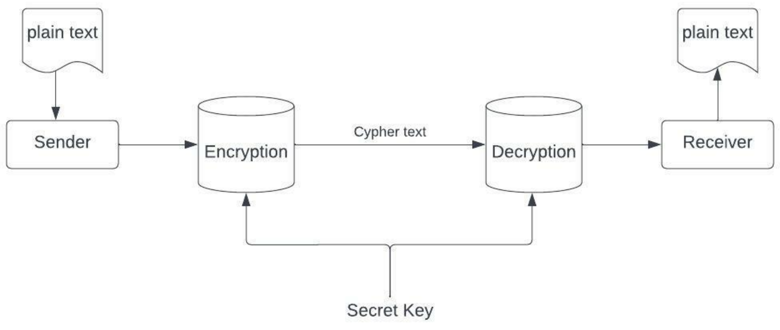


Figure 4.2: data flow chart of AES algorithm

This chat module contains encryptions. This application has used the AES encryption method. Which encrypts the message before sending and decrypts the message once it reaches the user. It contains a common secret key. This key was used to encrypt and decrypt the messages. First using this key system will encrypt the message and create cypher text. And this cypher text sends into SendBird. Once, this message was received the receiver side chat module will decrypt this message using the same secret key.

When the user makes any video/audio call with a person. Then the system will connect with the SendBird server, SendBird servers already store the firebase token of that user. Using this token call will be redirected to the actual user. When a call is received to a user, the initial call gets to GabChat. GabChat contains a firebase token. The system will check the firebase token which passes from the SendBird and the current user device firebase token. If both are the same, then create a notification and the user can hear a ringtone. Users can accept or decline this phone call. Also, while making a video call the user can share the screen, at this time application will check whether the user took any screenshots of the call. If the user took any screenshots, then the other user will get a notification.

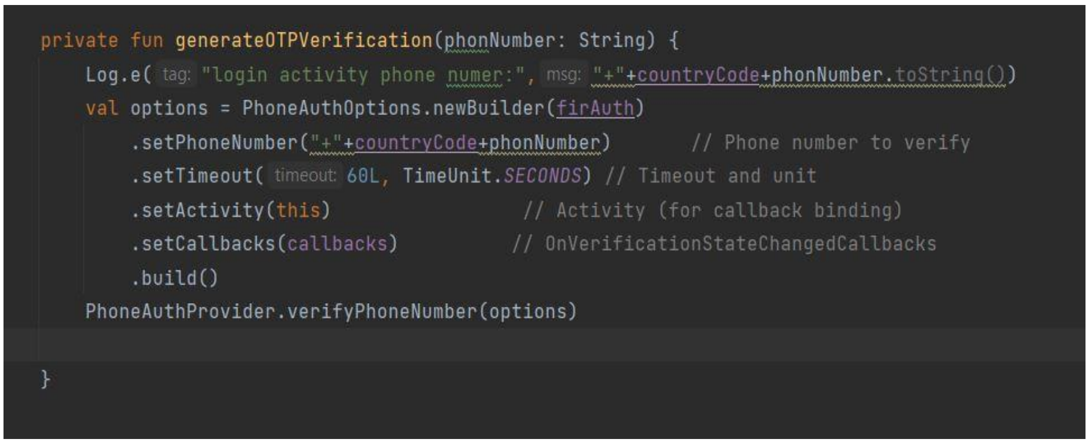
Also, users can edit user details using the setting menu option. If the user gets into the setting option, he/she can change the user's details, such as profile pic, and nickname. If the user wants to change the profile pic, he/she can select the pic from the device gallery and the selected picture can be set as a profile picture. When the user selects the picture from the gallery, first it will store the file into firebase and the user can see that picture

Chapter 5. Implementation

This chapter will discuss the implementation of the GabChat. Initially, we had to create a SendBird account, and generate a SendBird Id. Using this SendBird Id this GabChat will connect to the SendBird server. Then, we had to create a Firebase account and need to connect with the android studio project, then only we can use firebase features in this system. Then created a new project in firebase using this project package name. and download google service credentials from the firebase and paste it into the android studio, then only android studio communicates with firebase. Then connect the android studio with firebase login Gmail Id and link firebase features, such as Firebase Authentication, Firebase cloud storage, and Firebase real-time storage.

Authentication

Over here it authenticates the phone number. Using this authentication it will get into the system.



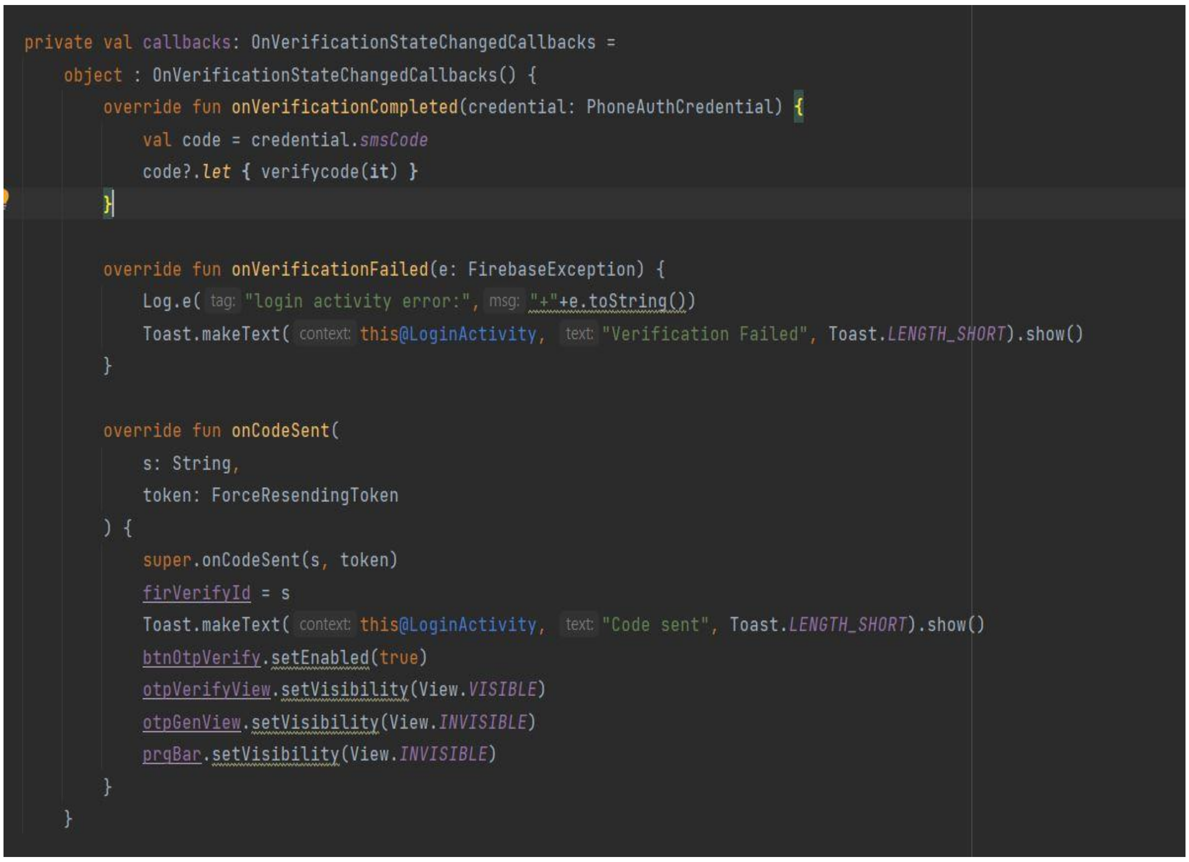
```
private fun generateOTPVerification(phonNumber: String) {
    Log.e( tag: "login activity phone numer:", msg: "+"+countryCode+phonNumber.toString())
    val options = PhoneAuthOptions.newBuilder(firAuth)
        .setPhoneNumber("+"+countryCode+phonNumber)       // Phone number to verify
        .setTimeout( timeout: 60L, TimeUnit.SECONDS) // Timeout and unit
        .setActivity(this)                 // Activity (for callback binding)
        .setCallbacks(callbacks)          // OnVerificationStateChangedCallbacks
        .build()
    PhoneAuthProvider.verifyPhoneNumber(options)

}
```

Figure 5.1: initialize the OTP object

Over here it initializes the user's phone number and all the values for the OTP generation were stored on the object's options. Also need to provide the instance of callback function "OnVerificationStateChangedCallbacks". Once firebase verifies the phone number then automatically it redirects to this callback function. Then it passes the options to the method "PhoneAuthProvider.verifyPhoneNumber" to request firebase to verify these details.

“OnVerificationStateChangedCallbacks” is a callback instance. Once firebase verifies the phone number it will automatically redirect to this instance. It contains implementations of firebase callback methods. Mainly it contains three implementations.

- onVerificationCompleted
- onVerificationFailed
- onCodeSent



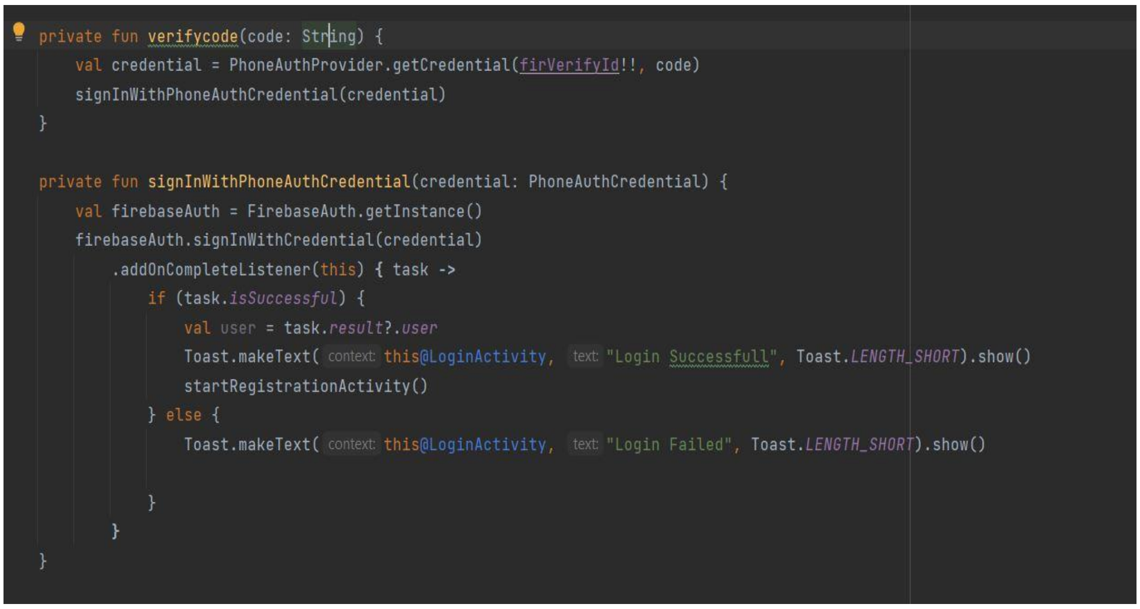
```kotlin
private val callbacks: OnVerificationStateChangedCallbacks =
    object : OnVerificationStateChangedCallbacks() {
        override fun onVerificationCompleted(credential: PhoneAuthCredential) {
            val code = credential.smsCode
            code?.let { verifycode(it) }
        }

        override fun onVerificationFailed(e: FirebaseException) {
            Log.e( tag: "login activity error:", msg: "+"+e.toString())
            Toast.makeText( context: this@LoginActivity,  text: "Verification Failed", Toast.LENGTH_SHORT).show()
        }

        override fun onCodeSent(
            s: String,
            token: ForceResendingToken
        ) {
            super.onCodeSent(s, token)
            firVerifyId = s
            Toast.makeText( context: this@LoginActivity,  text: "Code sent", Toast.LENGTH_SHORT).show()
            btnOtpVerify.setEnabled(true)
            otpVerifyView.setVisibility(View.VISIBLE)
            otpGenView.setVisibility(View.INVISIBLE)
            prgBar.setVisibility(View.INVISIBLE)
        }
    }
```

Figure 5.2: Generate OTP callback

onVerificationCompleted(credential: PhoneAuthCredential): Once the phone number verification was completed call was redirected to this method. Then it uses PhoneAuthCredential objects that pass to the sign-in method.

onVerificationFailed(e: FirebaseException): it will call when the verification is invalid, or the given phone number is invalid.

override fun onCodeSent(s: String, token: ForceResendingToken): it will call once firebase send OTP to user. Then the user will get the OTP message on phone. Using this OTP user can verify the login.



