CHAT APPLICATION USING FIREBASE IN KOTLIN

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

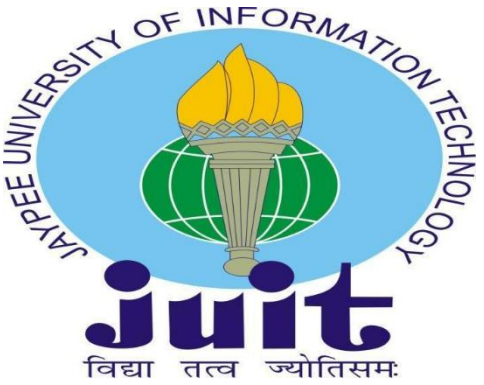
By

Srishti (191551)

Under the supervision of

Dr. Ravindara Bhatt

to



Department of Computer Science & Engineering and Information Technology

The Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

DECLARATION

I at this moment declare that the work presented in this report entitled Chat Application using Firebase and Kotlin in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the Department of Computer Science & Engineering and Information Technology, the Jaypee University of Information Technology Waknaghat is an authentic record of my work carried out over a period from February 2023 to April 2023 under the supervision of Dr. Ravindara Bhatt, Associate Professor (SG). I also authenticate that I have carried out the above-mentioned project work under the proficiency stream Cloud Computing. The matter embodied in the report has not been submitted for any other degree or diploma award.

Srishti 191551

This is to certify that the above statement made by the candidate is accurate to the best of my knowledge.

(Supervisor Signature)

Dr. Ravindara Bhatt

Associate Professor

Dated: 15 May 2023

PLAGIARISM CERTIFICATE

AS PROVIDED BY THE LRC OF JUIT:

ACKNOWLEDGEMENT

First and foremost, I want to praise God for His heavenly grace, which enabled us to finish the project work successfully. My supervisor, Dr. Ravindra Bhatt, Associate Professor, Department of CSE Jaypee University of Information Technology, Waknaghat, has my deepest gratitude and gratitude. My supervisor has a wealth of knowledge and a genuine interest in the "Research Area" needed to complete this assignment. This project was Her never-ending patience, academic leadership, constant encouragement, frequent and vigorous supervision, constructive criticism, insightful counsel, reviewing several subpar versions, and revising them at all levels made this project a possible joy to introduce this project and earnestly thank every individual who helped me in this project.

I express my earnest gratitude to the Jaypee University of Information Technology for giving me an open door and such a decent learning climate. I express my sincere gratitude to the Jaypee University of Information Technology, Solan, for offering help with everything and for giving productive analysis and support which prepared us for the fruitful culmination of the project. I want to offer my genuine thanks to everyone involved for giving me all-important help and support and motivation to embrace this study and make it conceivable.

I'm incredibly grateful to Dr Ravindra Bhatt (Associate Professor), (supervisor for the project and Associate professor) for his important direction and backing. I'm likewise thankful to the subjects of this review for their collaboration and interest. Finally, I thank God and my parents for every one of the endowments. I would also want to express my gratitude to everyone who has directly or indirectly assisted me in making this project a success. In this unusual scenario, I would like to thank the numerous staff members, both teaching and non-teaching, who have created their convenient assistance and helped with my project.

Finally, I must express my gratitude for my parents' unwavering support and patience.

Srishti 191551

TABLE OF CONTENTS

TITLE	PAGE NO
LIST OF FIGURES	6
	7
LIST OF ABBREVIATIONS	8
ABSTRACT	9
1-INTRODUCTION	10
1.1-PROBLEM STATEMENT	10
1.2-PURPOSE	11
1.3-TECHNIQUE	11
1.4 ORGANIZATION	11
2-LITERATURE SURVEY	11-14
3-SYSTEM DEVELOPMENT	15
3.1-SETTING UP FIREBASE	15
3.2-CREATE PROJECT BUTTON	17
3.3-CHOOSE THE PLATFORM	18-19
3.4-AUTHENTICATION AND FIREBASE	21-22
3.5-CONFIGURE REAL-TIME DATABASE IN FIREBASE	22-26
4-EXPERIMENTS & RESULT ANALYSIS	27
4.1-FUNCTIONAL PRECONDITIONS	27
4.1.1-USER AUTHENTICATION	27
4.1.2-NEW CONTACTS ADDING	27
4.1.3-MESSAGE SENDER	27
4.1.4-PUBLICITY MESSAGE	27
4.1.5-STATUS OF MESSAGE	27
4.2-NON-FUNCTIONAL PRECONDITIONS	28
4.2.1-PRIVACY	28
4.2.2-ROBUSTNESS	28
4.2.3-PERFORMANCE	28
4.3-USER CASE DIAGRAM	28
4.4-AUTHENTICATION SYSTEM	29
4.5-CONTACTS FORM	29
4.6-ACTIVITY DIAGRAM	30-31
4.7-USER GUIDE	32
4.7.1-SIGNING UP AS A MEMBER	32

4.7.1.1-GUIDELINES TO REMEMBER	32
4.7.2-LOGIN OPTIONS	33
4.7.2.1-VERIFICATION CODES IN CHAT APPLICATIONS	33
4.7.2.2-VERIFICATION CODE OPERATION	34
4.7.3-PRIVATE MESSAGING	34
4.7.3.1-CHAT PRIVATELY	35
4.7.3.2-BEST PRACTICES TO IMPLEMENT	35
4.7.4-MAKING A FRIEND	36
4.7.4.1-GUIDELINES TO REMEMBER	37
4.7.5-CHAT FORMS	38
4.7.5.1-GUIDELINES TO REMEMBER	39
4.7.6-FORM A GROUP	40
4.7.6.1-GUIDELINES TO REMEMBER	41
4.7.7-ACCOUNT PREFERENCES	42
4.7.7.1-BEST PRACTICES TO REMEMBER	42
5-CONCLUSIONS	45
5.1-FINAL THOUGHTS	45
5.2-THE FUTURE	45
5.3-CONTRIBUTIONS FROM APPLICATIONS	45

LIST OF FIGURES

NO OF FIGURES	Page No.
FIGURE 1	15
FIGURE 2	16
FIGURE 3	16
FIGURE 4	17
FIGURE 5	17
FIGURE 6	17
FIGURE 7	18
FIGURE 8	18
FIGURE 9	19
FIGURE 10	19
FIGURE 11	19
FIGURE 12	20
FIGURE 13	20
FIGURE 14	20
FIGURE 15	20
FIGURE 16	21
FIGURE 17	21
FIGURE 18	21
FIGURE 19	21
FIGURE 20	22
FIGURE 21	22
FIGURE 22	23
FIGURE 23	23
FIGURE 24	23
FIGURE 25	24
FIGURE 26	24

FIGURE 27	24
FIGURE 28	25
FIGURE 29	25
FIGURE 30	26
FIGURE 31	27
FIGURE 32	28
FIGURE 33	28
FIGURE 34	29
FIGURE 35	30
FIGURE 36	30
FIGURE 37	32
FIGURE 38	33
FIGURE 39	35
FIGURE 40	36
FIGURE 41	38
FIGURE 42	39
FIGURE 43	41

LIST OF ABBREVIATIONS

- IDE-INTEGRATED DEVELOPMENT ENVIRONMENT
- IOS-IPHONE OPERATING SYSTEM
- MVVM-MODEL VIEW VIEW MODEL
- UI-USER INTERFACE
- SDK-SOFTWARE DEVELOPMENT KIT
- IEEE-INSTITUTE OF ELECTRONICS AND ELECTRICAL ENGINEERS
- SMS-SHOR MESSAGE SERVICE
- GSM-GLOBAL SYSTEM OF MOBILE COMMUNICATION
- ID-IDENTITY DOCUMENT
- API-APPLICATION PROGRAMMING INTERFACE
- TCP/IP-TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL
- URLs-UNIFORM RESOURCE LOCATORS
- GCM-GENERAL CICULATION MODEL
- FCM-FUFUTUES COMMISSION MERCHANT
- SHA-1-SECURE HASH ALGORITHM

ABSTRACT

Humans are social beings that need to communicate regularly. One of the common methods of communication is to take advantage of the advancement of digital technology such as chat applications. Such a chatting application was developed using Android Studio IDE and Firebase Database. Communication through the web is turning out to be crucial nowadays. Online communication permits clients to speak with others quickly and advantageously. Considering this, the online communication application should be capable of offering the writings or pictures or some other documents more quickly with the least postponement or with no deferral. Firebase is one of the stages which gives an ongoing information base and cloud administration which permits the designer to make these applications effortlessly. Texting can be considered a stage to maintain communication. Android gives a better stage to create different applications for texting contrasted with different stages like iOS. The fundamental target of this paper is to introduce a product application for the starting of continuous communication between administrators/clients. The framework created on Android will empower the clients to speak with other clients through Chat messages with the assistance of the web. The framework requires the gadget to be associated with the web. This application depends on Android with the backend given by Google Firebase.

CHAPTER-1 INTRODUCTION

The purpose of this report is to give a thorough explanation of a chat application created with Kotlin, Data binding, MVVM architecture, Firebase real-time database for real-time chatting, and Firebase auth for authentication. The programmer has several functions including Login, Register, Chats, Users, Groups, Profile, Group Chatting, One-to-One Chatting, Seen & Delivered, Group Seen, Send Text & Photos, and Message Functionality. To highlight the application's functionality and technical features, the report gives a thorough overview of the application's features, architecture, and implementation.

This project entails creating a chat application utilizing Kotlin, Data binding, MVVM architecture, Firebase real-time database for real-time chatting, and Firebase auth for authentication. The application has several functions including Sending text and photographs, message capability, Seen & delivered, Group saw, Login, Register, Chats, Users, Groups, Profile, and Group Chatting. Real-time chatting can be implemented by utilizing Firebase's real-time database, and Firebase Auths provides safe user authentication. The MVVM pattern, which enables a distinct separation of responsibilities between the data, UI, and business logic layers, serves as the foundation for the application's design.

The UI code is made simpler and requires less boilerplate code thanks to data binding. The overall goal of this project is to develop an intuitive chat application that makes use of Firebase to provide users with a quick, dependable, and secure chat experience.

1.1 PROBLEM STATEMENT

The creation of a chat program that enables real-time user authentication while allowing for user communication. Every major programmer now available seems to have a chat feature, therefore yours should be no different! Making a chat tool, though, can seem like a daunting endeavor. There aren't any built-in chat features in Ukti, hence a server is required to organize and archive user talks.

Thankfully, there are some excellent frameworks available to support you. Without creating a single line of server code, you can synchronize real-time data with Firebase. You can get a message UI from Message Kit that is comparable to the built-in Messages app. You'll create RWRC, an anonymous chat application, in this lesson. This kind of tool is already familiar to you if you've used IRC or Slack. You will gain knowledge on various statuses like utilizing Firebase for anonymous authentication, making several chat channels, using Message Kit to create a comprehensive chat UI, real-time data synchronization with the Fire Store database possible ansendingnd photos, and utilizing Firebase Storage.

1.2 PURPOSE

The following are the project's goals:

- To create a chat application for real-time communication utilizing Firebase's real-time database.
- Using Firebase auth to guarantee secure user login.
- To create a user-friendly user interface using Kotlin and data binding.
- To separate the responsibilities of the data, UI, and business logic levels using the MVVM design.
- To integrate functionality like Login, Register, Chats, Users, Groups, Profile, Group Chatting, One-to-One Chatting, etc. sent text and photographs, group seen, seen and delivered, and message functionality.

