

CHATCONNECT-A REAL TIME CHAT AND COMMUNICATION APP

1. INTRODUCTION

1.1 OVERVIEW

In modern era, communication is a crucial part of every day life. People have been interacting with one another through a variety of media since the beginning of time. The distance between humans grew as new regions were discovered, and communication methods evolved. People utilised a variety of methods to communicate with one another, including letters and telegrams, but the main disadvantage of most of them was the length of time it took for them to arrive. Letters could take days to arrive, which was a significant problem with communication at the time.

The introduction of cell phones made the developers sought to construct a text-based messaging service that would enable instant communication capabilities. However, the primary constraint in the case of the novel notion was the small amount of message writing space, primarily 128 bytes. Many enhancements were made after the initial conception of the idea, and the first SMS was delivered in 1992. The next year, Aldiscon and Telia in Sweden launched the first commercial SMS service. Although SMS was the primary means of communication in the 2000s, its high costs were a drawback. Still, many individuals utilised it to converse quickly or to send any urgent messages. In the late 2000s, as the popularity of smartphones increased, a broad variety of messaging programmes based on various operating systems were made available and quickly gained widespread acceptance. The most well-known of these were Snapchat, WeChat, Viber, WhatsApp, and a few more.

Firebase is a framework which is useful for building portable and web applications for businesses which require real-time database which implies when one user updates a record in the database, the update should be conveyed to every single user instantly. Together with a variety of other Google services integrated into the service, it provides a simple and consistent foundation for various apps. When it comes to the creation of applications, the majority of the server-side work is handled by Firebase. From the perspective of a developer, there are many factors that make Firebase such a crucial tool in development. By only slightly delaying work, it maintains a state of harmony between the developer and the client. In case of the development of a communication or chat application, the most essential components or services offered by Firebase are: Real-time Database, Authentication, Storage, Cloud Messaging.

1.1.1 REAL-TIME DATABASE

Real-time Database is a cloud-hosted database. Data is stored as JSON and synchronized continuously to each associated client. The majority of user demand for any cross-platform application designed with the iOS, Android, and JavaScript SDKs is based on one Real-time database instance, and this instance gets updated with every fresh data. Since Firebase manages the majority of the backend for the applications, this functionality enables developers to forego the step of creating a database.

1.1.2 AUTHENTICATION

Firebase Authentication offers backend services, simple-to-use SDKs, and instant UI libraries to verify clients over an application. It supports passwords, email addresses or usernames, phone numbers, etc. as forms of authentication. Either by using FirebaseUI as a comprehensive drop-in authentication solution or by manually integrating one or more sign-in methods using the SDK, users can sign in to a Firebase app.

1.1.3 STORAGE

Firebase Storage is a cloud-based storage solution provided by Google's Firebase platform. It allows developers to store and serve user-generated content such as images, videos, and other files securely in the cloud. Firebase Storage offers a scalable and cost-effective way to store and share files across multiple devices and platforms.

Firebase Storage provides various features, including:

- ❖ Secure and scalable storage
- ❖ Cross-Platform Compatibility
- ❖ Easy Integration
- ❖ Flexible Pricing

Secure and scalable storage:

Firebase Storage offers secure and scalable storage for user-generated content. It ensures that data is encrypted and protected by Google's security infrastructure, making it a reliable and secure storage option.

Cross-Platform Compatibility:

Firebase Storage is designed to work seamlessly with various platforms, including Android, iOS, and web applications. This makes it easy for developers to create apps that can be used across different platforms.

Easy Integration :

Firebase Storage can be easily integrated into any Firebase project, making it easy for developers to use it for storing and sharing user-generated content. In the Firebase console, you can set up rules for your storage bucket, such as who can access it and what kind of data can be stored. The Firebase Storage SDK to upload and download files from your storage bucket.

Flexible Pricing:

Firestore offers flexible pricing plans that allow developers to pay only for the storage they use. This makes it a cost-effective solution for storing and sharing user-generated content.