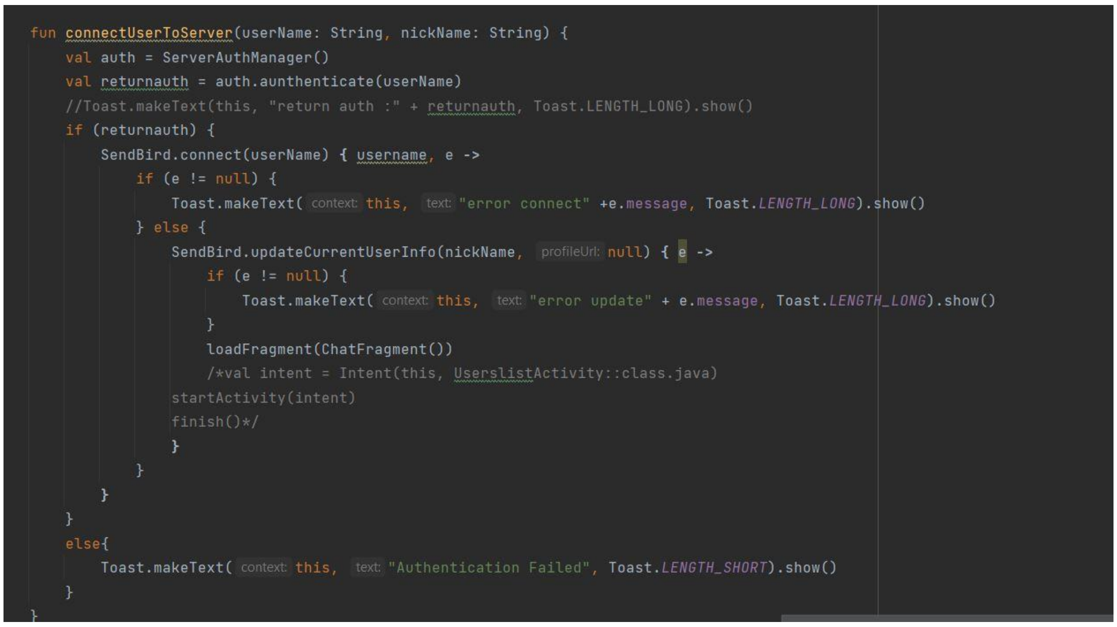
```
private fun verifycode(code: String) {
    val credential = PhoneAuthProvider.getCredential(firVerifyId!!, code)
    signInWithPhoneAuthCredential(credential)
}

private fun signInWithPhoneAuthCredential(credential: PhoneAuthCredential) {
    val firebaseAuth = FirebaseAuth.getInstance()
    firebaseAuth.signInWithCredential(credential)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                val user = task.result?.user
                Toast.makeText( context: this@LoginActivity,  text: "Login Successfull", Toast.LENGTH_SHORT).show()
                startRegistrationActivity()
            } else {
                Toast.makeText( context: this@LoginActivity,  text: "Login Failed", Toast.LENGTH_SHORT).show()

            }
        }
}
```

Figure 5.3: verify OTP

Before verifying the OTP, the system needs to create the object for PhoneAuthCredential. This object is created using PhoneAuthProvider.getCredential and OTP code and verification id. using these credentials all the firebase method firebaseAuth.signInWithCredential(credential). If the verification is a success It will redirect to the registration page, otherwise shows an error message.

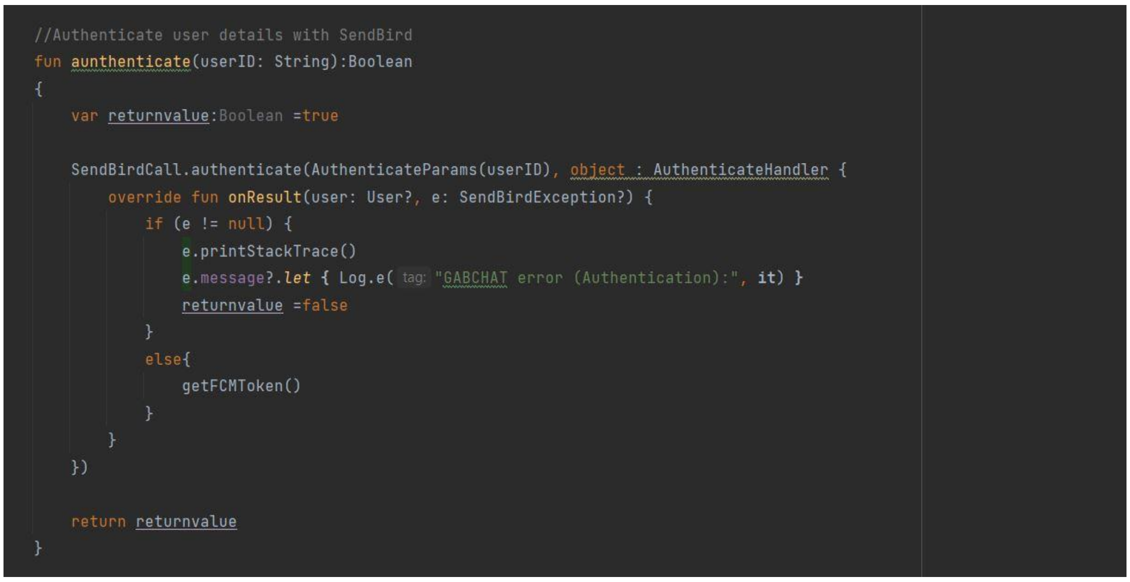
Connect to the SendBird server

After the firebase authentication, it will authenticate with SendBird, to verify with the user. For that, it will call "connectUserToServer" function with the parameter of the user-defined login username. in this function first, it needs to authenticate the user with SendBird. For that call authenticate function with the parameter users name(fig5).



```
fun connectUserToServer(userName: String, nickName: String) {
    val auth = ServerAuthManager()
    val returnauth = auth.aunthenticate(userName)
    //Toast.makeText(this, "return auth :" + returnauth, Toast.LENGTH_LONG).show()
    if (returnauth) {
        SendBird.connect(userName) { username, e ->
            if (e != null) {
                Toast.makeText( context: this,  text: "error connect" +e.message, Toast.LENGTH_LONG).show()
            } else {
                SendBird.updateCurrentUserInfo(nickName,  profileUrl: null) { e ->
                    if (e != null) {
                        Toast.makeText( context: this,  text: "error update" + e.message, Toast.LENGTH_LONG).show()
                    }
                    loadFragment(ChatFragment())
                    /*val intent = Intent(this, UserslistActivity::class.java)
                startActivity(intent)
                finish()*/
                }
            }
        }
    }
    else{
        Toast.makeText( context: this,  text: "Authentication Failed", Toast.LENGTH_SHORT).show()
    }
}
```

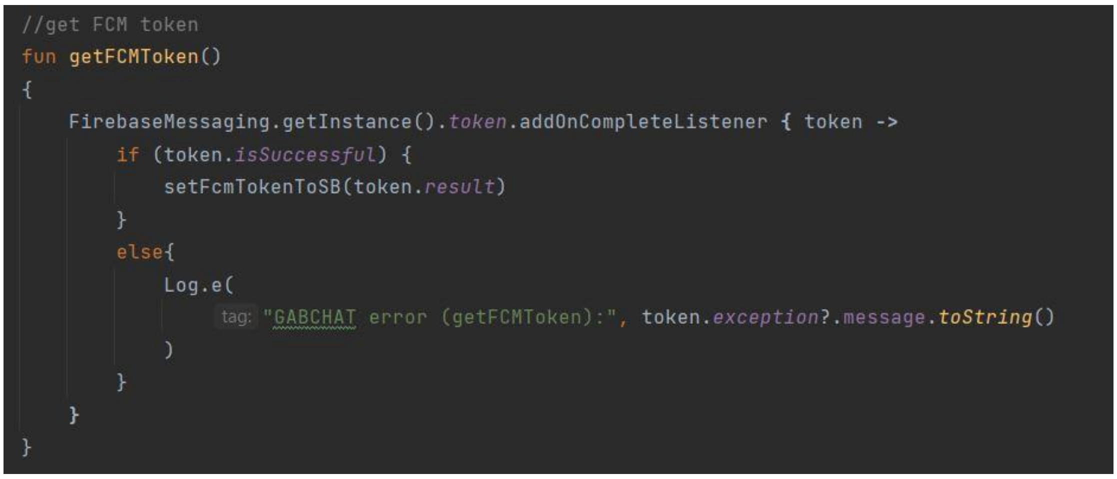
Figure 5.4: Connect user to SendBird



```
//Authenticate user details with SendBird
fun aunthenticate(userID: String):Boolean
{
    var returnvalue:Boolean =true

    SendBirdCall.authenticate(AuthenticateParams(userID), object : AuthenticateHandler {
        override fun onResult(user: User?, e: SendBirdException?) {
            if (e != null) {
                e.printStackTrace()
                e.message?.let { Log.e( tag: "GABCHAT error (Authentication):", it) }
                returnvalue =false
            }
            else{
                getFCMToken()
            }
        }
    })

    return returnvalue
}
```

Figure 5.5: Authenticate the user to SendBird call

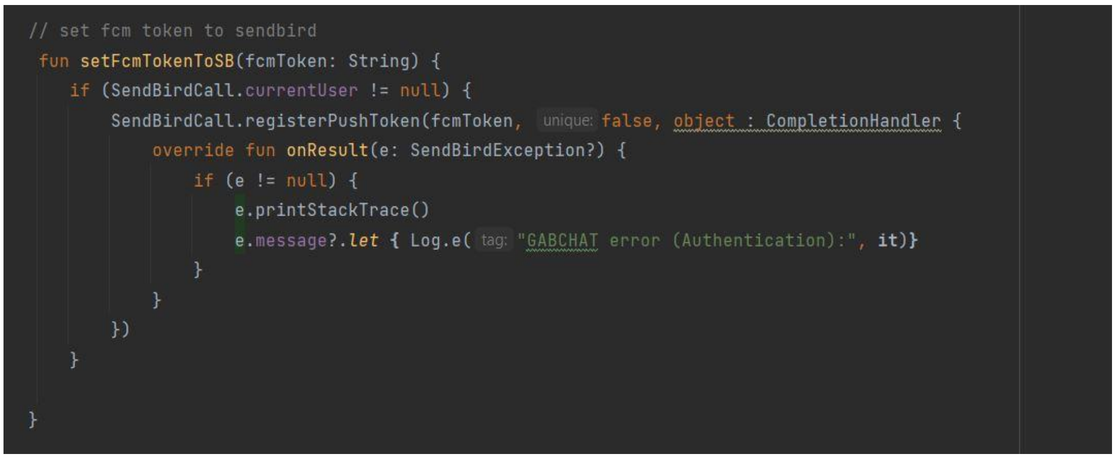
Figure 5 will authenticate the user with the SendBird call. Once this authentication was a success then only the user will get the call from others. If the authentication was successful then it will create a firebase instance for this particular user (fig 6), for that it will call another method getFCMToken(). Over here it will initialize a particular token for this user, and it will pass this token to setFcmTokenToSB method.



```
//get FCM token
fun getFCMToken()
{
    FirebaseMessaging.getInstance().token.addOnCompleteListener { token ->
        if (token.isSuccessful) {
            setFcmTokenToSB(token.result)
        }
        else{
            Log.e(
                tag: "GABCHAT error (getFCMToken):", token.exception?.message.toString()
            )
        }
    }
}
```

Figure 5.6: Generate Firebase token for the user.

setFcmTokenToSB method will store this firebase instance in SendBird server to identify the unique user.