1.3 TECHNIQUE

The following methodology was employed in the development of the chat application:

- Obtaining and analyzing requirements.
- System architecture and design.
- Data binding and Kotlin implementation.

- Debugging and testing.
- Deployment and upkeep.

1.4 ORGANIZATION

The structure of this report is as follows:

- A literature review of the relevant material published in common books, journals, transactions, and online portals is presented in Chapter 2.
- The system development process, encompassing analysis, design, development, algorithm, and model development, is discussed in Chapter 3.
- The experiments and the result analysis are presented in Chapter 4.
- The chat application's conclusions, future potential, and contributions are presented in Chapter 5.
- Finally, an IEEE formatted list of all references utilized in this paper is provided.

CHAPTER-2 LITERATURE SURVEY

To find the relevant material that is available in common books, journals, transactions, and online portals, a literature survey was undertaken. The development of chat applications, the use of Firebase real-time databases, Firebase authentication, Kotlin, data binding, and MVVM design were the main topics of the survey's three-to-five-year focus.

Text-based messaging services that enable instant communication have been developed since the introduction of mobile phones. For this reason, Friedhelm Hillebrand and Bernard Hillebert developed the concept of SMS in the Franco-German GSM collaboration in 1984. The size restriction that can be used to create a message, namely 128 bytes, was the new idea's main disadvantage. In 1992, the first SMS was delivered following several adjustments based on the original concept. The next year, Telia and Aldiscon of Sweden developed the first commercial SMS service.

Although SMS was frequently used for quick communication and the delivery of emergency messages, its high cost was a drawback. In the 2000s, SMS dominated the communication landscape. Different messaging applications based on various operating systems became accessible and gained popularity among the public with the introduction of smartphones in the late 2000s.

The most well-known ones were WeChat, Telegram, Viber, Snapchat, WhatsApp, and a few more. Google [2] developed the Android operating system for cell phones, which is widely used by users around. Android applications [3] are typically developed using the IDE Android Studio and a variety of programming languages, including Koltin, Java, Dart, and others. When creating applications, most of the server-side work is handled by Firebase. Because it is a NoSQL database, users can store, retrieve, and sync data in real time via sockets. Firebase is a crucial tool for development from the perspective of a developer due to several factors. Keeping work delays to a minimum promotes cooperation between clients and developers. The key elements or services offered by Firebase for creating chat or communication applications are:

Realtime Database: A cloud-hosted database is called Realtime Database. Each related client receives a constant sync of the data, which is saved in JSON format. Most user requests for cross-platform applications created using JavaScript, iOS, and Android [5] SDKs are based on a live database instance that is updated with fresh information. With the help of this capability, developers can bypass the database construction process and let Firebase take care of most of the application's backend. It offers a versatile expression-based rule language to specify when information can be created or read, as well as how it should be organized.

Firebase Authentication: Both users and developers can benefit from Firebase Authentication. The login feature can take some time and effort to develop and maintain. For login, Firebase offers a straightforward API. Users can manually integrate one or more login methods utilizing SDK to log into their Firebase app.

Storage: App developers who need to store and serve user-generated content, like photographs or other files, can utilize the Firebase Storage service. It offers secure document transfers and downloads for Firebase apps regardless of network situation.

A cross-platform option that enables developers to transmit messages dependably and without charge is cloud messaging. Developers have the option of sending notifications to users to encourage re-engagement and maintenance.

Wi-Fi connectivity was advised by Ekata M. Lamb Ture et al. (2016) for sending and receiving text messages and files. Because Wi-Fi technology is inexpensive and uses little power, they include a system that enables users to chat with one another. The devices can be connected using Wi-Fi hotspots that let users connect. The Wi-Fi range is constrained, and as the range expands, the signal strength may change, therefore this method may have drawbacks. Since there is no message security, information breaches may result.

A paradigm for intranet users was developed by Nikhil Chaudhari et al. in May 2018. Users can converse and interpret communications in the language of their choice in this paradigm. The ability to access the internet is not required. Costs associated with communication have decreased as a result. This model also features landmark identification, on-demand image theft alarms, and image backup. By using juxtaposition as a server, the method enables user communication via a single network. The messages have a maximum size, and an exception occurs when they go over that limit.

The current public app is examined by Sai Spandhana Reddy Emadi et al. in January 2019 in [9], where they also suggest a method for sending and receiving messages that makes use of a backend service named Firebase to store data. As a result, an instant messaging service will be accessible. The user would only have access to chat rooms. They can also choose to use a pdf writer and reader. In addition, they performed a survey of users' opinions, asking them about their favorite features, the reason they use messaging apps, how much time they spend using them daily as a whole, their preferred chat app, and some other things.

A system [10] that enables two users on the network to communicate using text and text-based media such as photos, audio, video, and text online in real-time was proposed by Ashita et al. (2020). They used the Android and Google Firebase operating systems to administer the contact back functionality, highlighting several features of the app and service.

Users communicate via a Firebase cloud messaging server, and data is stored locally and encrypted with a symmetric key, according to a concept [11] proposed by Noor Sabah et al. in 2017. The model also illustrates how to secure sessions using session keys and how to encrypt messages using a private and public key using the Xsalsa20 encryption algorithm. The user will have the ability to initiate chats with other users. On the server, Node.js and MongoDB are used to provide quick access.

The workings of an online text messaging program where users can connect and speak with one another using the application are described in the paper [12] by Prabhat Kumar Patel et al. on "Android Based Chat Messaging Application Using Firebase". Users can register for the app using their cell phone number, and the backend is powered by Google Firebase. Additionally, users can search for and interact with other users using their names. Users can access their accounts at any time, from any mobile phone, thanks to the authorization feature.

In their article [13], S. Nayak et al. (2017) introduce the idea of using a user-defined password to create a hash that is then used as the key for the aes256 method to encrypt user messages. The concept makes use of the TCP/IP protocol for data transfer as well as the HTTP protocol to establish a connection between the server and the user. The connection is established in two modules, the first of which helps with the protocol for traffic control and the second of which helps with the addition of new users. The concept also employs a crypto message that serves as a liaison between the database and the Google Cloud messaging server, which oversees creating a distinctive id and facilitating connection with devices.

In [14], Chatterjee et al. (2018) developed a mobile app with the notion that communication is crucial for data exchange. The availability of real-time data is the primary focus of this study. For real-time data processing connection from the user interface to the cloud and vice versa, they used Firebase. In his article, Walter Kriha discusses the methodical brief review of common ideas in NoSQL databases, methodologies, design patterns, and

several NoSQL class types of databases (document databases, columnar databases, key/value stores) related databases) as well as specific goods.

Various benefits and drawbacks of adopting NoSQL databases have been talked about. Comparative research was done by Supriya S. Pore and Salaya B. Patwari. Research NoSQL and SQL. The study emphasizes many sorts. It also distinguishes between databases like SQL and NoSQL. one of them. The SQL and NoSQL databases' axioms in this paper have been detailed.

According to the report, NoSQL databases do not apply the ACID property or data consistency. Databases Meenu Dave and Vatika Sharma have provided an overview of NoSQL databases emphasizing how it has decreased the use of dominance of SQL, including its history and traits. In his essay, Daniel Pan demonstrates how to connect Firebase to an Android application and the fundamentals of creating the framework of a Firebase database.

In a Landon Cox study, the contrast between SQLite and Firebase, too. Additionally, it emphasizes arranging data in the form of to store in Firebase and create a JSON tree. Maintaining categories and adding subcategories is our primary goal so that folks may more easily locate what they were seeking quickly. Our secondary objective was to quietly indicate well-known dishes that could be made quickly. Faster preparation time

Recommendations were thus introduced in place of deliveries. There are presently many visiting applications that are being used effectively by people. to strengthen the application, client feedback regarding what is needed, and what is accessible from the present applications is already there.

The latest applications, including Facebook, Telegram, Hike, WhatsApp, and Snap Chat Messenger, were used as the basis for this study. The 50 attendees, who were of all ages, were asked the following questions. Gatherings.

The mobile and web application development platform Firebase provides several services to assist developers in creating and maintaining their applications. A real-time database is one of the services offered by Firebase that can

be used to create chat applications. In this review of the literature, we'll look at some recent studies and research on using Firebase to create chat applications for Android

By Roshni M. and Sagar S., "Developing Mobile Chat Application Using Firebase and Android" (2021). This paper offers a guide on how to create an Android chat application using Firebase. The authors walk readers through the process of setting up a Firebase project, configuring a real-time database, and creating an Android Studio chat interface.

Raja Kumar and Naveen Kumar's "Real-time Chat Application Using Firebase and Android" was published in 2020. The concept and development of a real-time chat application utilizing Firebase and Android are covered in this paper. The authors detail how they managed user login using Firebase login, stored chat data in Firebase Realtime Database and sent push alerts using Firebase Cloud Messaging.

Paul Trebilcox-Ruiz's "Building a Chat Application with Firebase and Android" was published in 2019. This tutorial gives a general overview of how to create an Android chat application using Firebase. The author explains how to configure the real-time database, set up Firebase, and build a new project. Additionally, he shows how to create the conversation interface using Card View and Recycler View.

Siddharth Mishra and Sandeep Kumar Gupta's "Firebase for Mobile Development" was published in 2018. This book offers a comprehensive tutorial on using Firebase to create mobile applications. Firebase Realtime Database, Firebase Authentication, Firebase Cloud Messaging, and Firebase Storage are just a few of the services that the writers discuss. They also give examples of how to create chat applications using these services.

By Ravi Tamada, "Building a Chat Application Using Firebase" (2017). An overview of how to create a chat application using Firebase is given in this tutorial. The author explains how to configure the real-time database, set up Firebase, and build a new project. Additionally, he provides an example of how to create the chat interface using List View and Array Adapter.

In conclusion, Firebase is a well-liked framework for creating chat applications for Android. Step-by-step instructions for setting up a Firebase project, configuring the real-time database, and creating the chat interface using Android Studio are provided in numerous research studies and tutorials

CHAPTER-3 SYSTEM DEVELOPMENT

Kotlin chat application development calls for careful consideration of several technological factors. We will examine some current studies and research on the system development of chat applications for Android using Kotlin in this review of the literature.

Jocelyne Martins and Rodrigo Oliveira's "Developing a Real-Time Chat Application with Kotlin" is due out in 2021. This article provides a tutorial on how to create a Kotlin real-time chat application. The authors explain how to incorporate Firebase as a backend solution, handle data persistence, and implement the user interface.

By Saurabh Mishra and Alok Kumar, "Building a Chat Application with Kotlin and Firebase" (2020). In this paper, the design and development of an Android chat application utilizing Kotlin and Firebase are discussed. The writers go over user identification, real-time messaging implementation, and storing chat data in the Firebase Realtime Database.

By Mike James, "Building Chat Applications with Kotlin and Android" (2019). This tutorial gives a general overview of how to create an Android chat application using Kotlin. The author discusses how to incorporate Firebase as a backend solution, handle data persistence, and implement the user interface.

Eric Decamine published "Developing a Chat Application with Android and Kotlin" in 2018. This article offers a step-by-step lesson on how to create an Android chat application.

Ray Wunderlich’s "Kotlin and Firebase Chat App Tutorial" was published in 2017. The detailed instructions in this tutorial show you how to create a chat application using Kotlin and Firebase. How to design the user

interface, manage user authentication, store chat data in the Firebase Realtime Database, and implement real-time messaging are all topics covered by the author.