1.1.4 CLOUD MESSAGING (FCM)

Cloud Messaging (FCM) allows developers to send notifications and messages to their users across various platforms, including Android, iOS, and web applications. FCM is a reliable and scalable messaging solution that can handle large-scale messaging requirements with ease.

Real-Time Messaging: FCM provides real-time messaging capabilities, allowing developers to send notifications and messages to their users instantly.

Reliable and Scalable: FCM is a reliable and scalable messaging solution that can handle large-scale messaging requirements with ease.

Analytics and Reporting: FCM provides detailed analytics and reporting features that allow developers to monitor the performance of their messaging campaigns and track user engagement.

Customizable Notifications: FCM allows developers to create custom notifications and messages that can be tailored to the specific needs of their users.

1.2 PURPOSE

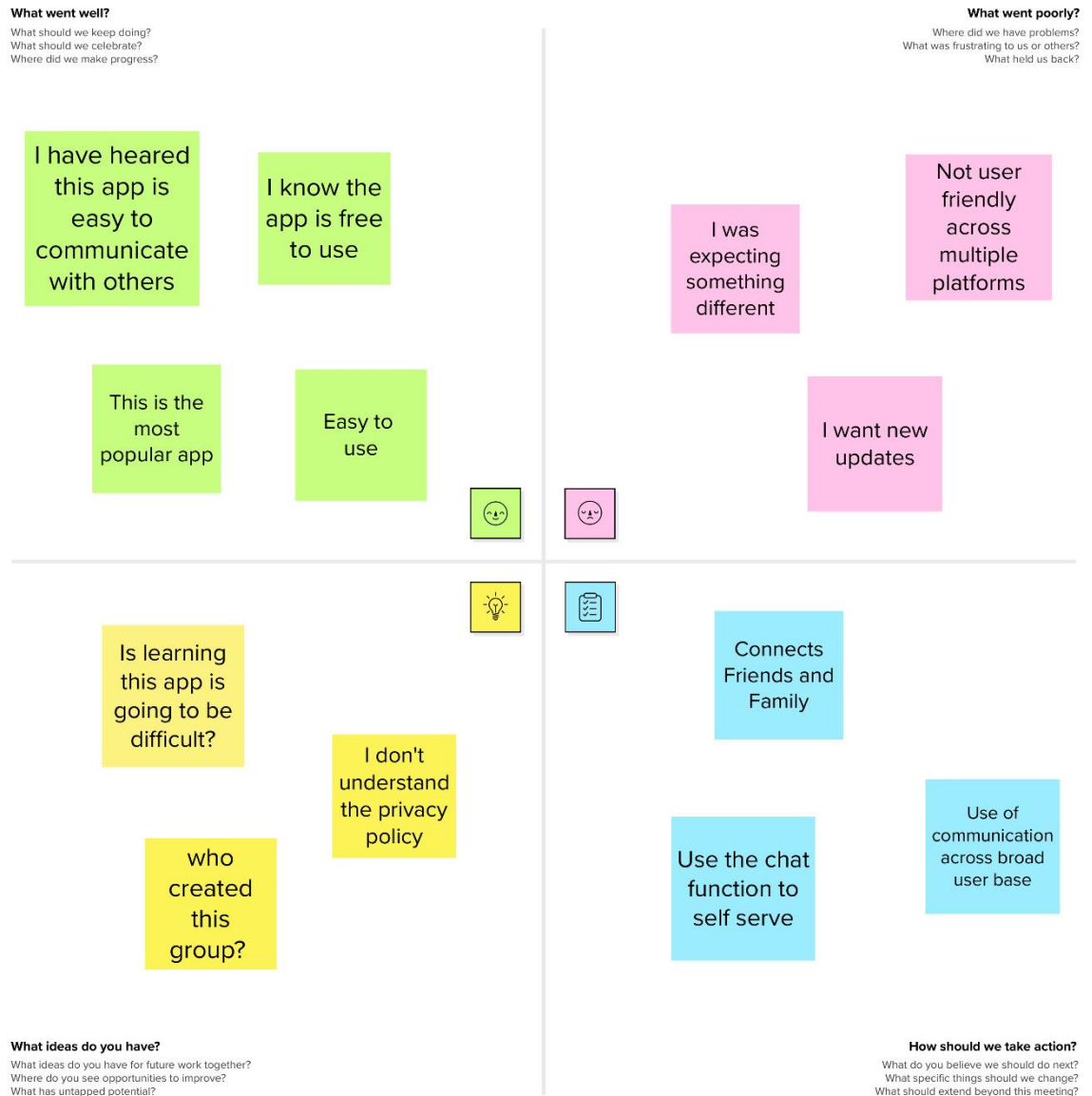
Internet-based communication is becoming increasingly important and relevant these days, keeping this communication real-time is essential as our lives have become more fast paced. Users of online communication can connect with others in a quick and appropriate way. Considering this, A communication application should be developed that could be to send files and messages instantaneously or with little to no delay. A real-time updating database that keeps