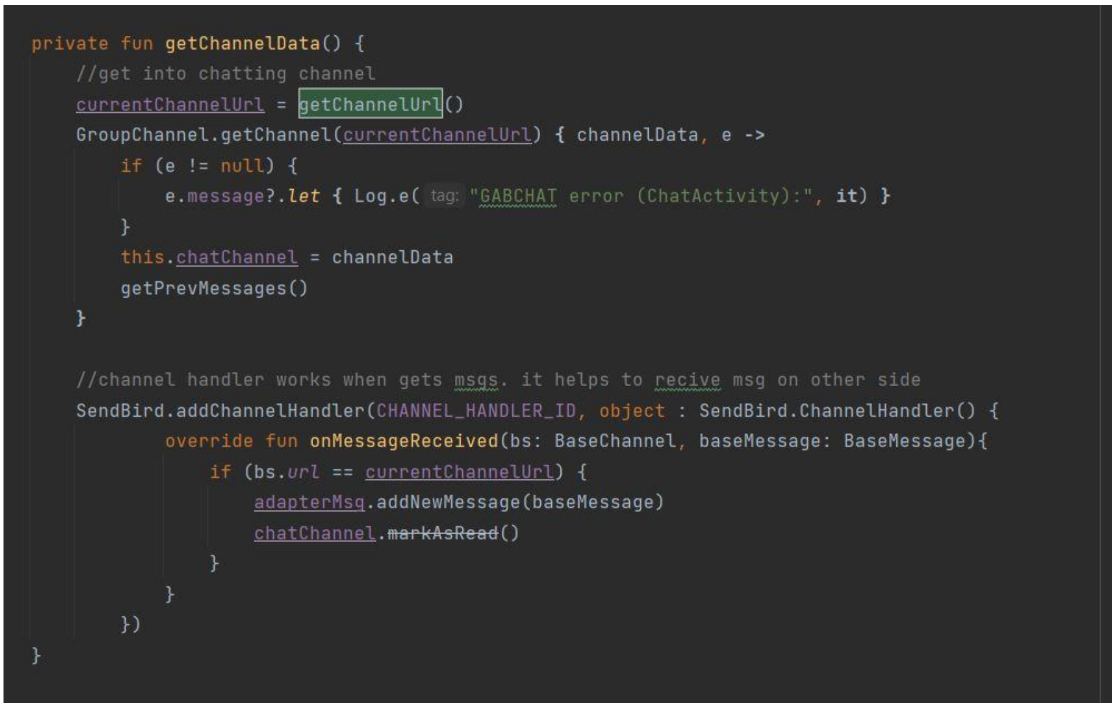
```
// set fcm token to sendbird
 fun setFcmTokenToSB(fcmToken: String) {
    if (SendBirdCall.currentUser != null) {
        SendBirdCall.registerPushToken(fcmToken, unique: false, object : CompletionHandler {
            override fun onResult(e: SendBirdException?) {
                if (e != null) {
                    e.printStackTrace()
                    e.message?.let { Log.e( tag: "GABCHAT error (Authentication):", it)}
                }
            }
        })
    }

}
```

Figure 5.7: Set Firebase token to SendBird.

Once the firebase token is stored in the SendBird server, then only the user will get the phone calls from another user. Because this firebase unique instance will help to identify the unique user. Once it stores the data in the SendBird server, then the system will connect the current user with SendBird using the method "sendbird.connect" with the parameter username (fig 4).

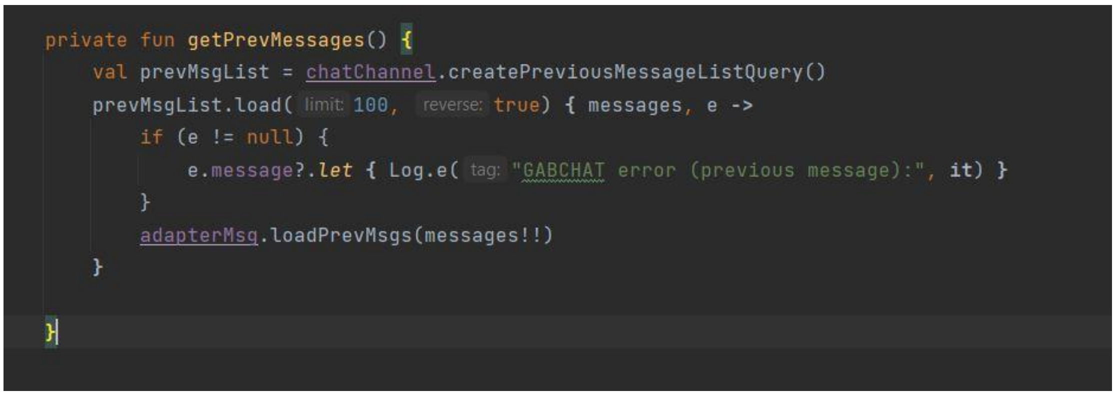
Chat

It is a user chat details. To get the chat data we call getChannelData method. In this method, initially, the system will create a user chat URL with the callee name using getChannelUrl method. Then using GroupChannel.getChannel method system will get the channel data. This data will be used to get the previous message.



```kotlin
private fun getChannelData() {
    //get into chatting channel
    currentChannelUrl = getChannelUrl()
    GroupChannel.getChannel(currentChannelUrl) { channelData, e ->
        if (e != null) {
            e.message?.let { Log.e( tag: "GABCHAT error (ChatActivity):", it) }
        }
        this.chatChannel = channelData
        getPrevMessages()
    }

    //channel handler works when gets msgs. it helps to recive msg on other side
    SendBird.addChannelHandler(CHANNEL_HANDLER_ID, object : SendBird.ChannelHandler() {
            override fun onMessageReceived(bs: BaseChannel, baseMessage: BaseMessage){
                if (bs.url == currentChannelUrl) {
                    adapterMsg.addNewMessage(baseMessage)
                    chatChannel.~~markAsRead~~()
                }
            }
        })
}
```

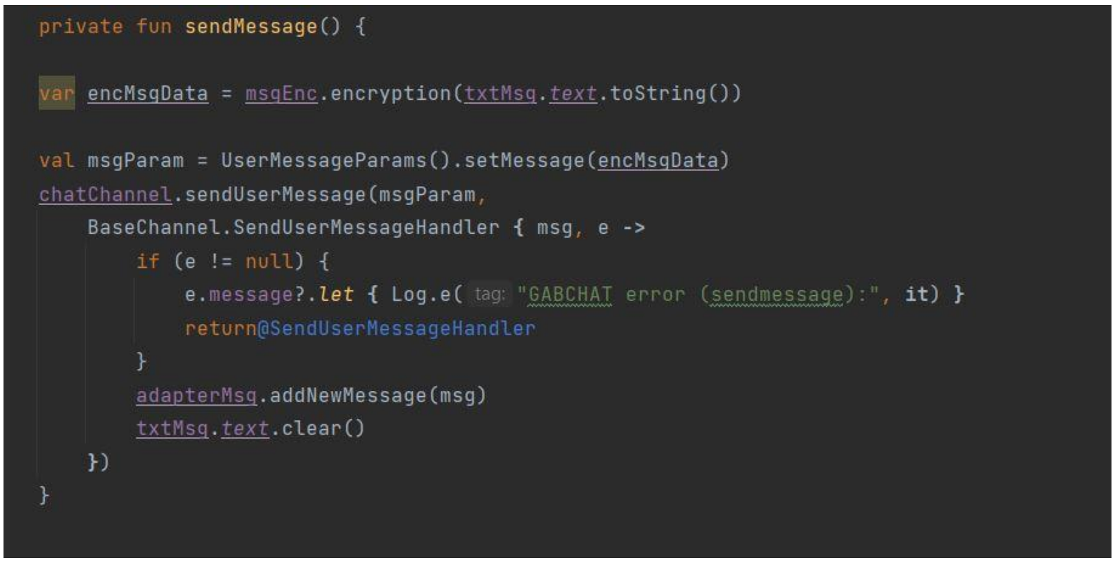
Figure 5.7: get channel data



```kotlin
private fun getPrevMessages() {
    val prevMsgList = chatChannel.createPreviousMessageListQuery()
    prevMsgList.load( limit: 100,  reverse: true) { messages, e ->
        if (e != null) {
            e.message?.let { Log.e( tag: "GABCHAT error (previous message):", it) }
        }
        adapterMsg.loadPrevMsgs(messages!!)
    }

}
```

Figure 5.8: get the previous message

To get the previous message we can use getPrevMessages function. On this function first, it will get the chat data from the getChannelData function. Using this data, it queries the messages and loads them into the recycler view using the recycler adaptor.



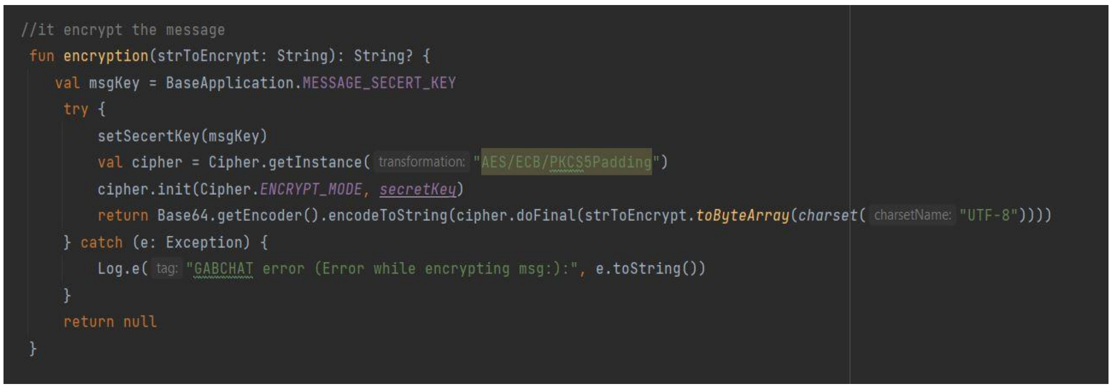
```kotlin
private fun sendMessage() {

var encMsgData = msgEnc.encryption(txtMsg.text.toString())

val msgParam = UserMessageParams().setMessage(encMsgData)
chatChannel.sendUserMessage(msgParam,
    BaseChannel.SendUserMessageHandler { msg, e ->
        if (e != null) {
            e.message?.let { Log.e( tag: "GABCHAT error (sendmessage):", it) }
            return@SendUserMessageHandler
        }
        adapterMsg.addNewMessage(msg)
        txtMsg.text.clear()
    })
}
```

Figure 5.9: Send Message.

The system will encrypt the message before sending the message to the server. Then using this encrypted data call UserMessage.Params function. This is a parameter for sendUserMessage method. Once this message will send to SendBird, then the server will give a response with SendUserMessageHandler. This response message will add to the adaptor.