In summary, careful consideration of many technological elements is necessary while creating an Android chat application using Kotlin. Step-by-step instructions on how to construct the user interface, deal with data persistence, and integrate Firebase as the backend solution are provided by recent studies and research. These tools can assist programmers in creating powerful and effective chat applications for the Android operating system.

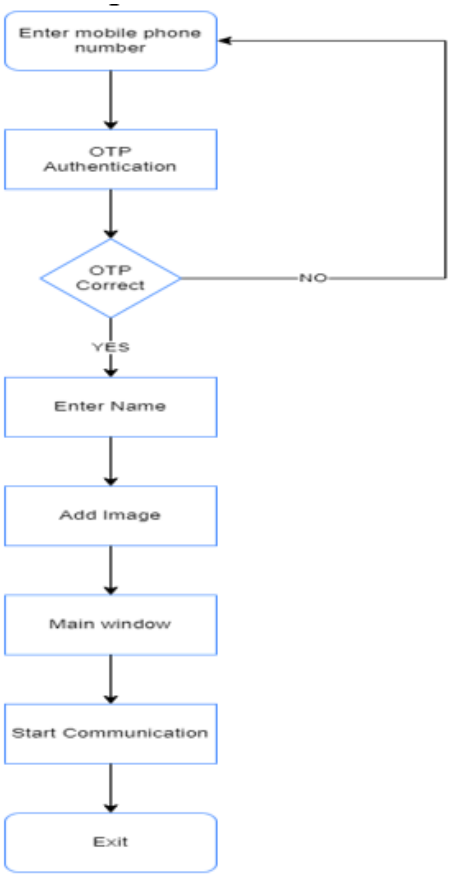


Fig. 1 Working flow chart of application

FIGURE 1

The system development process involved the following steps:

3.1 SETTING UP FIREBASE

- For the following reasons, Firebase may be necessary:
- Firebase phone number-based user authentication for your app.
- Push notifications from FCM, formerly known as GCM.
- It could appear a little ominous at first. But do not panic; instead, let's examine the proper procedure step by step.
- Registration of a Firebase account and projects
- To sign up for Firebase and set up a Firebase project, follow these instructions:
- At the Firebase console, create a Firebase account. You may log into Firebase using your Google account.

3.2 CREATE PROJECT BUTTON

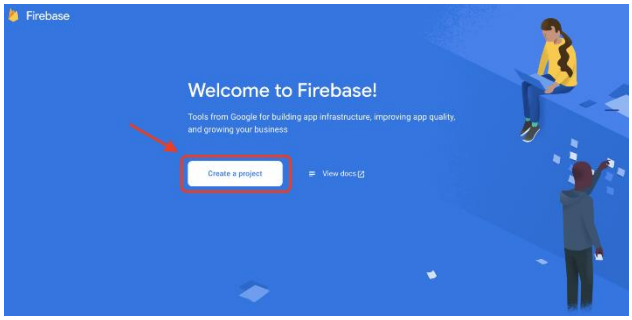


FIGURE 2

- Reminder: From the Project name dropdown menu, choose your Google project if you have one registered for your mobile app.
- If necessary, you can also change your Project ID. Each project receives an automated identification number. Firebase functionalities that are accessible to the public use this ID. It will be applied, for instance, to your Firebase Hosting subdomain and database URLs. You can alter it if a particular subdomain is required.
- Click Continue after completing the necessary fields (such as Project name and Project ID).
- Add a Project

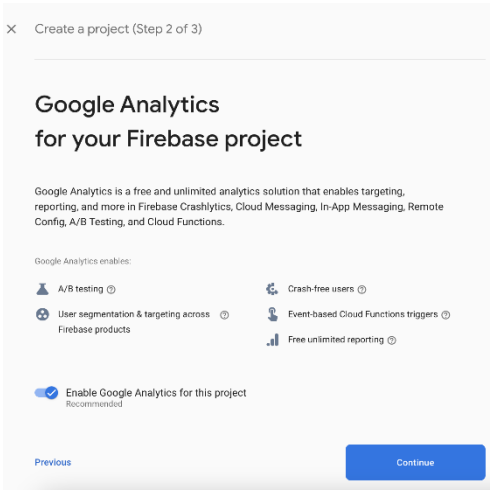


FIGURE 3

- Click Create Project after configuring Google Analytics for your endeavor.

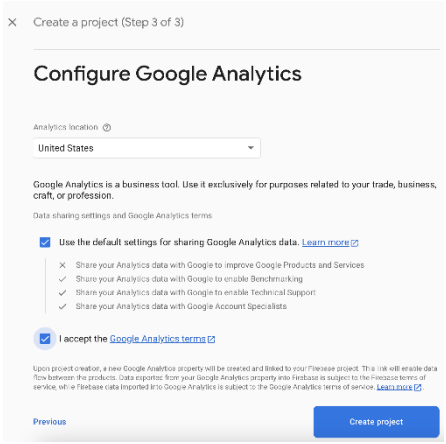


FIGURE 4

- then click on "Next"

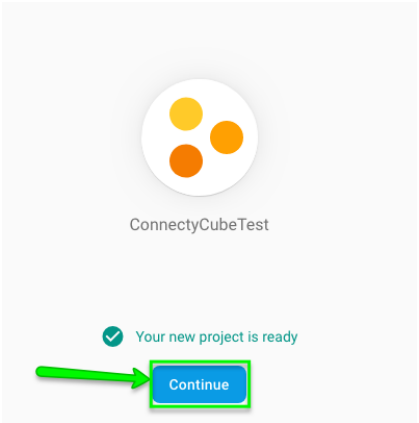


FIGURE 5

3.3 CHOOSE THE PLATFORM FOR WHICH FIREBASE IS REQUIRED

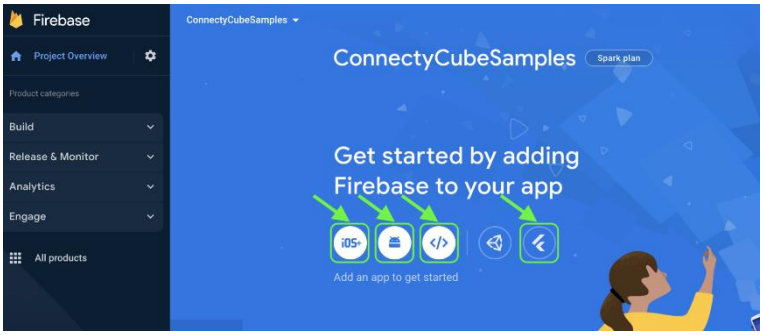


FIGURE 6

- On the Add Firebase to Your Android App screen, complete the forms and click Register App.

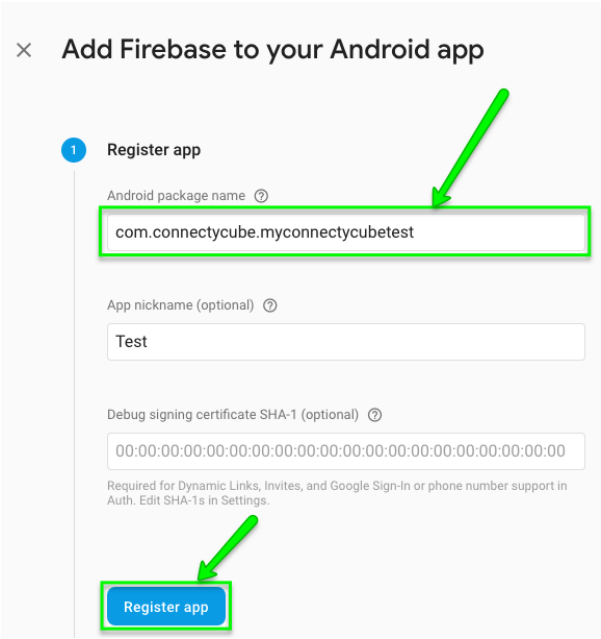


FIGURE 7

- Link the Firebase SDK.
- To successfully add an Android project to Firebase, the following conditions must be met:
- OS 4.0 or later for Android
- at least Google Play Services 15.0.0
- A recent release of Android Studio
- A step-by-step tutorial for connecting Firebase SDK to your Android project is provided here:
- Get the Google-Services. JSON configuration file.

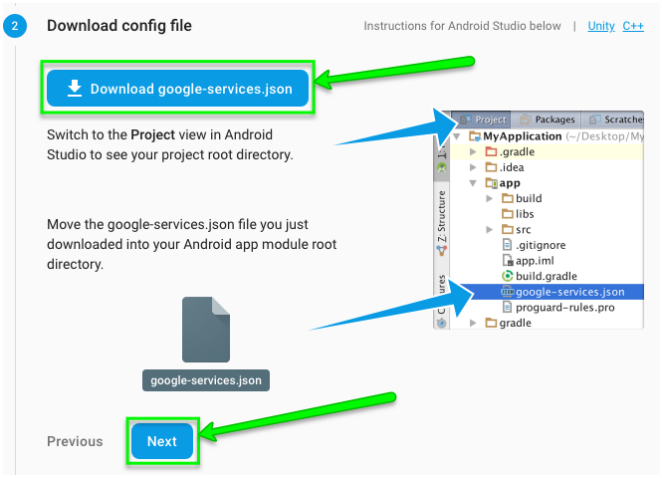


FIGURE 8

- Upload the Google Services. JSON file you just downloaded into the root directory of your Android app module in Android Studio's Project view.

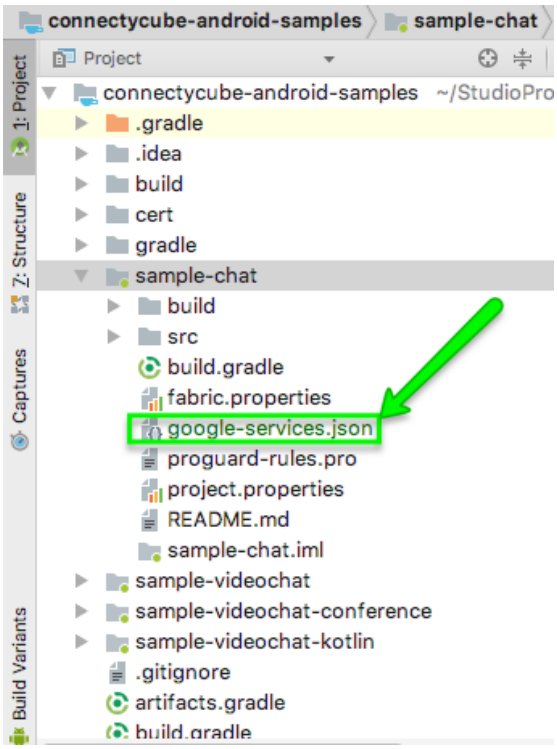


FIGURE 9

- In your Firebase console, add the Firebase SDK to the instructions.