track of all the data being exchanged is required for such a system to function. Such a real-time database server is available through the Google Firebase service, which also offers a variety of other features. Firebase makes it relatively simple to create communication-based applications.

A Chatconnect application is introduced that will enable real-time text messaging between two users on a network, as well as the transfer of assets including photographs, audio, and videos. To handle the backend of the communication operation, the Android operating system is used and Google Firebase, showcasing the numerous functionalities of both the operating system and the service. It also showcases the use of Compose's declarative UI and state management capabilities. It also includes examples of how to handle input and navigation using composable functions and how to use data from a firebase to populate the UI.

2.PROBLEM DEFINITION AND DESIGN THINKING

2.1 EMPATHY MAP



2.2 BRAINSTROMING MAP

Berjin

Voice calls	Files and image sharing	Real -time Events
Quick registration using the number of mobile phones	Video calling	Use of Chat reactions
Functions of setting Reminders for plans	Real time location sharing	GIFs and Sticker sharing

Bobin sundar

Delete read messages and conversations	create communities	Share picture, music videos and files
Multimedia messages	Scheduled and silent messages	synchronize histroy and transfer calls
Chat bots	Messages broadcasting	Encryption

Auxlin renix

Push notifications	cloud synchronization	Multi linguistic support
Geolocation	One-to-one and group chat	Import contacts
Multimedia support	Chat analytics	Privacy settings

Beljin

Language translation	security	Message reactions
Theme management	onference call	Chat bubble
Storage capacity	User presence	Payment options