Encryption

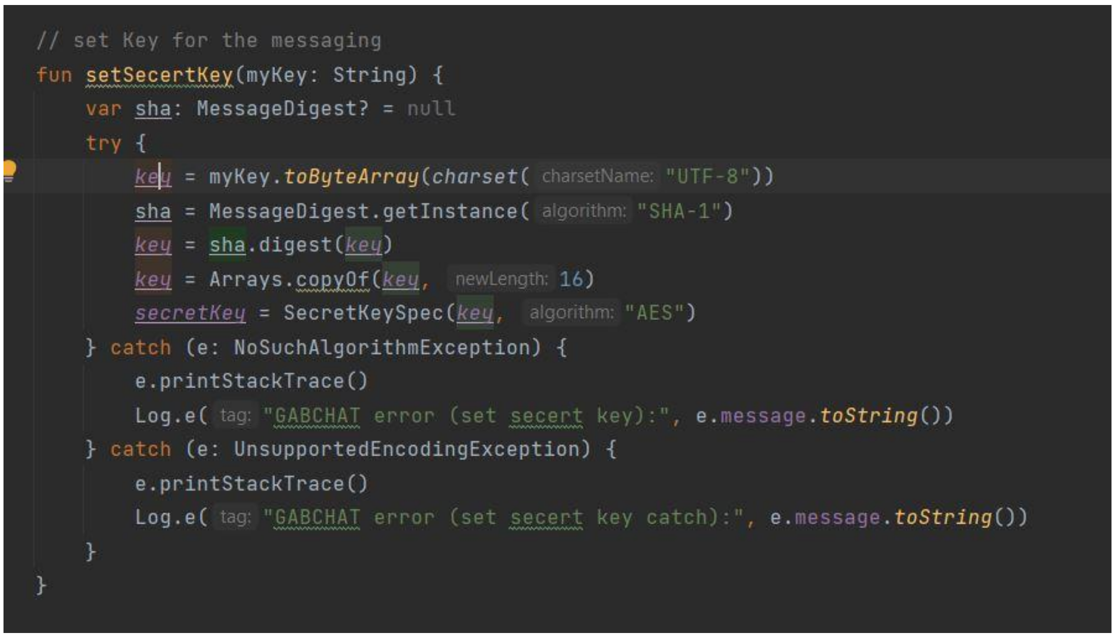
Encryption is a major part of this project. When a user sends a message to another person, first it will encrypt the message using the AES algorithm, and then the system will decrypt the message.



```kotlin
//it encrypt the message
 fun encryption(strToEncrypt: String): String? {
    val msgKey = BaseApplication.MESSAGE_SECERT_KEY
     try {
         setSecertKey(msgKey)
         val cipher = Cipher.getInstance( transformation: "AES/ECB/PKCS5Padding")
         cipher.init(Cipher.ENCRYPT_MODE, secretKey)
         return Base64.getEncoder().encodeToString(cipher.doFinal(strToEncrypt.toByteArray(charset( charsetName: "UTF-8"))))
     } catch (e: Exception) {
         Log.e( tag: "GABCHAT error (Error while encrypting msg:):", e.toString())
     }
     return null
 }
```

Figure 5.10: Encrypt Message.

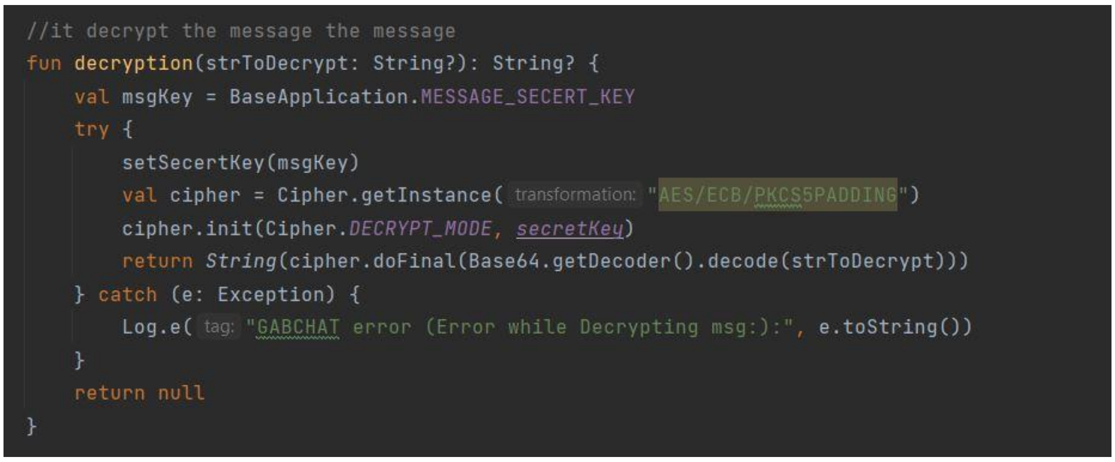
The key was kept in the base application manager of this project. To encrypt the message first it will call the encryption function with the message string. Then set the secret key for the encryption using the function setSecertKey (fig:11). In this function, it will convert the user key into the secret key. This same key is used to decrypt the message.



```
// set Key for the messaging
fun setSecertKey(myKey: String) {
    var sha: MessageDigest? = null
    try {
        key = myKey.toByteArray(charset( charsetName: "UTF-8"))
        sha = MessageDigest.getInstance( algorithm: "SHA-1")
        key = sha.digest(key)
        key = Arrays.copyOf(key,  newLength: 16)
        secretKey = SecretKeySpec(key,  algorithm: "AES")
    } catch (e: NoSuchAlgorithmException) {
        e.printStackTrace()
        Log.e( tag: "GABCHAT error (set secert key):", e.message.toString())
    } catch (e: UnsupportedEncodingException) {
        e.printStackTrace()
        Log.e( tag: "GABCHAT error (set secert key catch):", e.message.toString())
    }
}
```

Figure 5.11: Set Secret Key.

Using this key, the encryption method will create a cypher text and it will send back to the user.



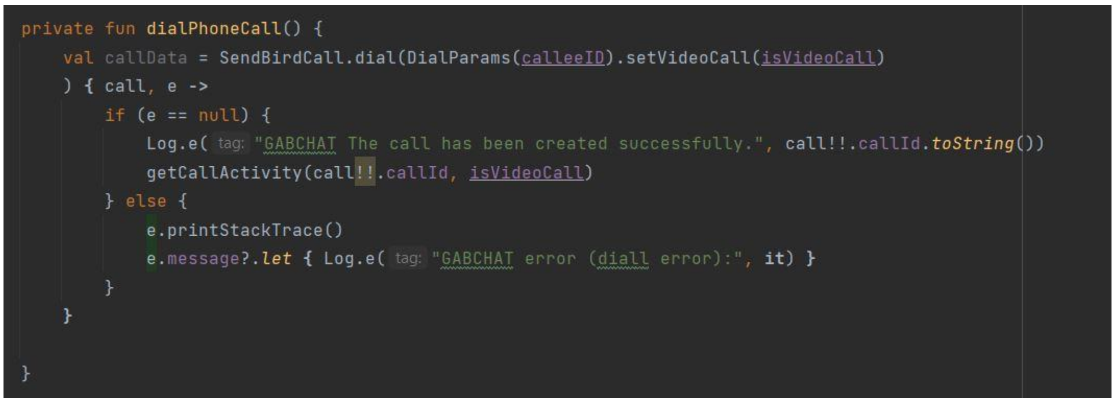
```
//it decrypt the message the message
fun decryption(strToDecrypt: String?): String? {
    val msgKey = BaseApplication.MESSAGE_SECERT_KEY
    try {
        setSecertKey(msgKey)
        val cipher = Cipher.getInstance( transformation: "AES/ECB/PKCS5PADDING")
        cipher.init(Cipher.DECRYPT_MODE, secretKey)
        return String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt)))
    } catch (e: Exception) {
        Log.e( tag: "GABCHAT error (Error while Decrypting msg:):", e.toString())
    }
    return null
}
```

Figure 5.12: Decrypt message.

When the user gets the cypher text from the server, then it will call the decryption method to decrypt the message. First, it will set the secret key, then it will create a cypher instance for the AES algorithm and using this instance and cypher text it will decrypt the message.

Call

When the user clicks on the audio call or video call button, it will call the dialPhoneCall method. In this method, first, create a parameter to call to the selected callee. This parameter contains calleeId and the Boolean value is isVideoCall. If it is a video call, then the isVideoCall will be true else it will be false. Then these parameter values will be passed to SendBirdCall.dial method. Then the call will hit on the server. The server will find the user and using the fcm token id it will redirect the call to the selected user.



```
private fun dialPhoneCall() {
    val callData = SendBirdCall.dial(DialParams(calleeID).setVideoCall(isVideoCall)
    ) { call, e ->
        if (e == null) {
            Log.e( tag: "GABCHAT The call has been created successfully.", call!!.callId.toString())
            getCallActivity(call!!.callId, isVideoCall)
        } else {
            e.printStackTrace()
            e.message?.let { Log.e( tag: "GABCHAT error (diall error):", it) }
        }
    }

}
```

Figure 5.13: Dial phone call

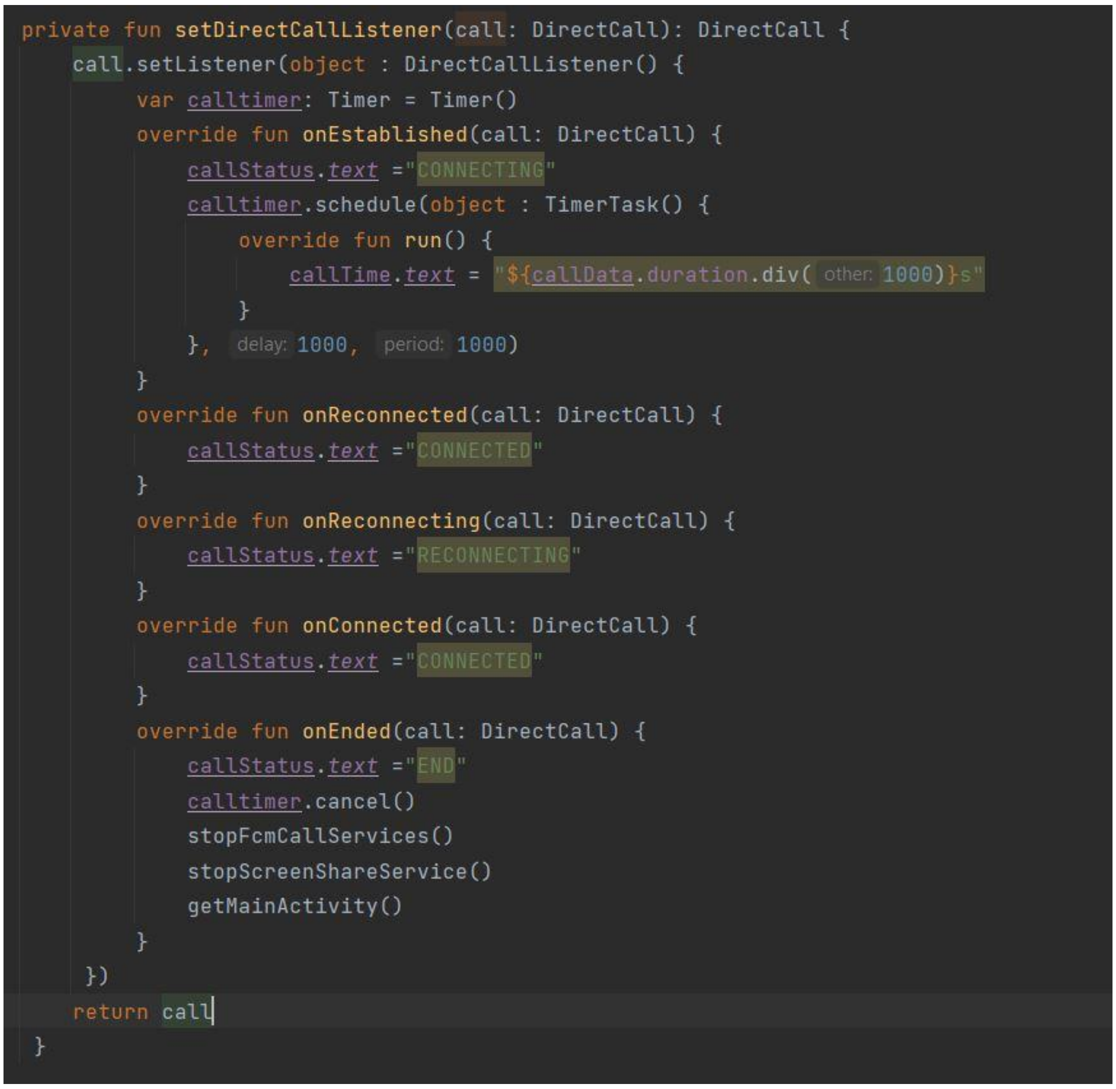
Once the call is established, then the system will call the method setDirectCallListener with calldata parameter. In this function, the user will get control over the call. This function contains a call listener. This listener will contain a few functions.