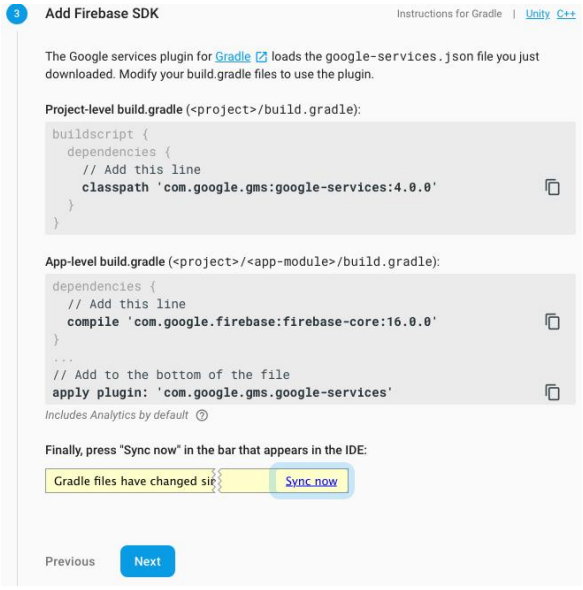


FIGURE 10

- For your project build, include the Google services plugin. File Gradle

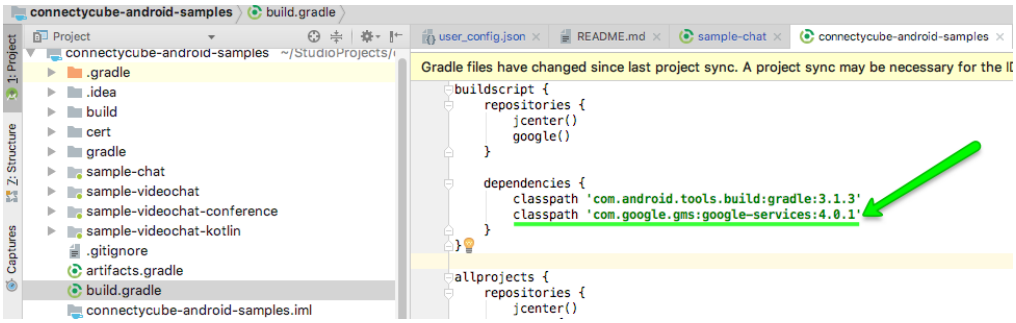


FIGURE 11

- In your module build, include the Google services plugin at the bottom. file for Gradle.

FIGURE 12

- The pop-up in your Android Studio should say "Sync Now."

FIGURE 13

3.4 AUTHENTICATION AND FIREBASE

- Users of your app can log in using their phone numbers with this option. The user receives an SMS with a verification code and authenticates in your app using that code if you use this approach for user authentication.
- To add Firebase authentication to your Android project, you must take the following actions:
- Include a dependency for Firebase authentication in your Module build. the Gradle file

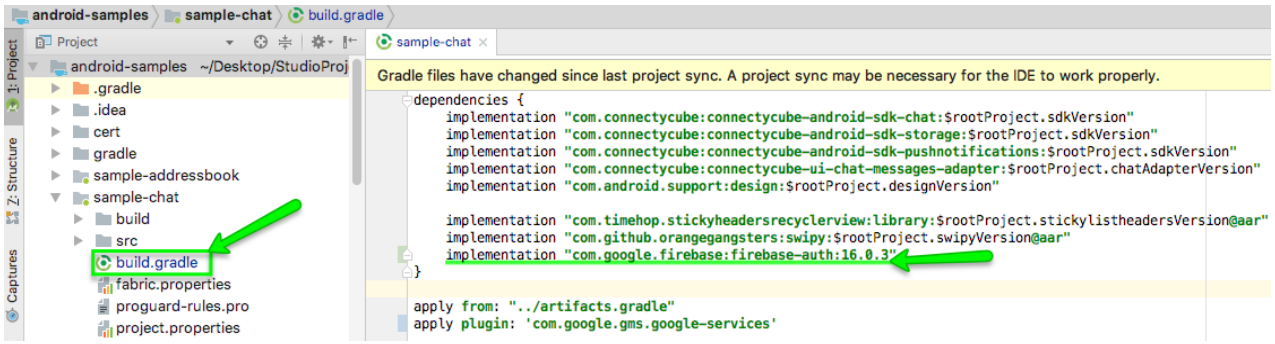


FIGURE 14

- Follow Android Studio's prompts to sync your project:

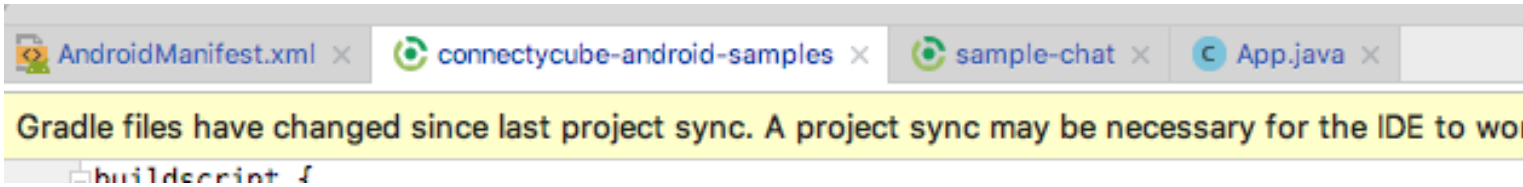


FIGURE 15

- Check the instructions for Authenticating Your Client to find the SHA-1 hash for your app.
- In the Project settings tab of your Firebase interface, add the SHA-1 hash of your app:

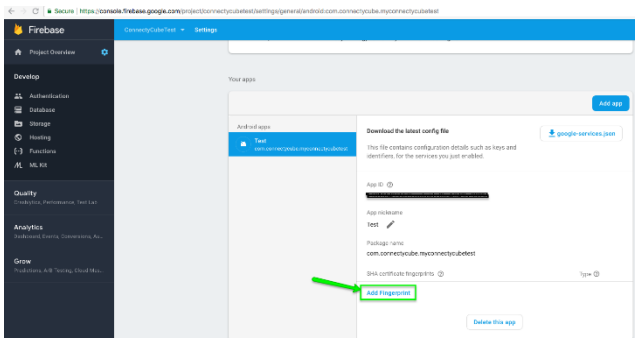


FIGURE 16

- After that:

FIGURE 17

- Go to Authentication in the Firebase console. Method of sign-in section:

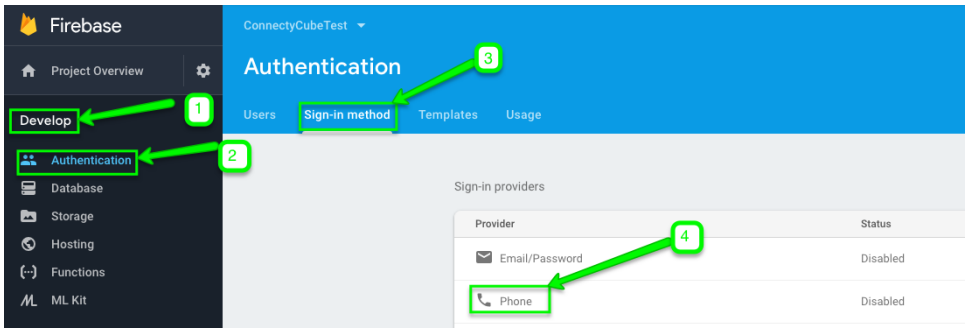


FIGURE 18

- Activate the phone number login option:

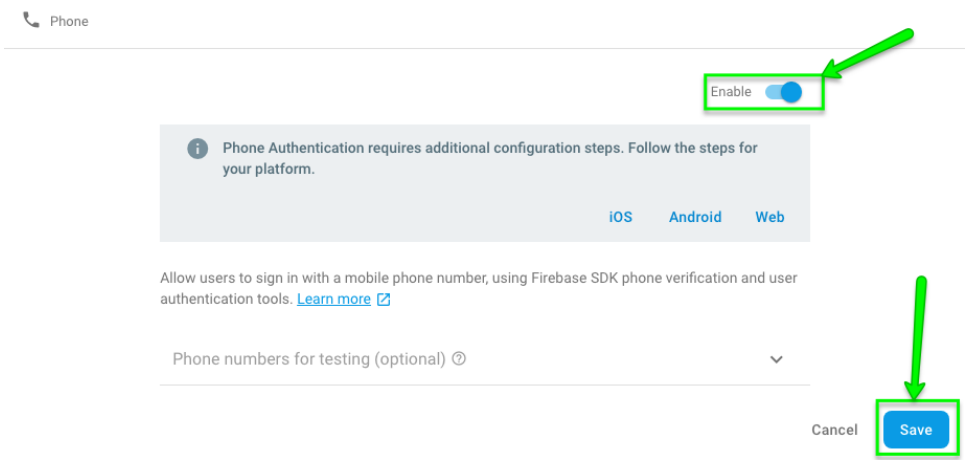


FIGURE 19

- Access your Firebase console and find your Project ID:

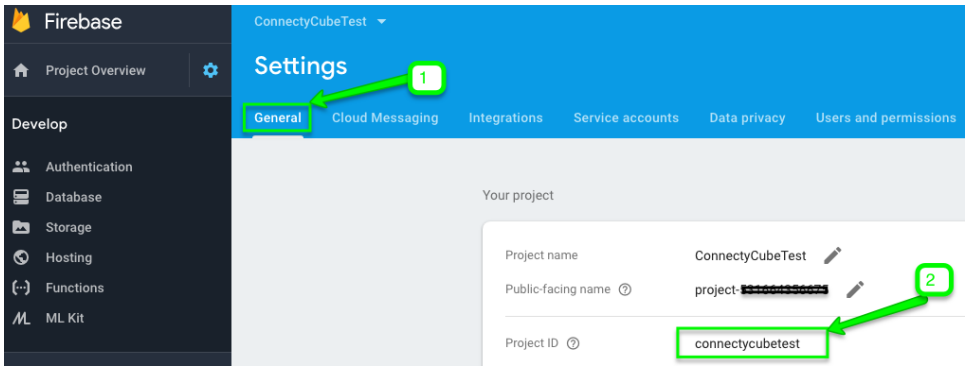


FIGURE 20

- Run a test by copying it. Once your user has successfully logged in, you may find him or her in the Users area of your Dashboard under "Your App."

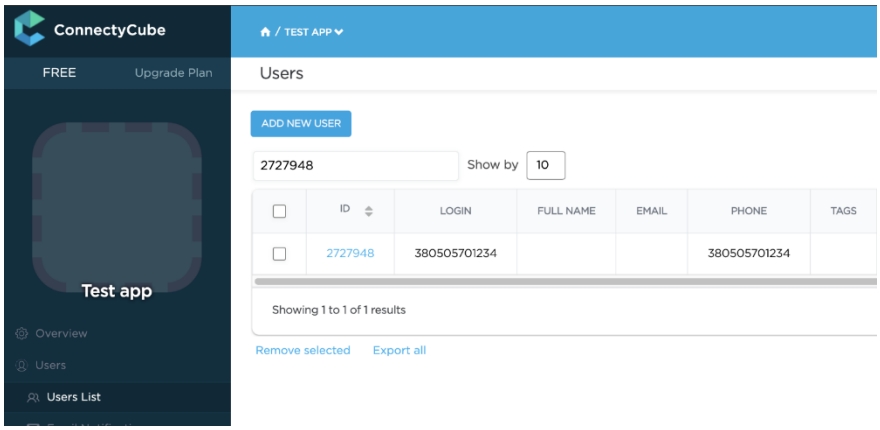


FIGURE 21

3.5 CONFIGURE REAL-TIME DATA IN FIREBASE

We learned about the Firebase Real-time database, its main features, and alternatives in the section before this one. We will now go over how to set up and configure an Android application to use Firebase's Real-time database. The first step will remain the same, however in this section, Kotlin will be used in place of Java. To set up and set up the application to use a Real-time database in Firebase, let's start with the initial steps and go into further detail on each one.