3.RESULTS

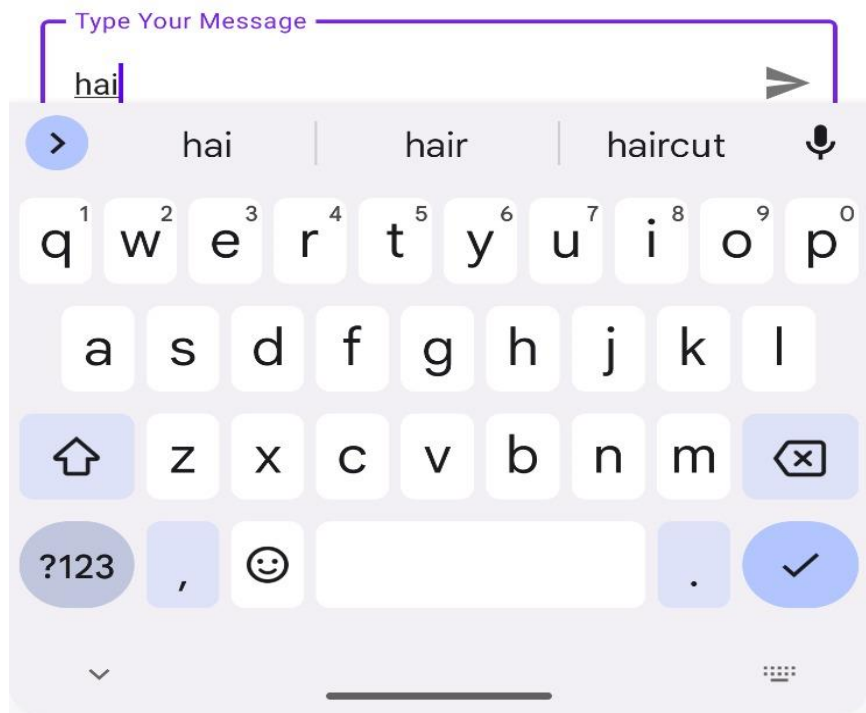
3.1 AUTHENTICATION OPTION



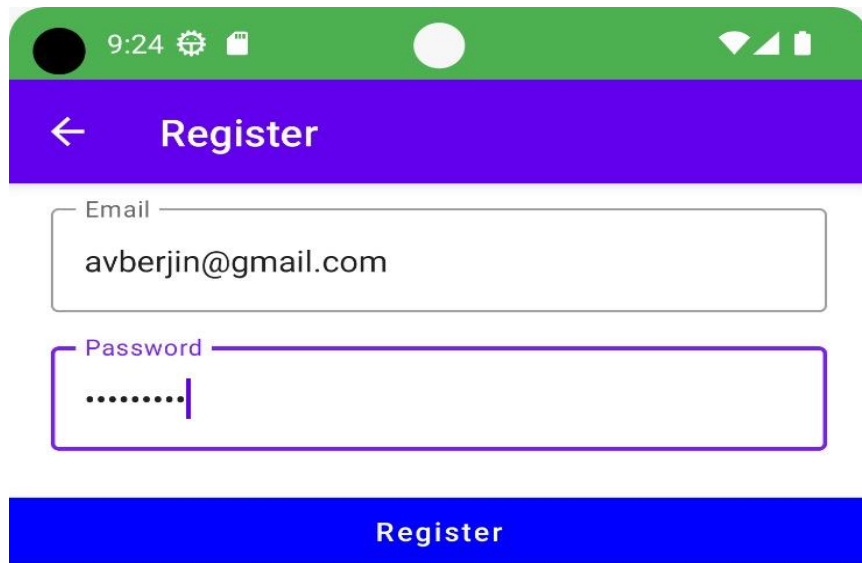
Air Bird 🦜



3.2 REGISTER



3.3 HOME SCREEN



A mobile app mockup for a registration screen. The top status bar is green and shows the time 9:24, a gear icon, a battery icon, and signal strength. Below this is a purple header bar with a white back arrow and the text "Register". The main area is white and contains two input fields. The first field is labeled "Email" and contains the text "avberjin@gmail.com". The second field is labeled "Password" and contains seven dots. Below the input fields is a blue button with the text "Register".

9:24

← Register

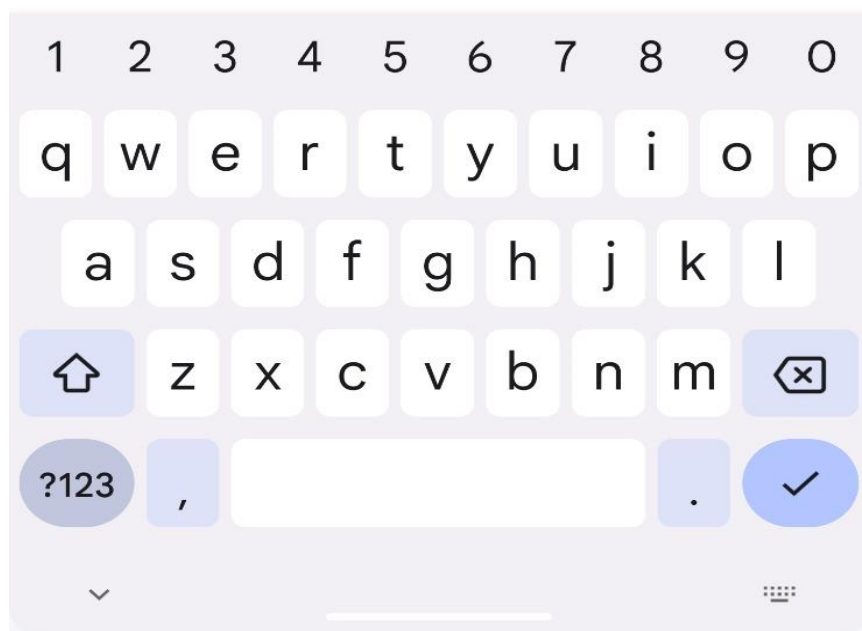
Email

avberjin@gmail.com

Password

.....