onEstablished: Initially, call status will be calling. When the user will accept the call will be established and it will redirect to here and the status will be changed to "connecting" and the timer will start.

onReconnected: When the connection has some error and if it will reconnect then the function will get here.

onConnected: Once the other end-user accepts the call then the data will get into this function.

onEnded: When the call ends, then this function will call, and the call status will be changed to "End".

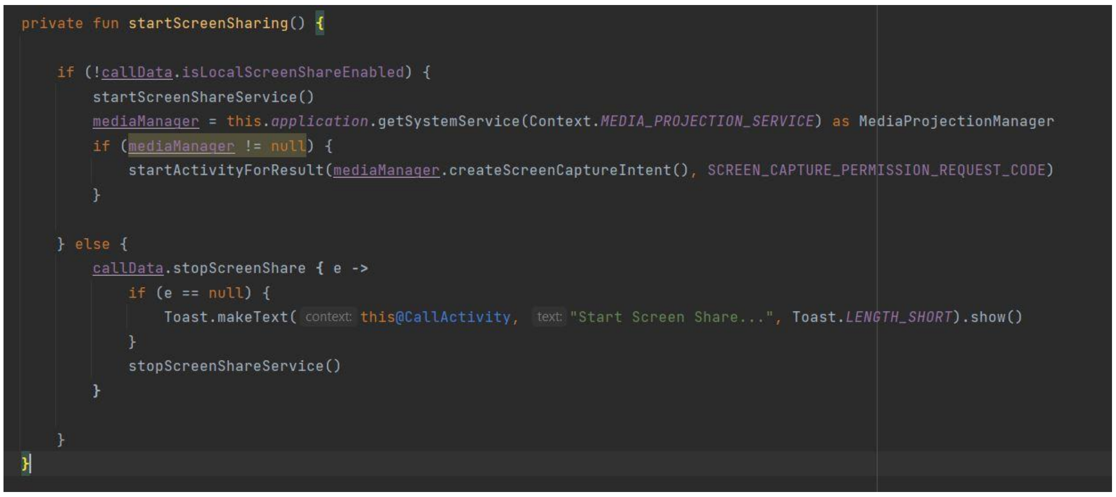


```
private fun setDirectCallListener(call: DirectCall): DirectCall {
    call.setListener(object : DirectCallListener() {
        var calltimer: Timer = Timer()
        override fun onEstablished(call: DirectCall) {
            callStatus.text ="CONNECTING"
            calltimer.schedule(object : TimerTask() {
                override fun run() {
                    callTime.text = "${callData.duration.div( other: 1000)}s"
                }
            }, delay: 1000, period: 1000)
        }
        override fun onReconnected(call: DirectCall) {
            callStatus.text ="CONNECTED"
        }
        override fun onReconnecting(call: DirectCall) {
            callStatus.text ="RECONNECTING"
        }
        override fun onConnected(call: DirectCall) {
            callStatus.text ="CONNECTED"
        }
        override fun onEnded(call: DirectCall) {
            callStatus.text ="END"
            calltimer.cancel()
            stopFcmCallServices()
            stopScreenShareService()
            getMainActivity()
        }
     })
    return call
 }
```

Figure 5.14: Call Listener

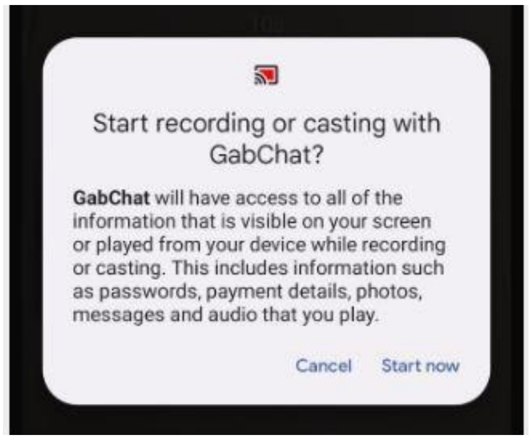
Screen share

When the user makes a phone call on time user can share the screen with others. We need to add further permissions to the original manifest. Screen sharing is possible using the FOREGROUND_SERVICE permission. Android uses a media projection manager to share the screen. To share the screen first it asks permission for MediaProjectionManager(fig 16).



```kotlin
private fun startScreenSharing() {

    if (!callData.isLocalScreenShareEnabled) {
        startScreenShareService()
        mediaManager = this.application.getSystemService(Context.MEDIA_PROJECTION_SERVICE) as MediaProjectionManager
        if (mediaManager != null) {
            startActivityForResult(mediaManager.createScreenCaptureIntent(), SCREEN_CAPTURE_PERMISSION_REQUEST_CODE)
        }

    } else {
        callData.stopScreenShare { e ->
            if (e == null) {
                Toast.makeText( context: this@CallActivity,  text: "Start Screen Share...", Toast.LENGTH_SHORT).show()
            }
            stopScreenShareService()
        }

    }
}
```

Figure 5.15: Start screen share permission request

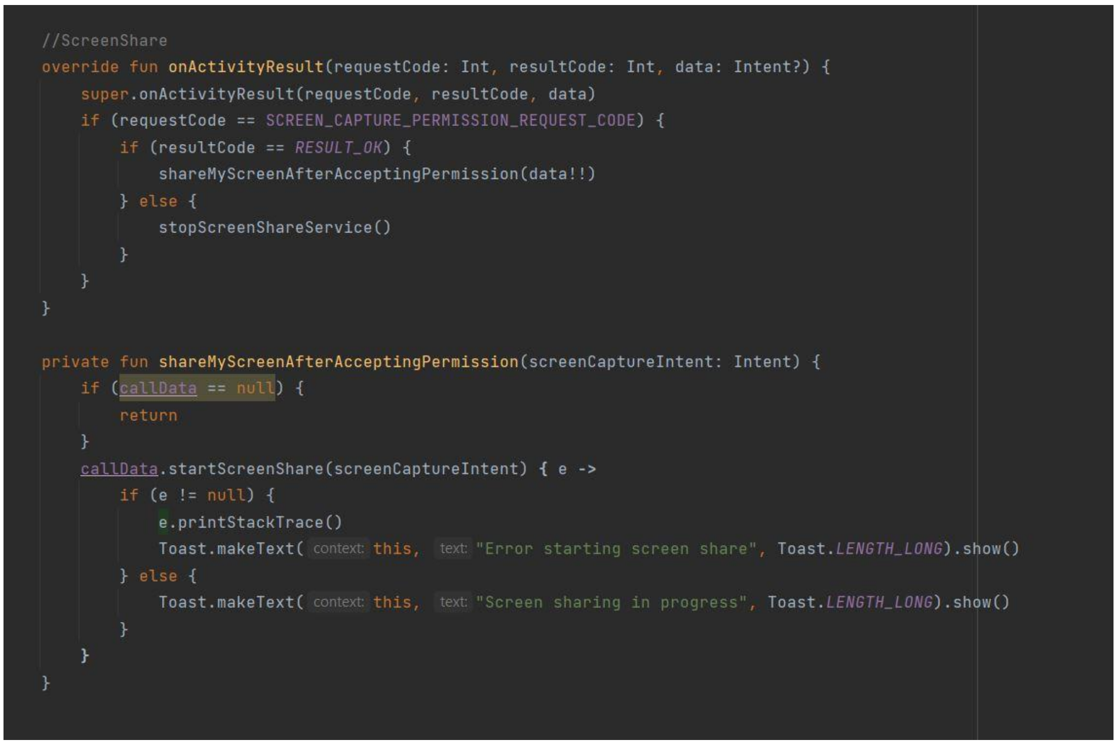


Start recording or casting with GabChat?

GabChat will have access to all of the information that is visible on your screen or played from your device while recording or casting. This includes information such as passwords, payment details, photos, messages and audio that you play.

Figure 5.16: Start screen share permission

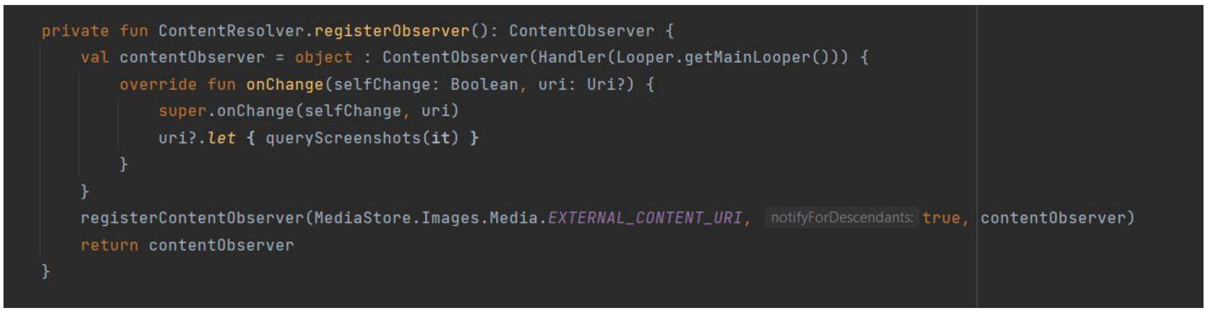
Once the user accepts the permission it will call onActivityResult method. It will call shareMyScreenAfterAcceptingPermission method, it will call SendBird method startScreenShare and it share the screen.



```
//ScreenShare
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == SCREEN_CAPTURE_PERMISSION_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            shareMyScreenAfterAcceptingPermission(data!!)
        } else {
            stopScreenShareService()
        }
    }
}

private fun shareMyScreenAfterAcceptingPermission(screenCaptureIntent: Intent) {
    if (callData == null) {
        return
    }
    callData.startScreenShare(screenCaptureIntent) { e ->
        if (e != null) {
            e.printStackTrace()
            Toast.makeText( context: this,  text: "Error starting screen share", Toast.LENGTH_LONG).show()
        } else {
            Toast.makeText( context: this,  text: "Screen sharing in progress", Toast.LENGTH_LONG).show()
        }
    }
}
```