- In the first step, we'll start a fresh Android Studio project, call it Firebase Real-time Database Example, and add an empty activity written in Kotlin.

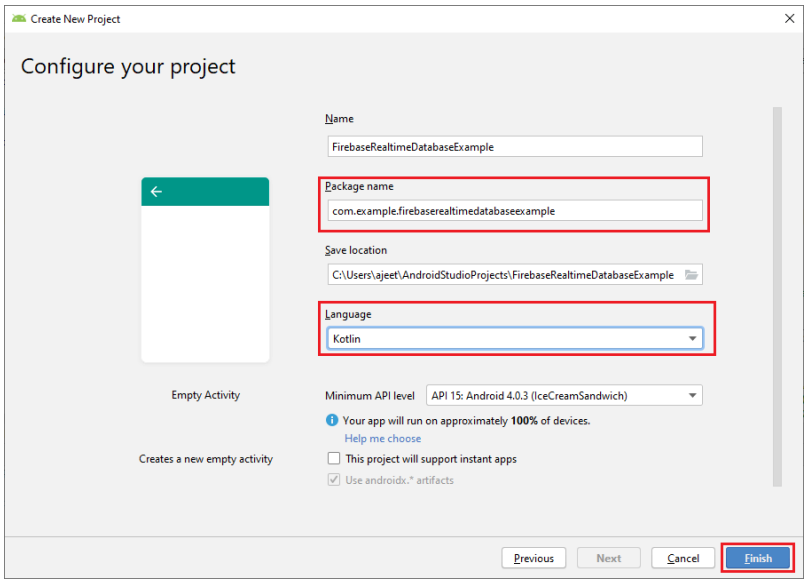


FIGURE 22

- In the following step, we will either use Firebase Assistant or a console manually to connect our Android application to Firebase. Next, we will add to our app. Gradle files all the necessary libraries and plugins. Additionally, we will add all our projects to our repository, maven Local().

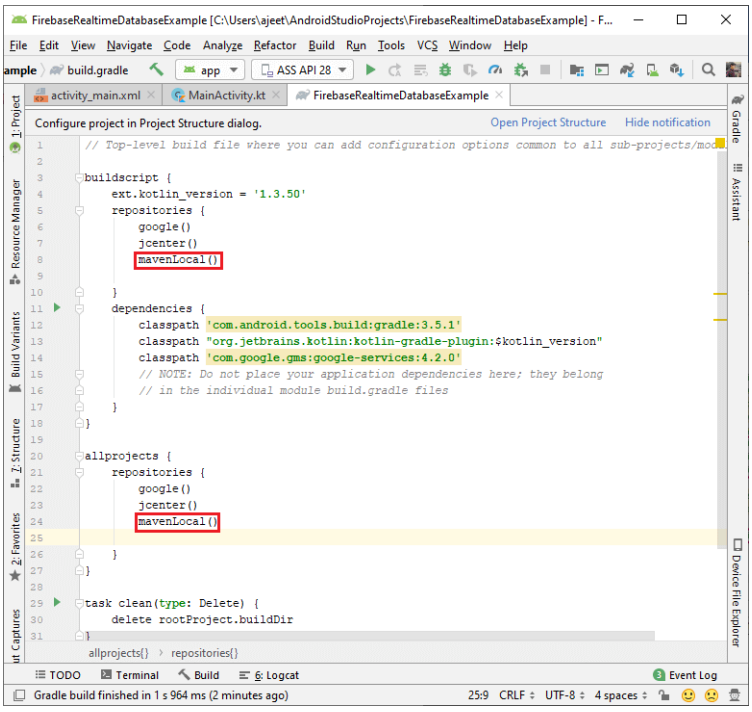


FIGURE 23

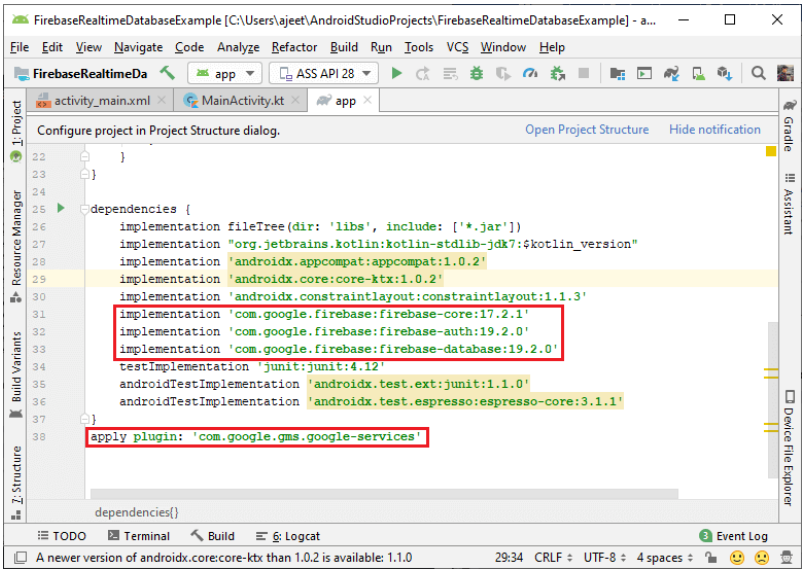


FIGURE 24

- The Real-time database will be examined in the following phase when we visit the Firebase console. There will be two options under Developers-> Database: cloud Fire store and real-time database.

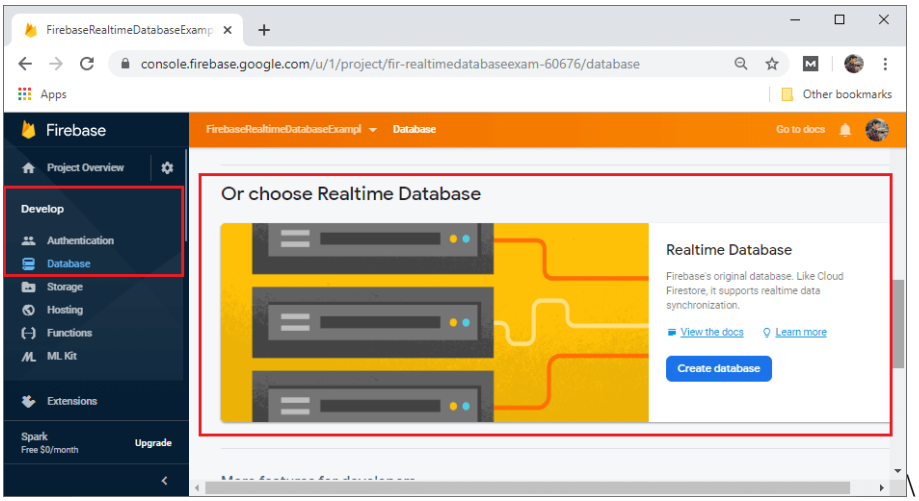


FIGURE 25

- By selecting build database in the following step, we will build a database. A popup box appears after clicking Establish a database, allowing us to establish a database with precise rules. These guidelines will be discussed later in this section. However, for the time being, we'll choose to start in test mode, where anyone can access our data, and we'll adjust these rules later. Finally, we chose Enable.

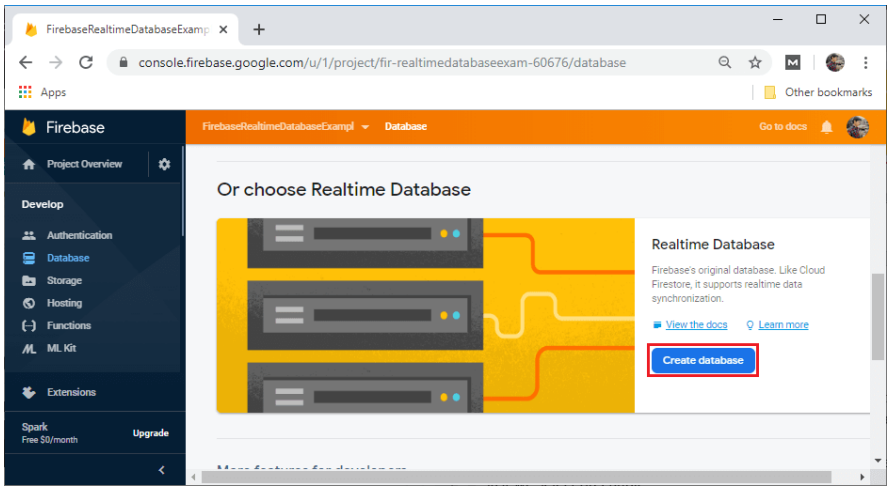


FIGURE 26

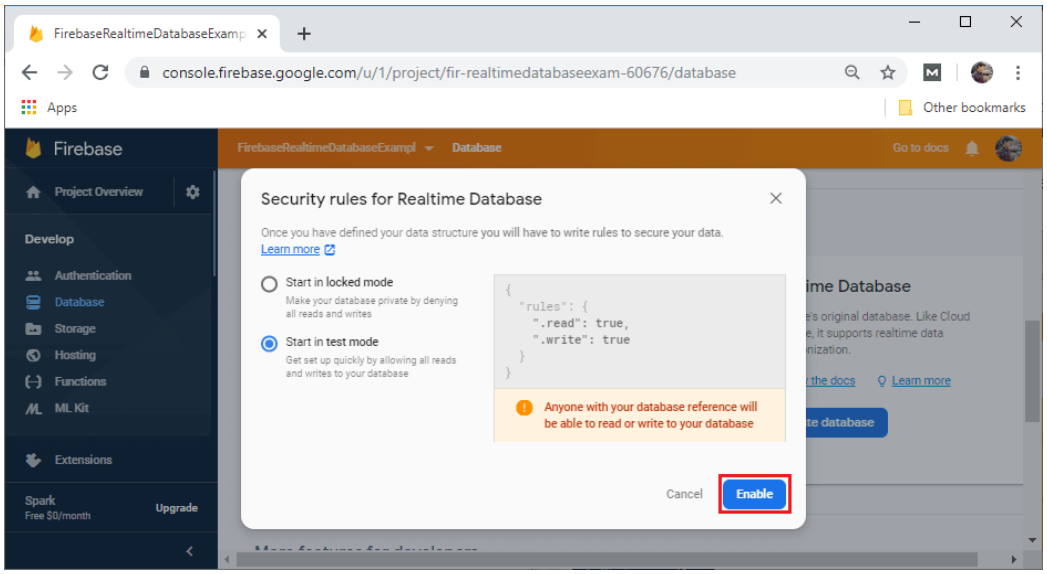


FIGURE 27

- The real-time database will automatically be enabled with a database after selecting Enable. For storing data, security rules, backups, and usage, respectively, we have Data, Rules, Backups, and Usage.