Register



4.ADVANTAGES AND DISADVANTAGES

4.1 ADVANTAGES

A real-time chat application is a popular feature in many mobile applications, including Android apps. Google Firebase is a widely used platform for developing chat applications on the Android platform due to its real-time database and other features. Some of the advantages are:

Real-time updates:

Firebase provides real-time updates, allowing users to receive messages instantly as they are sent. This feature is essential for a chat application where users need to receive messages quickly and efficiently.

Scalability:

Firebase is designed to handle large amounts of data, making it suitable for chat applications with a high volume of users. As your chat application grows, Firebase can easily scale up to accommodate the increased traffic, ensuring that users continue to receive fast and responsive service.

Offline support:

Firebase provides offline support, allowing users to send and receive messages even when they are not connected to the internet. This feature ensures that users can stay connected to the chat application even in low connectivity areas, such as subways or remote areas.

Cross-platform compatibility:

Firebase supports multiple platforms, including Android, iOS, and web, making it easy to create a chat application that can be accessed from any device. This feature is particularly useful if you plan to develop your chat application for

multiple platforms, ensuring that users can stay connected regardless of the device they are using.

Security:

Firebase provides secure user authentication and data storage, ensuring that user data is protected. This feature is essential for any chat application, as user privacy and security are critical.

Analytics and Monitoring:

Firebase provides robust analytics and monitoring tools that help developers understand user behavior and performance issues. This feature is particularly useful for chat applications, as it allows developers to track engagement levels and ensure that the application is performing optimally.

4.2 DISADVANTAGES

Google Firebase is a popular choice for developing real-time chat applications on the Android platform. It offers several advantages such as real-time updates, scalability, offline support, cross-platform compatibility, and security. However, it also has some disadvantages such as a steep learning curve, cost, limited features, and vendor lock-in.

Learning curve:

Firebase can have a steep learning curve for developers who are new to the platform. If not familiar with Firebase, it may need to spend some time learning how to use it effectively. Additionally, Firebase has a unique data model, which may require some adjustment if you are used to working with traditional SQL databases.

Cost:

While Firebase has a free plan, it can become costly as the usage and the number of users grow. Firebase charges based on the amount of data stored, the number of users, and the amount of data transferred. As you scale your chat application, you may need to upgrade to a paid plan to meet your needs.

Limited features:

Firebase has a limited set of features compared to other chat application platforms, which may be a disadvantage for more complex applications. For example, Firebase's real-time database is designed for simple data structures and may not be suitable for more complex chat applications.

Vendor lock-in:

Developing an application with Firebase can create a dependency on the platform, which can be difficult to move away from if needed. If you ever decide to move your application to a different platform, it may require significant effort and resources.

Performance issues:

While Firebase is designed to handle large amounts of data, it may not be suitable for extremely high-traffic chat applications. If your application experiences high levels of traffic, you may need to consider using a different platform that can handle the load more efficiently.

5. APPLICATIONS

The chatconnect application in Google Firebase Android platform has numerous applications, ranging from social networking to customer support. Below are some of the common applications of the chatconnect application in Google Firebase Android platform:

Social Networking:

The chatconnect application can be integrated into social networking platforms, allowing users to engage with one another in real-time. Users can share messages, images, and videos, as well as create groups and communities. Examples of social networking applications that use Firebase's real-time chat feature include WhatsApp, Telegram, and Instagram.

Customer Support:

The chatconnect application can be used to provide customer support for businesses. Customers can engage with customer support representatives in real-time, allowing them to get help quickly and efficiently. This feature is particularly useful for businesses that operate in multiple time zones and need to provide support to customers around the clock.