Figure 5.17: Start screen share

Screenshot detection

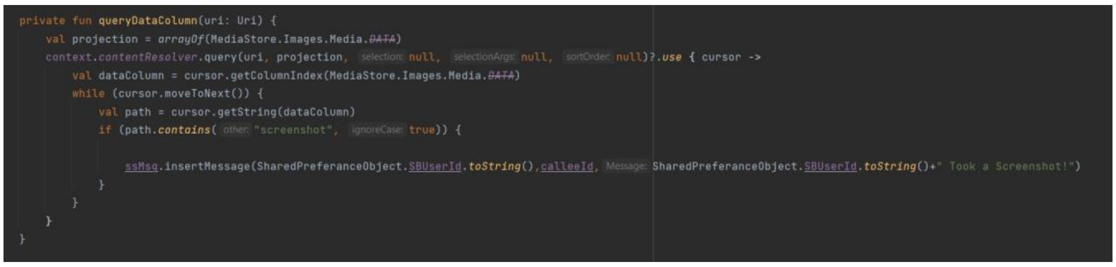


```
private fun ContentResolver.registerObserver(): ContentObserver {
    val contentObserver = object : ContentObserver(Handler(Looper.getMainLooper())) {
        override fun onChange(selfChange: Boolean, uri: Uri?) {
            super.onChange(selfChange, uri)
            uri?.let { queryScreenshots(it) }
        }
    }
    registerContentObserver(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,  notifyForDescendants: true, contentObserver)
    return contentObserver
}
```

Figure 5.18: Content resolver

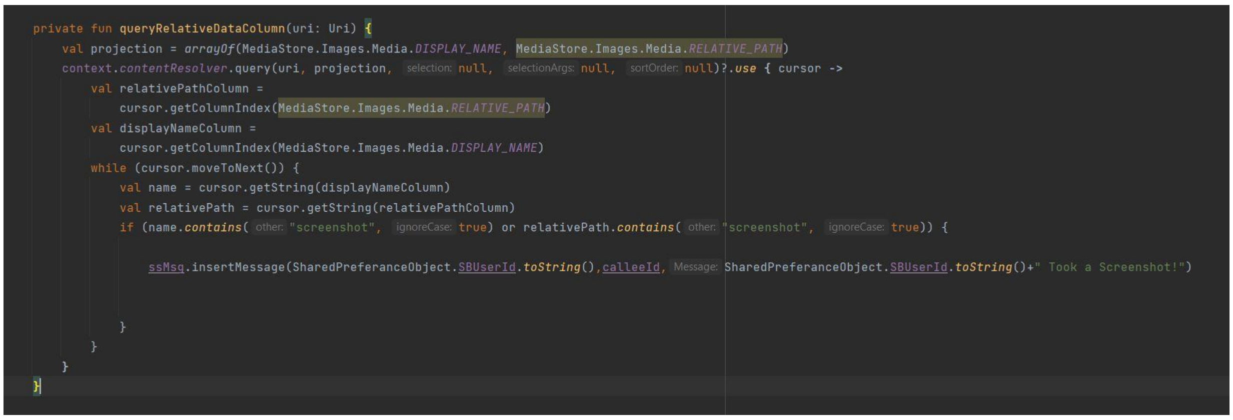
This is the main module of this project. When the user takes screenshots of a chat or video call another end user will get the notification. For that, we are using ContentObserver to find the changes in images on the user's phone, and it has a handler parameter with the onChange function(fig5.18). if the user’s device has any changes, then this function will trigger. After that register this contentObserver using the registerContentObserver method. So now any changes to the system will be effective in our application. When any file has been changed, we will get the data on

URI, we can get the data from MediaStore.Images. but the higher version of API 29 has relative_path and display_name, and other devices have data. So we created two functions queryDataColumn(fig 5.19) for below API 29 and queryRelativeDataColumn(fig 5.20) for higher than AI 29.



```
private fun queryDataColumn(uri: Uri) {
    val projection = arrayOf(MediaStore.Images.Media.DATA)
    context.contentResolver.query(uri, projection,  selection: null,  selectionArgs: null,  sortOrder: null)?.use { cursor ->
        val dataColumn = cursor.getColumnIndex(MediaStore.Images.Media.DATA)
        while (cursor.moveToNext()) {
            val path = cursor.getString(dataColumn)
            if (path.contains( other: "screenshot",  ignoreCase: true)) {

                ssMsg.insertMessage(SharedPreferanceObject.SBUserId.toString(),calleeId,  Message: SharedPreferanceObject.SBUserId.toString()+" Took a Screenshot!")
            }
        }
    }
}
```

Figure 5.19: query data column



```
private fun queryRelativeDataColumn(uri: Uri) {
    val projection = arrayOf(MediaStore.Images.Media.DISPLAY_NAME, MediaStore.Images.Media.RELATIVE_PATH)
    context.contentResolver.query(uri, projection,  selection: null,  selectionArgs: null,  sortOrder: null)?.use { cursor ->
        val relativePathColumn =
            cursor.getColumnIndex(MediaStore.Images.Media.RELATIVE_PATH)
        val displayNameColumn =
            cursor.getColumnIndex(MediaStore.Images.Media.DISPLAY_NAME)
        while (cursor.moveToNext()) {
            val name = cursor.getString(displayNameColumn)
            val relativePath = cursor.getString(relativePathColumn)
            if (name.contains( other: "screenshot",  ignoreCase: true) or relativePath.contains( other: "screenshot",  ignoreCase: true)) {

                ssMsg.insertMessage(SharedPreferanceObject.SBUserId.toString(),calleeId,  Message: SharedPreferanceObject.SBUserId.toString()+" Took a Screenshot!")


            }
        }
    }
}
```

Figure 5.20: query relative data column

Chapter 6. Testing and Evaluation

This application is a social media application, with chat, video/ audio calls and screen sharing. To test this application. we have installed GabChat on 3 android mobile phones.

- Google Pixel 6 – Android 12
- Mi A1 – Android 9
- Poco X3 Pro – Android 11

Then login 3 users in 3 this different android phones. Then verified this user's login. For that first gave the mobile number and generate OTP and verify the user. It redirected to the registration page and provided a username and password and the user successfully login to this application. Verified the authentication. One mobile number can be used only for one user. If the user log-in again using the same phone number, it will auto-populate the username on the registration page.

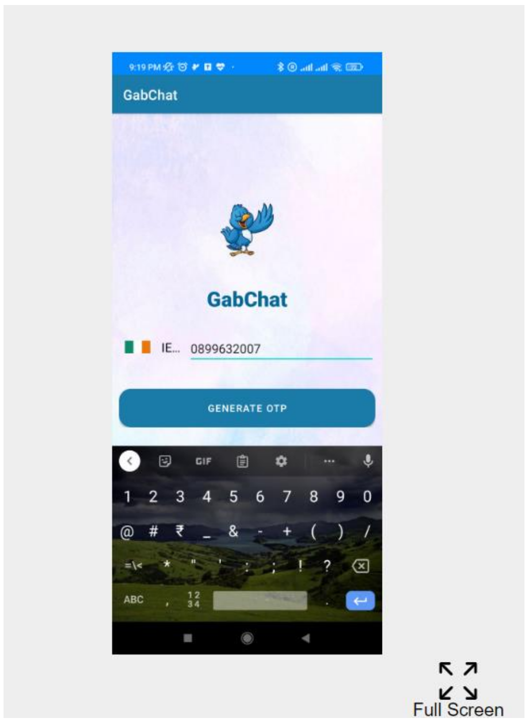


Figure 6.1 Generate OTP

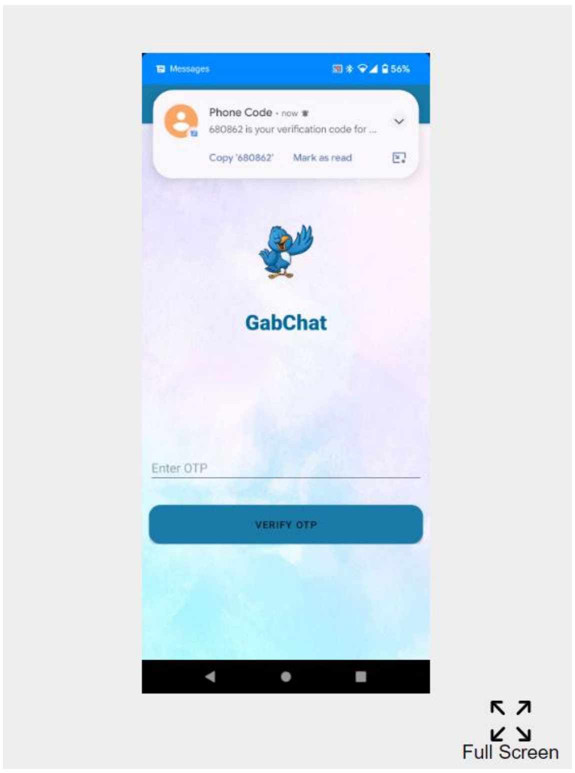


Figure 6.2 OTP verification

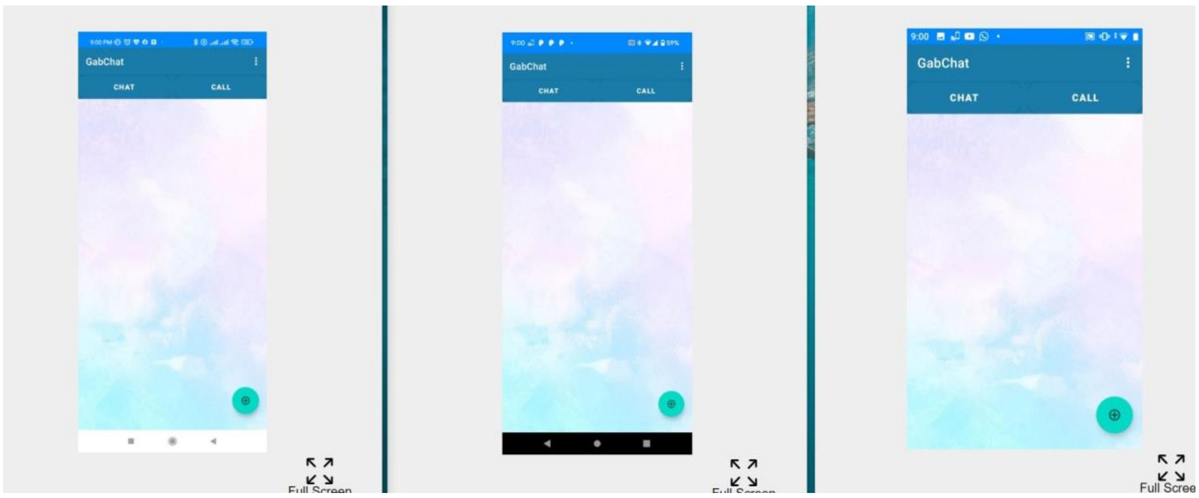


Figure 6.3 Main page

Then we verified that the available user this. By selecting these users from this list, they can communicate with each other.

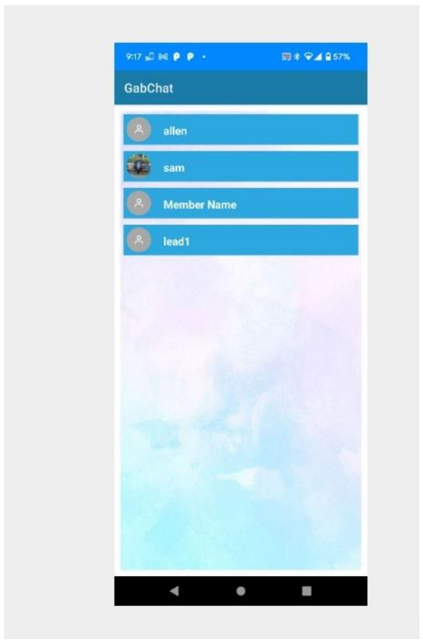


Figure 6.4 User List

Verified that the user can send messages to any individual selected from his contact list and the chat page shows all the previous messages of that person. Also, check that the message was received properly on the selected user and no other user gets any messages. Moreover, verified that this message was a real-time communication between two users. Also verified if anyone took any screenshots of the chat. If taking any screenshot it will detect, and the user will get a notification.