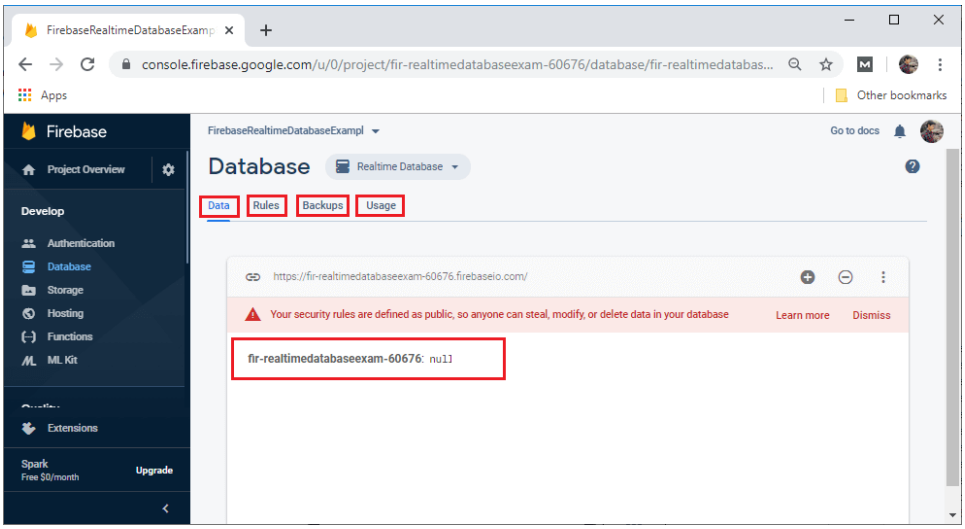


FIGURE 28

- We'll discuss the Firebase Database Rules before moving on to the subsequent phases.
- A declarative rules language is offered by the real-time database. It specifies when our data can be read from and written to, how it should be indexed, and how it should be formatted. Only authenticated users are permitted to read or write data in our database since read and write access is restricted by default.
- We can set up our rules for public access to get going without setting up authentication. Due to these rules, anyone can read and write to our database, including those who aren't using our app.
- The following rules will be used if we wish to allow authenticated users to read and write to our database:
- This will guarantee that nobody other than users who have successfully authenticated via Firebase can read or write to our database.
- The following steps will include accessing the console, going to the database rules, and modifying these rules to apply to authenticated users.

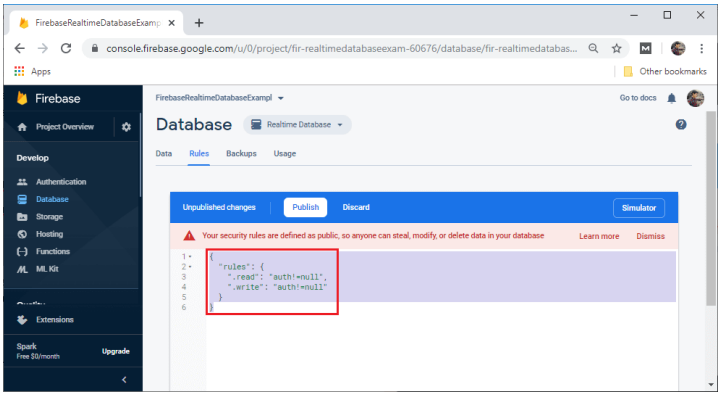


FIGURE 29

- We will publish the rules after the necessary revisions have been made.

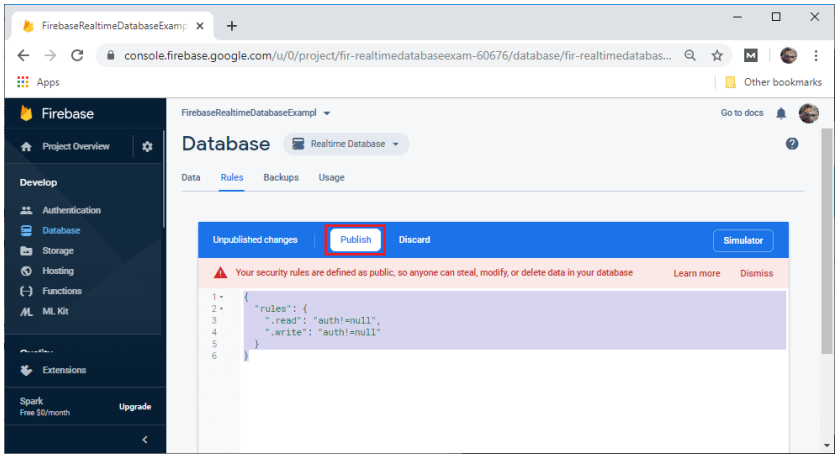


FIGURE 30

- Now that our database has been configured with certain rules, we can use it. We will learn how to do read and write operations in a real-time database in the following section.

CHAPTER-4 EXPERIMENTS & RESULT ANALYSIS

4.1 FUNCTIONAL PRECONDITIONS

4.1.1 User Authentication

- A user must have a working phone number to register for the application.
- The user must be requested to enter their phone number when installing the application.
- If the user doesn't complete it, the application should end.
- The user's phone number will serve as the account's unique identifier on the chat application.

4.1.2 New Contacts Adding

- The programmer ought to recognize every contact in the user's phonebook. The contacts must be automatically added to the user’s contact list on Chat Application if any of the contacts have user accounts there.
- If any of the contacts haven't signed up for Chat Application yet, the user should have the ability to invite them by sending them a regular text message inviting them to do so with a link to the Chat Application on the Google Play Store.

4.1.3 Message Sender

- Any contact on the user's Chat Application contact list should be able to receive instant messages from the user.
- By showing a tick next to the message sent, the user should be informed when the communication is successfully delivered to the intended recipient.

4.1.4 Publicity Message

- Groups of contacts should be able to be created by the user. To these groups, users ought to be allowed to broadcast messages.

4.1.5 Status of Message

- The user must be able to learn if the message they sent was read by the intended recipient. Two ticks must appear next to the message read to indicate that the receiver has read it.

4.2 NON-FUNCTIONAL PRECONDITIONS

4.2.1 Privacy

- To guarantee privacy, exchanged messages between users should be encrypted.

4.2.2 Robustness

- To enable recovery if a user's device malfunctions, a backup of the user's chat history must be kept on distant database servers.

4.2.3 Performance

- The application must convey messages immediately and be small.

4.3 USE CASE DIAGRAM

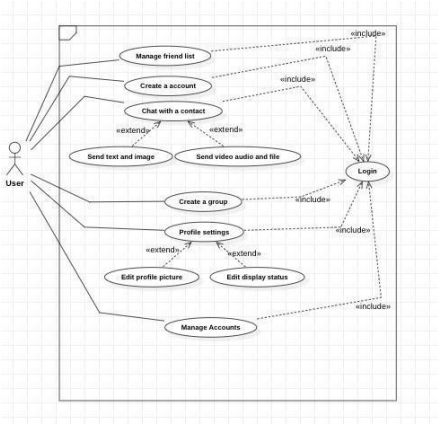


FIGURE 31

4.4 AUTHENTICATION SYSTEM

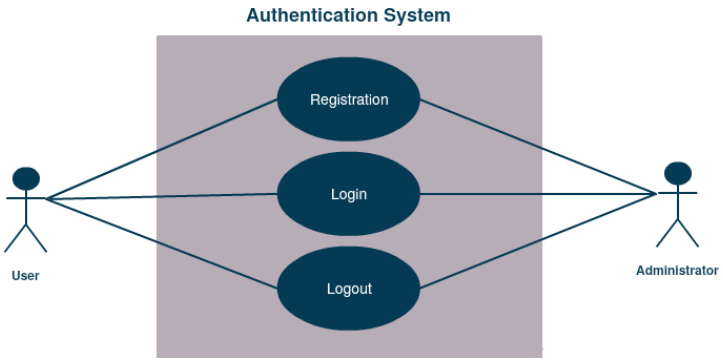
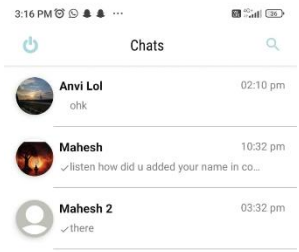


FIGURE 32

4.5 CONTACTS FORM



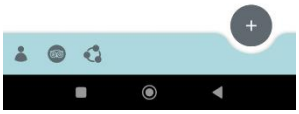


FIGURE 3

4.6 ACTIVITY DIAGRAM

- The Android Manifest is an XML file that gives the operating system crucial details about an Android application. It contains details on the package name, permissions, functions, services, broadcasters, and content suppliers of the application. This post will go over how to use Kotlin to build and edit the Android Manifest file.

- Kot lin’ Android Manifest File Creation
- You must perform the following actions in Kotlin to produce an Android manifest file:
- Activate the Android Studio project.
- Navigate to your project directory's "app" folder.
- The "app" folder should be right-clicked, then choose New > Android Resource File.
- Give the file a name in the New Resource File dialogue, such as AndroidManifest.xml.
- Choose "Manifest" from the Resource type drop-down menu.
- To create, click the "Create" button.

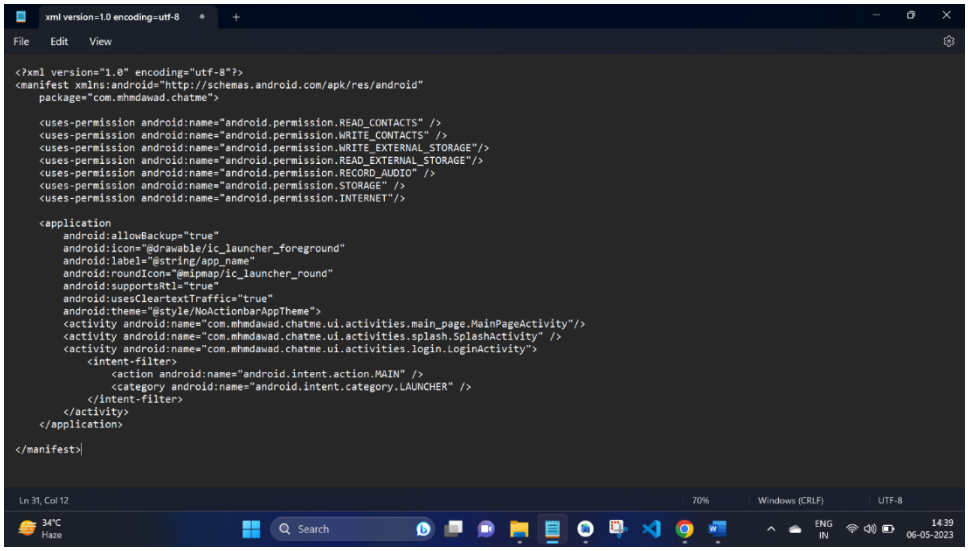


FIGURE 34

- Build configurations and dependencies for an Android app project are specified in the Build. The Gradle configuration file is part of the Gradle build system. We'll go over how to set up the build. Gradle file for an Android app in this article.
- Android Studio's Build. Gradle File Creation
- Follow these steps to generate a build. Gradle file in Android Studio:
- Activate the Android Studio project.
- Go to the "app" folder in the project tree.
- Choose "New > File" by performing a right-click on the "app" folder.
- Click "OK" after naming the file "build. Gradle".