Collaboration:

The chatconnect application can be used for collaboration, allowing team members to work together in real-time. Users can share files, messages, and other information, ensuring that everyone is up to date with the latest information. This feature is particularly useful for remote teams that need to collaborate on projects.

E-commerce:

The Real-time chat applications can be used in e-commerce platforms to provide real-time customer support, allowing customers to engage with representatives and get help with their orders. Additionally, real-time chat can be

used to send order updates and notifications, providing customers with timely information about their orders.

Gaming:

The chatconnect application allow players to communicate with each other in real-time. Players can send messages, images, and videos, as well as create groups and communities. This feature is particularly useful for multiplayer games, as it allows players to communicate with each other and coordinate their gameplay.

Education:

Real-time chat applications can be used in educational applications to allow students to communicate with each other and with their teachers in real-time. Students can ask questions, share information, and collaborate on projects, ensuring that everyone is up to date with the latest information. This feature is particularly useful for online learning, as it allows students to engage with each other and with their teachers in real-time.

Healthcare:

The chatconnect application provides a real-time communication between patients and healthcare providers. Patients can engage with healthcare providers in real-time, allowing them to get help quickly and efficiently. Additionally, real-time chat can be used to send notifications and reminders, ensuring that patients stay on top of their healthcare.

6.CONCLUSION

The Chatconnect application in Google Firebase Android platform is a powerful tool that allows user to users to send and receive text messages. Firebase is a step forward in the right direction in the context of application development in the sense that it provides an all round service for developers. It uses Composite declarative UI is a programming paradigm that allows developers to create user interfaces using a declarative language rather than imperative code. Google Firebase provides developers with a range of tools to develop composite declarative UIs, including Firebase Hosting, Firebase Authentication, and Firebase Realtime Database.

The development of communication-based applications is made comparatively easy using Firebase. A system is presented that enables the transfer of assets like images, audio, and movies, as well as real-time text messaging between two users on a network. The Android operating system and Google Firebase are utilised to manage the communication operation's backend, displaying the vast functionalities of both the platform and the service.

7.FUTURE SCOPE

As for the Chatconnect application, it has numerous upgrade paths from implementing call functionality or any other enhanced future works to the flexible nature of the application and the application will continue to upgrade as long as the technology upgrades. So, the application is extremely future-proof.

There is always some place for enhancements in any software application, however good and efficient the application may be. The Chatconnect application is basically instant messaging between the peers. In future the application may further developed to include some features such as

- ❖ Voice messaging
- ❖ Group calling
- ❖ Live streaming
- ❖ Messages auto delete after a given time
- ❖ Personalized message tunes

8.APPENDIX

SOURCE CODE:

Navigation.kt:

```
package com.berjin.flashchat.nav

import androidx.navigation.NavHostController
import com.berjin.flashchat.nav.Destination.Home
import com.berjin.flashchat.nav.Destination.Login
import com.berjin.flashchat.nav.Destination.Register
object Destination {
    const val AuthenticationOption = "authenticationOption"
    const val Register = "register"
    const val Login = "login"
    const val Home = "home"
}
class Action(navController: NavHostController) {
    val home: () -> Unit = {
        navController.navigate(Home) {
            popUpTo(Login) {
                inclusive = true
            }
            popUpTo(Register) {
                inclusive = true
            }
        }
    }
    val login: () -> Unit = { navController.navigate(Login) }
    val register: () -> Unit = { navController.navigate(Register) }
    val navigateBack: () -> Unit = { navController.popBackStack() }
}
```

Home.kt:

```
package com.berjin.flashchat.view.home

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Send
import androidx.compose.runtime.Composable
```

```

import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.berjin.flashchat.Constants
import com.berjin.flashchat.view.SingleMessage

@Composable
fun HomeView(
    homeViewModel: HomeViewModel = viewModel()
) {
    val message: String by homeViewModel.message.observeAsState(initial = "")
    val messages: List<Map<String, Any>> by
homeViewModel.messages.observeAsState(
        initial = emptyList<Map<String, Any>>().toMutableList()
    )

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Bottom
    ) {
        LazyColumn(
            modifier = Modifier
                .fillMaxWidth()
                .weight(weight = 0.85f, fill = true),
            contentPadding = PaddingValues(horizontal = 16.dp, vertical = 8.dp),