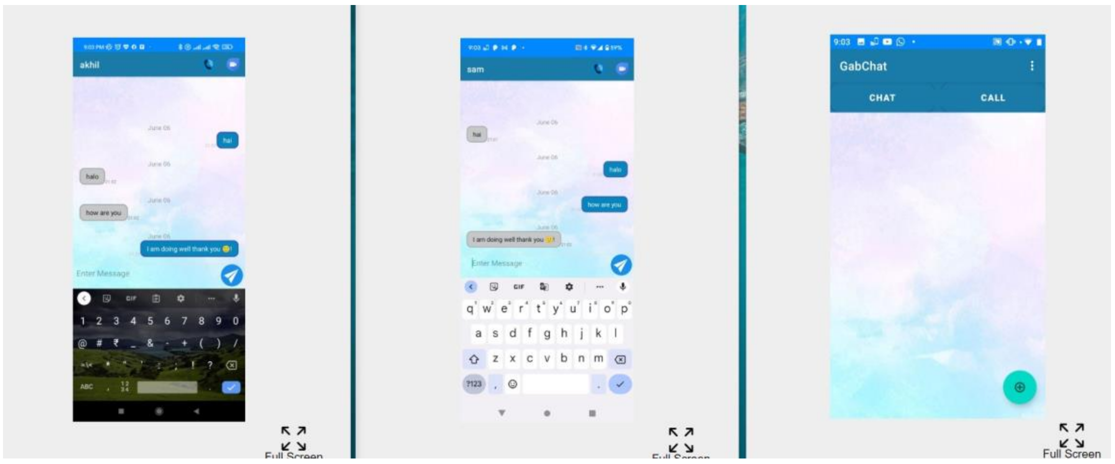


Figure 6.5 real-time chat

Also verified the processing of audio/video call working. Users can select a user from the contact list, and he can make an audio/video call, and verify that this call was received on the selected user only. Then verified that this selected user can accept or decline this phone call. Then verify that the video call. Both users can see each other. Moreover, verified the screen sharing. While making a phone call user can share the screen with the same user. And another end-user can see it on the screen. But at the same time, if anyone took any screenshots, the other user would get a notification acknowledgement.

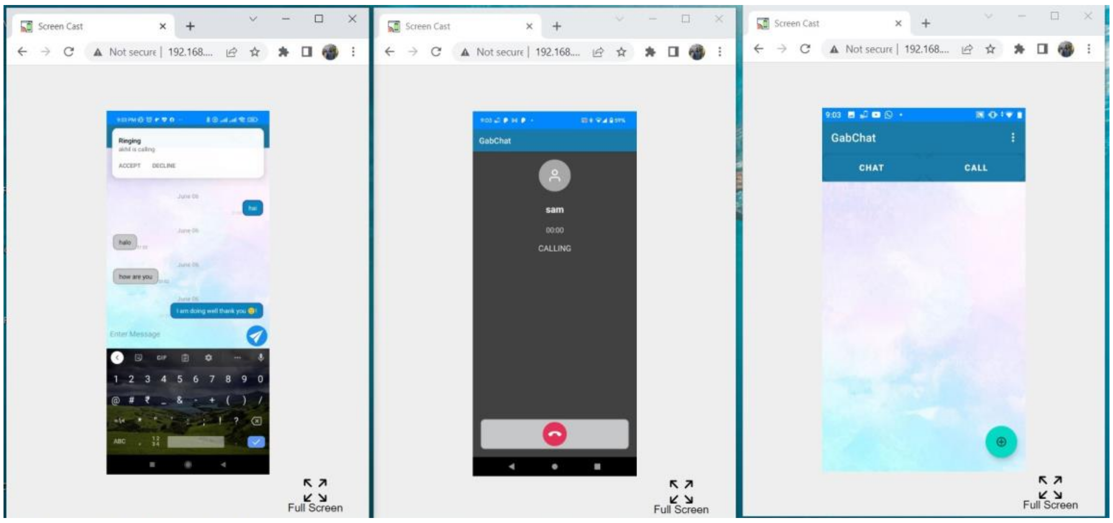


Figure 6.6 audio call established

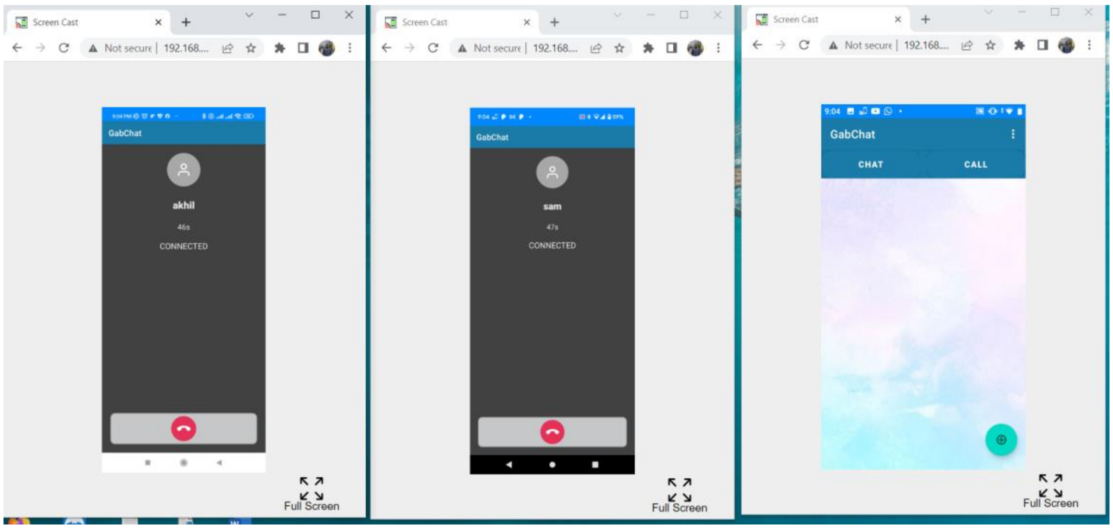


Figure 6.7 Phone call accept

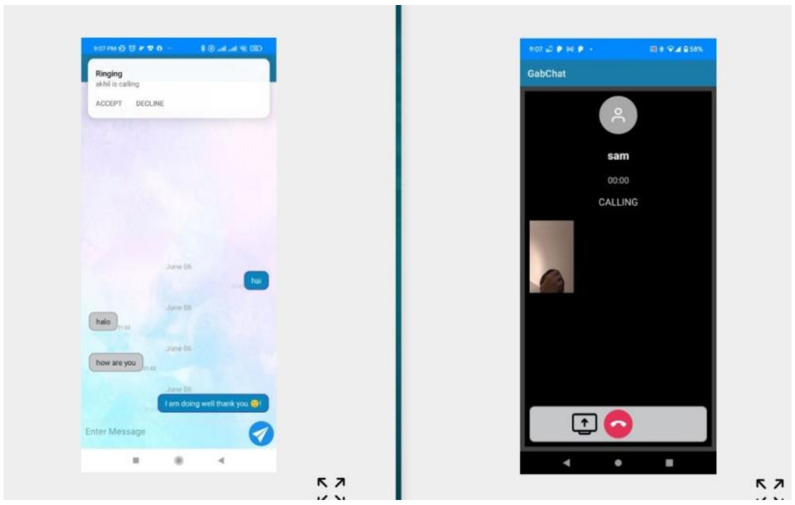

Ringing
akhil is calling
ACCEPT DECLINE
hai
halo
how are you
I am doing well thank you

Figure 6.8 video call

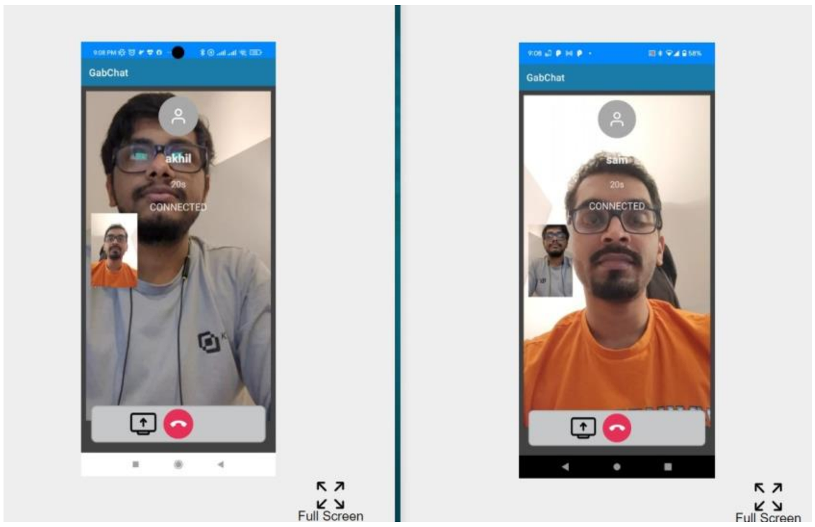


Figure 6.9 video call accept

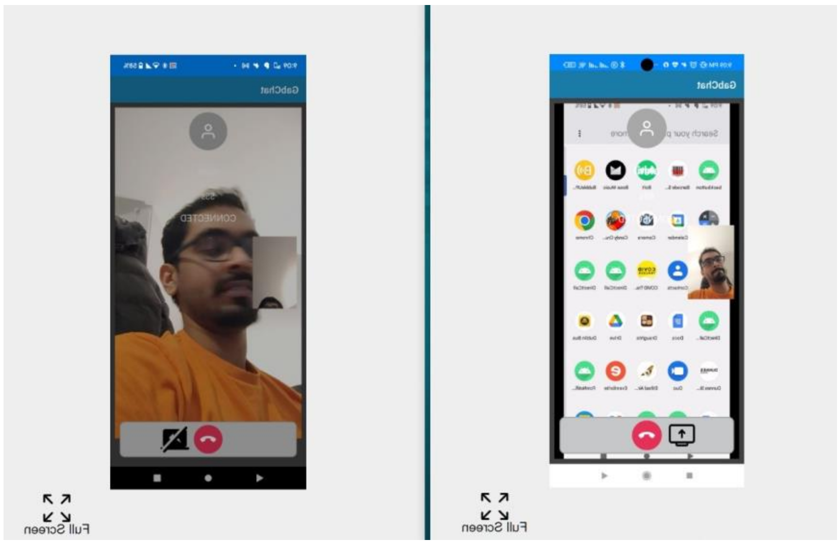


Figure 6.8 Screenshare

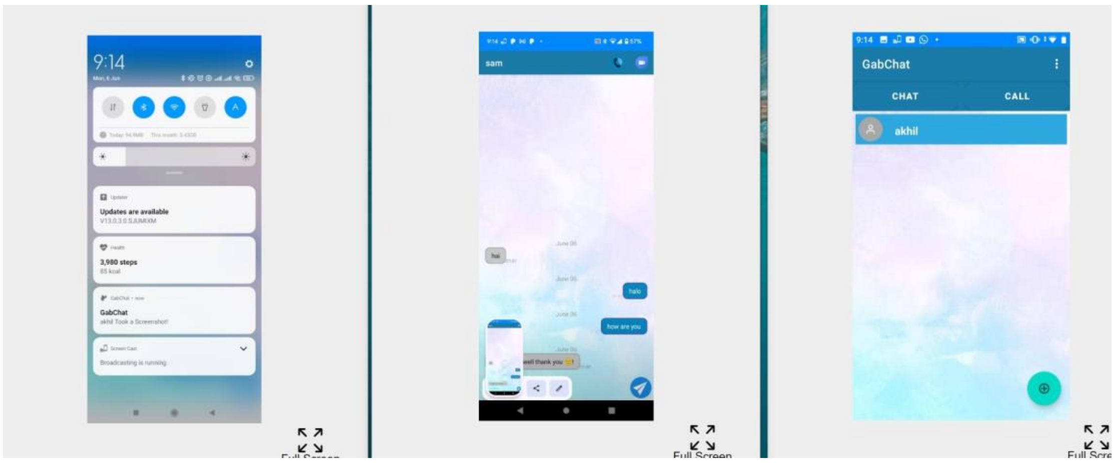


Figure 6.9 screenshot detection

Also verified that multiple users can use this application at the same time. They can call each other and chat swith each other.

s

Chapter 7. Conclusions and Future Work

Social Media is becoming a necessary part of day-to-day life. WhatsApp, Messenger, Snapchat, Skype etc are crucial parts of every human's life where they can communicate with each other in terms of messages, calls, sharing screens etc. As now all the apps are popular still the user runs for a better one where they can find all in one piece. This project is a step to implement the thought called all in one place. The "GabChat" app allows users to communicate with each other in any form irrespective of their location. This application contains messaging, voice calls, video calls and screen sharing methods. Additional features which make this App stand-alone or compete with other apps are the screenshot feature. The user gets notified when the other person tries to take a screenshot of a chat or video call. Also, all the data with encrypted using the AES algorithm.