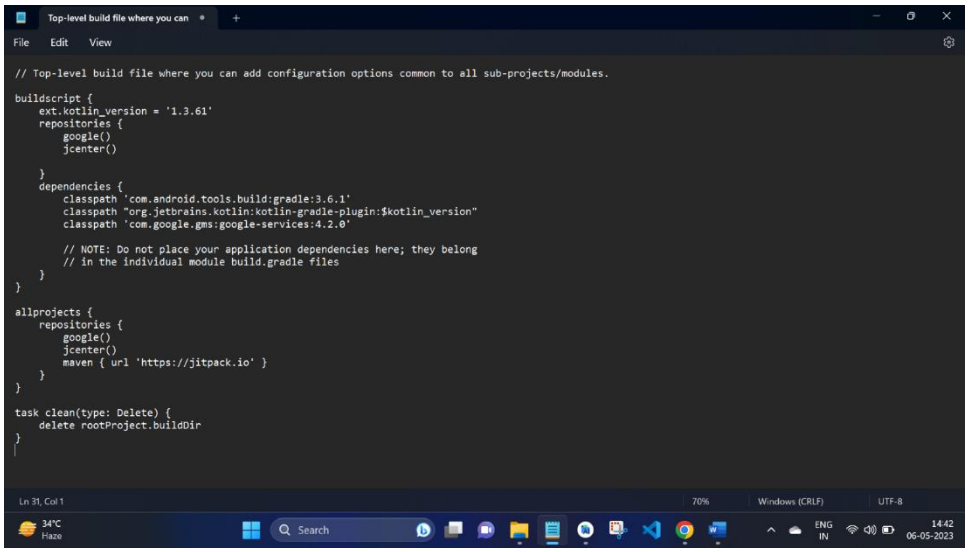


FIGURE 35

- A strong build tool for software projects is called Gradle. Project configurations and dependencies are defined using the DSL named Groovy. We'll talk about using Gradle to build a project in this tutorial.
- How to Create a Gradle Project
- These steps can be used to create a Gradle project:
- Launch your favorite IDE, such as IntelliJ IDEA or Android Studio.
- Choose Gradle as the build system when starting a new project.
- Select the project type (such as Android, Java, or Kotlin).
- Set up the project parameters, including the project name and location.
- Once your project has been created, a "build. Gradle" file may be found in the root directory. The project configuration parameters, such as dependencies and build settings, are contained in this file.

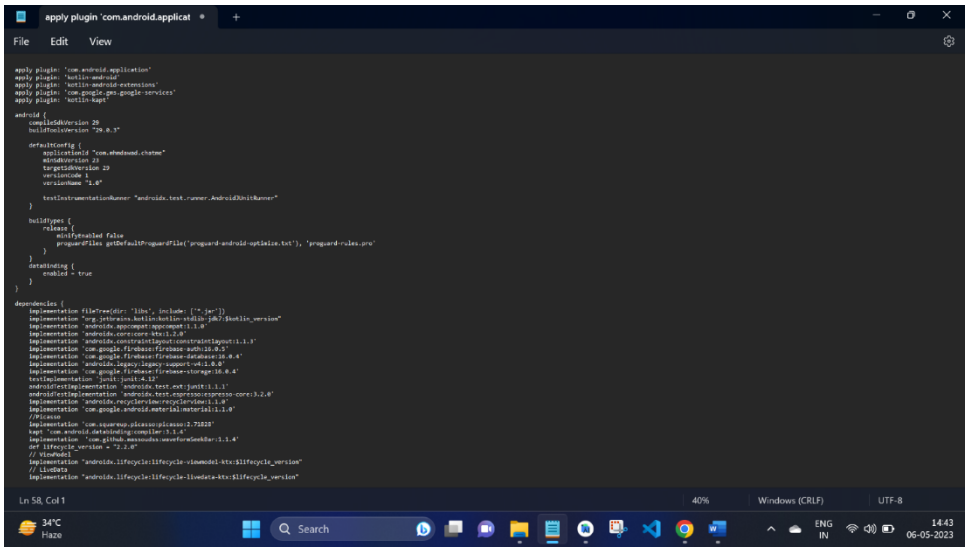


FIGURE 36

4.7 USER GUIDE

4.7.1 Signing Up as a Member

- A common feature in many chats software is the ability for users to sign in using their phone numbers Users that do not wish to memorize numerous usernames and passwords will find this authentication approach to be convenient. We'll go over how to sign up for a chat application with a phone number in this article.
- To sign into a chat programmer using your phone number, follow these steps:
- Launch the chat programmer: Launch the chat programmer on your phone.
- The sign-in process can be started by clicking the "Sign In" button.
- "Sign In with Phone Number" should be chosen: Choose the "Sign in with phone number" option.
- Enter Phone Number: In the space provided, enter your phone number. Please double-check your country code and phone number entries.
- You will be given a ver
- verification code through SMS or phone call to confirm your phone number. In the space provided, type the verification code.
- Set Up Profile: You will be prompted to do so after your identity has been verified. Your name, profile photo, and status update are all editable.
- Start Making Use of the Chat Application: After creating your profile, you may begin making use of the chat application.

4.7.1.1 GUIDELINES TO REMEMBER:

- Verify Phone Number: Before using the chat programmer, make sure to confirm your phone number. This makes it easier to guarantee the safety of your account.

- Use a Strong Password: Use a strong and distinct password if the chat application asks for one in addition to your phone number.
- Keep Your Phone Number Current: Ensure that your phone number is current. If you misplace your phone number, this aids in account recovery.
    - Enable Two-Factor Authentication: If you want more security, think about enabling two-factor authentication. This reduces the likelihood of someone breaking into your account

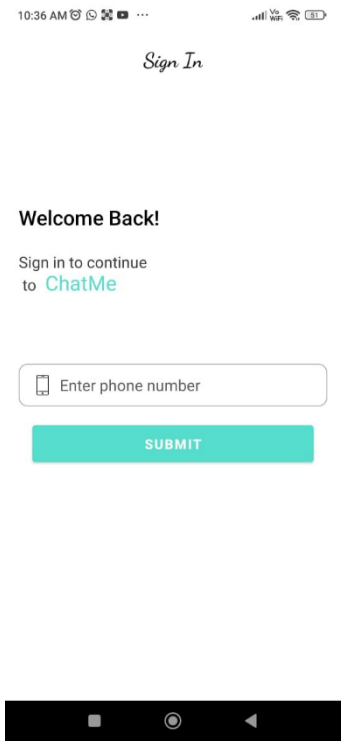


FIGURE 37

4.7.2 Login Options

In chat applications, verification codes are used to validate the user's phone number during the registration or sign-in procedure. They are used to confirm that the user has access to the phone number they provided and are often issued via SMS or phone call. The significance of verification codes in chat programmers and their operation will be covered in this article.

4.7.2.1 Verification codes in chat applications are important:

In chat applications, verification codes are crucial since they boost user account security. Chat programmers can stop fraudulent accounts and unauthorized access by confirming the user's phone number. Verification codes aid in making sure users are entering their phone numbers and not those of third parties.

4.7.2.2 Verification Codes' Operation:

The chat programmer calls or SMSs a verification code to the phone number provided when a user registers or logs in using their phone number. The user then enters the code in the chat application's designated field. The user is validated and may continue the sign-up or sign-in process if the code is accurate.

- Best practices for entering verification codes in chat programmers are listed below:
- Use challenging verification codes: Be sure to use challenging verification codes. This lessens the likelihood of user accounts being accessed improperly.
- Use Two-Factor Authentication: For additional security, think about implementing two-factor authentication in addition to verification codes. Users utilizing two-factor authentication must submit an additional form of identification, such as a password or biometric authentication.
- Inform Users of Code Expiration: Inform users of the verification code's expiration time. Users who might obtain a code that has expired won't be confused or frustrated because of this.
- Users should be given the option to resend verification codes if they did not receive them the first time or if they became invalid. This makes it possible for users to confirm their phone number and continue the sign-up or sign-in procedure.

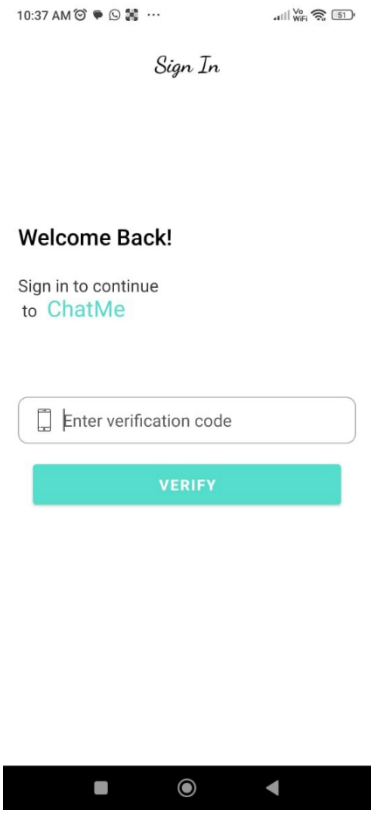


FIGURE 38

4.7.3 Private Messaging

4.7.3.1 Chatting privately in chat programmers:

Users of chat applications can communicate privately with one another thanks to a key feature called private talking. Depending on the architecture of the application, the need for scalability, and the user experience, the private chatting feature can be implemented in a variety of ways. Here are a few typical methods for implementing private chat:

- Peer-to-peer Chat: Using this method, private chat messages are exchanged directly between the two conversation participants. No server or middleman is used. For small-scale chat systems with a modest user base, this approach works best.
- Client-Server Chat: Using this method, the chat messages are transmitted over the server of the chat programmer. The server serves as a go-between for the users and makes sure that the
- Messages are reliably and securely sent. This method works best for chat programmers with lots of users.

4.7.3.2 Best Practices to Implement:

- End-to-end encryption is a crucial component of any chat application that uses private messaging. It makes sure that communications are encrypted at the sending device and that only the intended recipient can decrypt them. As a result, the messages can't be intercepted by anybody else, adding another layer of protection.
- Message deletion: Users should have the option to remove messages they no longer wish to keep from the private messaging tool. This functionality must function in a way that guarantees that the deleted messages are completely erased from the devices of the sender and recipient as well as the server hosting the chat application.
- Private messaging should provide seen and delivered indicators that help users know when their messages have been delivered.
- By letting users know when their communication has been received and read, this feature lowers the likelihood of confusion.
- Control of Notifications: Users should be able to modify the chat application's notification settings for private talks. The user needs to be able to block alerts altogether, mute notifications for a predetermined amount of time, or turn off notifications for private talks.

In conclusion, private messaging is a crucial function of chat programmers that enables users to converse secretively and securely. To make sure that this feature is secure, dependable, and user-friendly, careful preparation and attention to detail are needed while implementing it. Developers of chat applications can add a private chatting function that improves user experience and boosts the overall success of the programmer by

using the best practices described in this article.

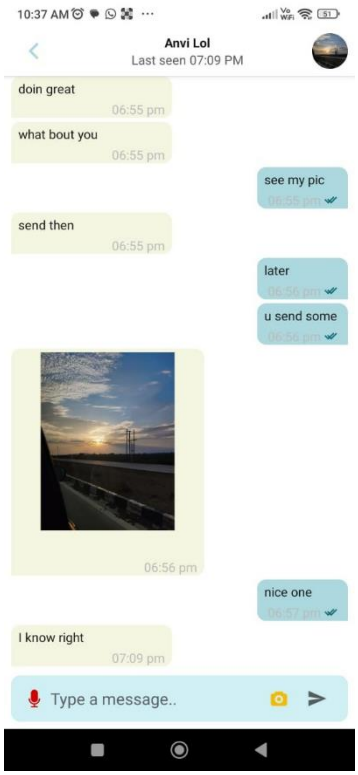


FIGURE 39

4.7.4 Making a Friend

A frequent feature that enables users to connect with their friends and speak with them on the platform is the ability to add friends in a chat application. We'll talk about how to add friends in a chat programmer in this article.

To add a buddy to a chat application, follow these steps:

- Using their username, email address, or phone number, the user should be able to search for their friends on the chat programmer.

- Send a friend request: The user ought to be able to send their acquaintance a friend request once they have located them. A message or a note introducing the user to their friend should be included in the friend request.
- When a user sends a friend request to a friend, they should wait for the friend to accept the request. When a user's friend accepts a request, the chat programmer should let them know.
- Start chatting: After the friend request has been approved, the user ought to be allowed to begin their chat session with their friend.

4.7.4.1 Guidelines to Remember:

- User Privacy: Users should be able to manage their privacy settings and decide who can add them as friends in the chat application. Users can utilize this function to preserve their privacy and make sure that only friends they know or want to connect with send them friend invitations.
- Mutual links: The chat programmer should provide links between users that they have in common, such as friends, affinity groups, or interests. With the use of this tool, users can locate and connect with individuals that they might know or have similar interests.
- User verification is necessary for the chat programmer to protect against phone or spam accounts. Verification options include social media accounts, email addresses, and phone numbers.
- The chat programmer can recommend new friends to the user based on their shared hobbies, locations, or relationships. Users can utilize this function to locate and connect with people they may not know but who could wind up being close friends.

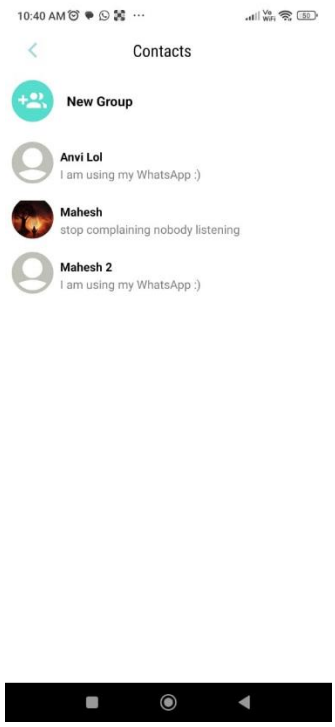


FIGURE 40

4.7.5 A Chat Forms

An integral component of a chat program that enables users to message their friends or groups is a chat form. The essential elements of a conversation form in a chat application will be covered in this article.

The main elements of a conversation form in a chat application are as follows:

- Text Input Field: The user enters their message in this field. Users should be able to input messages in the chat form using their device's keyboard. To make typing easier for the user, the text input field may also have options like auto-correction, spell-checking, and suggestions.

- Use the send button to deliver your message to the intended recipient. The user should be able to send messages by clicking the send button on the chat form. To prevent sending empty messages, the send button should only be activated when there is text entered into the input area.
- Attachments: Users of the chat form may also choose to add documents, images, or videos to their messages. Users can share media with their friends or a group using this tool. A button or symbol in the conversation form can be clicked to access the attachment capability.
- Emojis and Stickers: Adding emojis and stickers to a message is another option available on the chat form. Emojis and stickers allow users to convey their feelings and give the message a more unique touch.
- Recording of Voice and Video Messages: The chat form may also include the capability of recording voice and video messages. Users can send messages without typing thanks to this feature. By clicking on an icon or button, you can access the voice and video recording capability.

4.7.5.1 Guidelines to Remember:

Design with the user in mind: The chat form should be user-friendly. It should be simple to access and use the text input box, send button, attachments, and other functions.

- Consistency: The chat form's design should be the same throughout the entire application. Every conversation should have a consistent layout and operation of the text input field, send button, and other elements.
- Error handling should be done correctly in the chat form. If there are any errors, such as a failed message sending, the user should be alerted so they can try again.

- Accessibility: People with impairments should be able to utilize the chat form.

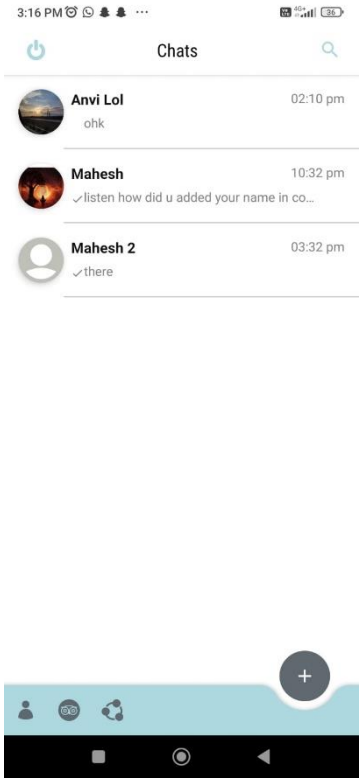


FIGURE 41

4.7.6 Form a group

An advantageous feature that enables users to talk with numerous individuals simultaneously in a chat programmer is group creation. We will go over how to form a group in a chat application in this article.

To form a group in a chat application, follow these steps:

- Launch the chat programmer: Launch the chat programmer on your phone.
- Go to the group area by navigating to it within the chat application. This is typically located in the sidebar or the main menu.
- Click "Create Group" to begin: To begin creating a new group, click the "Create Group" button.
- Select Your Group Name and Profile Image: Give your group a name. You may also include a profile photo.

- Group Members to Add: Choose the people you want to add to the group of contacts. Multiple contacts can be added to the group.
- When you have finished adding all the members to the group, click "Create" to start the creation of the group.
- Start a Conversation: After the group has been created, you can start a conversation with the other members. Each group member can message, post images, and upload videos to the group.
-

4.7.6.1 Guidelines to Remember:

- Group Naming: Give your group a memorable name that has meaning. The group's mission should be reflected in the name.
- Adding Group Members: Ensure that the group contains all necessary connections. Later, you can extend invitations to more group members.
- Set the group settings to your preferences in the group settings section. You have control over several things, including who can join the group and view the messages.
- Admin Controls: As the group's administrator, you have control over the group's settings and members. You can modify the group's settings, remove members, and more.

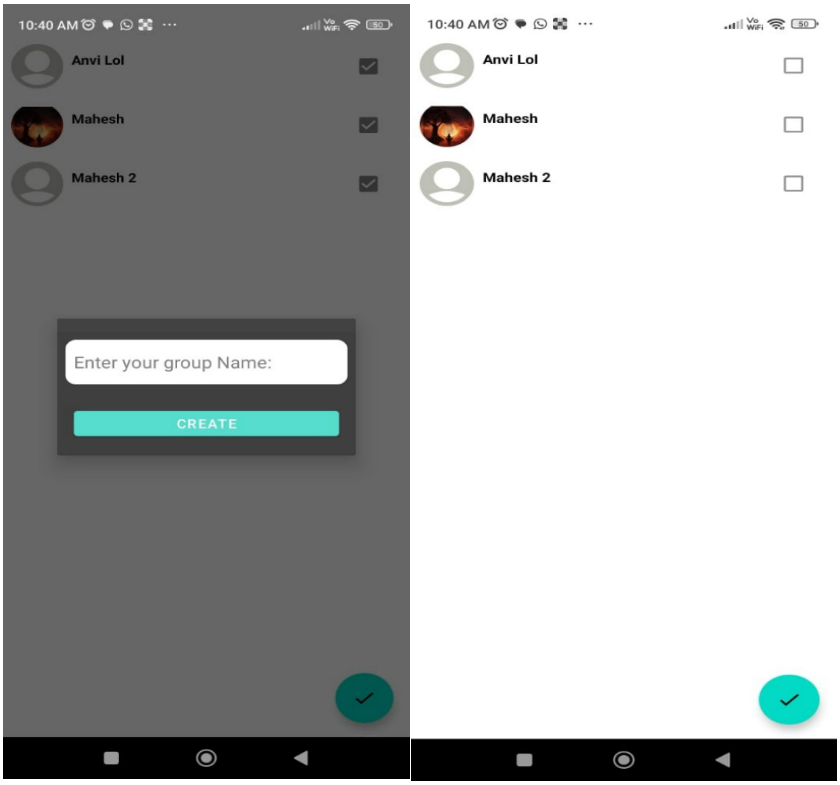


FIGURE 42

4.7.7 Account Preferences

Any chat programmer should have profile settings since they let users manage their accounts and customize their profiles. We will go through how to access and manage profile settings in a chat programmer in this article.

In a chat programmer, use these steps to access and manage profile settings:

- Launch the chat programmer: Launch the chat programmer on your phone.
- Go to the profile area by navigating to it within the chat application. This is typically located in the sidebar or the main menu.
- select "Settings" To access profile settings, click the "Settings" button. Edit Profile Information: You have the option of changing your name, profile image, status update, and other details.

- Change Account Information: You can modify your account information, including your password and email address.
- Manage Privacy Settings: You have control over your privacy settings, which include things like who can see your profile photo and status update.
- You can control your notification settings, including loudness, vibration, and others.
- Sign Out: By clicking the "Sign Out" button, you can log out of the chat application.

4.7.7.1 Best Practices to Remember:

- Update Your Profile Information routinely: Be careful to routinely update your profile information. This makes it simpler for your contacts to identify you.
- Manage Privacy Settings: Adjust your privacy settings to your tastes. You have control over who can view your profile details, status updates, and other information.
- Manage Notification Settings: Adjust the settings for your notifications to suit your tastes. When and how notifications are delivered is at your discretion.
- Maintain Current Account Information: Ensure that your account information is current. If you forget your password or misplace your email address, this aids in account recovery.

FIGURE 43

CHAPTER-5 CONCLUSIONS

5.1 FINAL THOUGHTS

In conclusion, the chat application was successfully developed utilizing the Firebase real-time database and the Kotlin programming language to accomplish the specified goals. With additional capabilities including group chat, one-on-one chat, and the ability to transmit text and photographs, the programmer offers real-time talking and authentication with Firebase. Data binding and the MVVM architecture have made it possible to implement code in an effective and structured way. The efficiency of the developed system was demonstrated in the experiment and result analysis chapter, where results were compared using various test cases and methods.

5.2 THE FUTURE

There is always an opportunity for improvement and potential future additions, just like with any software development. The implementation of new capabilities like video calling, voice messaging, and the ability to share files is one potential future scope for this chat programmer. The application could also be improved for increased performance and scalability.

5.3 CONTRIBUTIONS FROM APPLICATIONS

The created chat application can be used in a variety of situations where real-time communication is required, such as in professional or social contexts. As it offers a dependable and effective backend solution for real-time data synchronization, the use of Firebase's real-time database and authentication has ramifications for other software projects as well.

In conclusion, the chat application that was created has shown off the possibilities of the Firebase real-time database and the Kotlin programming language, and it has the potential to be improved upon in the future and contribute to the software development field.

6-REFERENCES

https://console.firebase.google.com/u/0/

https://codingwitht.com/firebase-phone-authentication-android-tutorial-2020/

https://portal.bazeuniversity.edu.ng/student/assets/thesis/20210215120658149063642.pdf

https://gscen.shikshamandal.org/wp-content/uploads/2022/sp/BCCA20-21/22.pdf

https://image.slidesharecdn.com/finaldoconchatapp-170418182505/75/chat-application-full-documentation-6-2048.jpg?cb=1665822116

https://www.slideshare.net/MuhammadAshiqurRahma/chat-application-full-documentation

https://digitalcommons.harrisburgu.edu/cgi/viewcontent.cgi?article=1009&context=csms_student-coursework

https://www.studocu.com/in/document/chitkara-university/artificial-intelligence/project-report-upload/27371355

https://www.academia.edu/40977586/WEB_BASED_CHAT_APPLICATION_MINOR_PROJECT_REPORT_COMPUTER_SCIENCE_and_ENGINEERING

https://objectpartners.com/2017/09/06/real-time-chat-application-with-kotlin-and-firebase/

https://blog.logrocket.com/how-to-build-chat-application-flutter-firebase/