As a thought feature, this app can be modified in such a way that, any user can block the initiated user when they try to take the screenshot also, we will add block/unblock features. So, users can block the unwanted users. Once a user blocks another user, then he can't make any chat or calls. Also added unblock features. If the user wants to unblock that user, she/she can unblock him.

Also needs to upgrade the new encryption method for this project. Currently, it used the AES algorithm. It is not that much-secured algorithm. If someone finds the secret key, he/she can encrypt all messages. So, in future, we are planning to build a real-time secret key and encryption. For each message, it will create a new secret key and encrypt the data. Nowadays, reels and status are so famous in all applications. So, we are planning to integrate these features in future. So, users can upload reels and images into this system.

Moreover, Currently, we are using the SendBird server was using and maybe they can access the data. we are planning to change the SendBird and will create our server. So, we can create a more secure system.

References

[1]. Sendbird Docs, Send your first message: Chat Android SDK. Available at https://sendbird.com/docs/chat/v3/android/quickstart/send-first-message (Accessed: May 5, 2022)

[2]. Sendbird Docs, Make your first call: Calls Android SDK. Available at https://sendbird.com/docs/calls/v1/android/quickstart/make-first-call (Accessed: May 5, 2022)

[3]. Read and Write Data on Android Firebase Documentation, Available at https://firebase.google.com/docs/database/android/read-and-write (Accessed: May 5, 2022)

[4]. Authenticate with Firebase on Android using a Phone Number | Firebase Documentation. Available at https://firebase.google.com/docs/auth/android/phone-auth (Accessed: May 5, 2022)

[5]. Hussain b, (2017), Applications Platforms and different types of Software. Available at https://medium.com/computing-technology-with-it-fundamentals/applications-platforms-and-different-types-of-software-28f0fc1ef075 (Accessed: May 6, 2022)

[6]. O'Dea S. (2022) Statista, Smartphone subscriptions worldwide 2016-2027. Available at https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (Accessed: May 6, 2022)

[7]. Wallen J. (2021) Why is Android more popular globally, while iOS rules the US? Available at https://www.techrepublic.com/article/why-is-android-more-popular-globally-while-ios-rules-the-us/ (Accessed: May 6, 2022)

[8]. Harvey C. (2021) 11 Best Android IDEs for Developers of 2022. Available at https://www.developer.com/mobile/top-android-ides-for-developers/ (Accessed: May 7, 2022)

[9]. Harvey C. (2017) Eclipse IDE for Android Development Overview. Available at https://www.developer.com/mobile/android/eclipse-ide-android-development/ (Accessed: May 7, 2022)

[10]. Harvey C. (2017) IntelliJ IDEA – Android Programming IDE Overview. Available at https://www.developer.com/mobile/android/intellij-idea-android-programming-ide-overview/ (Accessed: May 7, 2022)

[11]. Harvey C. (2017) NetBeans – Android Programming IDE Overview. Available at https://www.developer.com/mobile/android/netbeans-android-programming-ide-overview/ (Accessed: May 7, 2022)

[12]. Harvey C. (2017), Visual Studio (with Xamarin) – Android Programming IDE Overview. Available at https://www.developer.com/mobile/android/visual-studio-with-xamarin-android-programming-ide-overview/ (Accessed: May 7, 2022)

[13]. Harvey C. (2017), Android Studio – Android Programming IDE Overview. Available at https://www.developer.com/mobile/android/android-studio-android-programming-ide-overview/ (Accessed: May 7, 2022)

[14]. GeeksforGeeks, (2022), Top Programming Languages for Android App Development. Available at https://www.geeksforgeeks.org/top-programming-languages-for-android-app-development/ (Accessed: May 7, 2022)

[15]. Jhonson E. (2021), Kotlin vs Java: Which is the Best Choice for Android App Development? Available at https://medium.com/javarevisited/kotlin-vs-java-which-is-the-best-choice-for-android-app-development-7c9fc782d2c9 (Accessed: May 7, 2022)

[16]. Talend, SQL vs NoSQL: Differences, Databases, and Decision. Available at https://www.talend.com/resources/sql-vs-nosql/#:~:text=SQL%20is%20the%20programming%20language,generally%20do%20not%20use%20SQL. (Accessed: May 7, 2022)

[17]. Ngak d. (2011), Then and now: a history of social networking sites. Available at https://www.cbsnews.com/pictures/then-and-now-a-history-of-social-networking-sites/.(Accessed: May 8, 2022)

[18]. Samur A.(2018) The History of Social Media: 29+ Key Moments. Available at https://blog.hootsuite.com/history-social-media/(Accessed: May 8, 2022)

[19]. Barot T. and Oren E. Guide to chat apps. (Accessed: May 8, 2022)Available at https://towcenter.gitbooks.io/guide-to-chat apps/content/introductionthe_dawn_of/a_brief_ history.html ss

[20]. Patrizio A. (2021) The history and evolution of video sconferencing . Available at: https://www.google.com/amp/s/www.techtarget.com/whatis/feature/The-history-and-evolution-of-video-conferencing%3famp=1 (Accessed: May 12, 2022)

[21]. Alhabash S. and Mengyan Ma (2017) A Tale of Four Platforms: Motivations and Uses of Facebook, Twitter, Instagram, and Snapchat Among College Students? Available at: https://doi.org/10.1177/2056305117691544 (Accessed: May 13s, 2022)

[22]. Heath A. (2021) Facebook is planning to rebrand the company with a new name Available at: https://www.theverge.com/2021/10/19/22735612/facebook-change-company-name-metaverse (Accessed: May 13s, 2022)

[23]. Statista Research Department (2022), Facebook: number of monthly active users worldwide 2008-2022 Available at: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/#:~:text=How%20many%20users%20does%20Facebook,used%20online%20social%20network%20worldwide. (Accessed: May 15, 2022)

[24]. KEMP S. (2022) Twitter statistics and trends , datareportal, Available at : https://datareportal.com/essential-twitter-stats#:~:text=Advertisers%20could%20reach%20465.1%20million,'active'%20social%20media%20platforms. (Accessed: May 15, 2022)

[25]. Murphy D. (2014) 44 Percent of Twitter Accounts Have Never Tweeted , pcmag Available at: https://www.pcmag.com/news/44-percent-of-twitter-accounts-have-never-tweeted (Accessed: May 15, 2022)

[26]. Blagdon J. (2012) Instagram for Android breaks 1 million downloads in less than a day, the verge, Available at: https://www.theverge.com/2012/4/4/2924600/instagram-android-1-million-downloads (Accessed: May 15, 2022)

[27]. sStatista Research Department, (2022) Instagram: number of global users 2020-2025, Social Media & User-Generated Content, Statista , Available at: https://www.statista.com/statistics/183585/instagram-number-of-global-users/ (Accessed: May 15, 2022)

[28]. Ameri A. (2021), Why You Should Never, Ever Use WhatsApp for Business Communication, beekeeper, Available at: https://www.beekeeper.io/blog/why-you-shouldnt-use-whatsapp-for-business-communication/ (Accessed: May 15, 2022)

[29]. Webfx, Top 14 Advantages and Disadvantages of Social Media, Webfx , Available at: https://www.webfx.com/social-media/learn/social-media-marketing-advantages-and-disadvantages/ (Accessed: May 18, 2022)

[30]. Melnichenko L.(2021), Top 10 Chat APIs and SDKs for Android & iOS Worth Your Attention, helpcrunch, Available at: https://helpcrunch.com/blog/chat-apis/ (Accessed: May 18, 2022)

[31]. Alexsam (2019), Top 10 Chat, Audio & Video Calling API & SDK Providers for Enterprise Business, HOW TO BECOME AN AUTHOR, Available at: https://habr.com/en/post/453374/ (Accessed: May 18, 2022)

[32]. Elgersma C.(2018), Everything you need to know about Snapchat, physorg, Available at: https://phys.org/news/2018-06-snapchat.html (Accessed: May 18, 2022)

[33]. https://en.wikipedia.org/wiki/Timeline_of_Snapchat

[34]. IQBAL M.(2022), Snapchat Revenue and Usage Statistics (2022), business of apps, Available at: https://www.businessofapps.com/data/snapchat-statistics/ (Accessed: May 18, 2022)

[35]. Julija A (2022) , WhatsApp Statistics: Revenue, Usage, and History, fortunly, Available at: https://fortunly.com/statistics/whatsapp-statistics/ (Accessed: May 20, 2022)

[36]. Olson P. (2018), Exclusive: WhatsApp Cofounder Brian Acton Gives The Inside Story On #DeleteFacebook And Why He Left $850 Million Behind, forbes, Available at:https://www.forbes.com/sites/parmyolson/2018/09/26/exclusive-whatsapp-cofounder-brian-acton-gives-the-inside-story-on-deletefacebook-and-why-he-left-850-million-behind/?sh=6b7ec5aa3f20 (Accessed: May 20, 2022)

[37]. IQBAL M.(2022), WhatsApp Revenue and Usage Statistics (2022), business of apps, Available at: https://www.businessofapps.com/data/whatsapp-statistics/ (Accessed: May 18, 2022)

[38]. Preston A. How to build in-app chat using Kotlin, Send Bird doc, Available at: https://sendbird.com/developer/tutorials/kotlin-chat-tutorial-part-1 (Accessed: May 20, 2022)

[39]. Preston A., How to build an Android video chat app with Sendbird Calls & Firebase, Send Bird doc, Available at: https://sendbird.com/developer/tutorials /android-video-chat-sendbird-calls (Accessed: May 20, 2022)

[40]. Rodriguez W. How to add screen sharing to your Android app with Sendbird Calls Send Bird doc, Available at: https://sendbird.com/developer/tutorials/screen-sharing-android (Accessed: May 20, 2022)

[41]. Forbes Expert Panel (2021s) , 15 Must-Have Features Of A Successful, User-Friendly Mobile App, Forbes, , Available at: https://www.forbes.com/sites/forbes techcouncil/2021/05/24/15-must-have-features-of-a-successful-user-friendly-mobile-app/?sh=45f5827d6a7a (Accessed: May 20, 2022)