```

```

        verticalArrangement = Arrangement.spacedBy(4.dp),
        reverseLayout = true
    ) {
        items(messages) { message ->
            Boolean
                val isCurrentUser = message[Constants.IS_CURRENT_USER] as

                SingleMessage(
                    message = message[Constants.MESSAGE].toString(),
                    isCurrentUser = isCurrentUser
                )
        }
    }

    OutlinedTextField(
        value = message,
        onChange = {
            homeViewModel.updateMessage(it)
        },
        label = {
            Text(
                "Type Your Message"
            )
        },
        maxLines = 1,
        modifier = Modifier
            .padding(horizontal = 15.dp, vertical = 1.dp)
            .fillMaxWidth()
            .weight(weight = 0.09f, fill = true),
        keyboardOptions = KeyboardOptions(
            keyboardType = KeyboardType.Text
        ),
    )

```

```

        singleLine = true,

        trailingIcon = {

            IconButton(

                onClick = {

                    homeViewModel.sendMessage()

                }

            ) {

                Icon(

                    imageVector = Icons.Default.Send,

                    contentDescription = "Send Button"

                )

            }

        }

    )

}

}

```

HomeViewModel:

```

package com.berjin.flashchat.view.home

import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.ktx.auth
import com.google.firebase.firestore.ktx.firestore
import com.google.firebase.ktx.Firebase
import com.berjin.flashchat.Constants
import java.lang.IllegalArgumentException

class HomeViewModel : ViewModel() {

    init {

```

```

        getMessages()

    }

    private val _message = MutableLiveData("")
    val message: LiveData<String> = _message

    private var _messages = MutableLiveData(emptyList<Map<String,
Any>>>().toMutableList())
    val messages: LiveData<MutableList<Map<String, Any>>> = _messages

    fun updateMessage(message: String) {
        _message.value = message
    }

    fun addMessage() {
        val message: String = _message.value ?: throw
IllegalArgumentException("message empty")

        if (message.isNotEmpty()) {
            Firebase.firestore.collection(Constants.MESSAGES).document().set(
                hashMapOf(
                    Constants.MESSAGE to message,
                    Constants.SENT_BY to Firebase.auth.currentUser?.uid,
                    Constants.SENT_ON to System.currentTimeMillis()
                )
            ).addOnSuccessListener {
                _message.value = ""
            }
        }
    }

    private fun getMessages() {

```

```

        Firebase.firestore.collection(Constants.MESSAGES)

            .orderBy(Constants.SENT_ON)

            .addSnapshotListener { value, e ->

                if (e != null) {

                    Log.w(Constants.TAG, "Listen failed.", e)

                    return@addSnapshotListener

                }

                val list = emptyList<Map<String, Any>>().toMutableList()

                if (value != null) {

                    for (doc in value) {

                        val data = doc.data

                        data[Constants.IS_CURRENT_USER] =

                            Firebase.auth.currentUser?.uid.toString() ==
data[Constants.SENT_BY].toString()

                        list.add(data)

                    }

                }

                updateMessages(list)

            }

        }

        private fun updateMessages(list: MutableList<Map<String, Any>>) {

            _messages.value = list.asReversed()

        }

    }
}

```

Login.kt:

```
package com.berjin.flashchat.view.login

import androidx.compose.foundation.layout.*
import androidx.compose.material.CircularProgressIndicator
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.unit.dp
import androidx.lifecycle.viewmodel.compose.viewModel
import com.berjin.flashchat.view.Appbar
import com.berjin.flashchat.view.Buttons
import com.berjin.flashchat.view.TextFormField

@Composable
fun LoginView(
    home: () -> Unit,
    back: () -> Unit,
    loginViewModel: LoginViewModel = viewModel()
) {
    val email: String by loginViewModel.email.observeAsState("")
    val password: String by loginViewModel.password.observeAsState("")
    val loading: Boolean by loginViewModel.loading.observeAsState(initial = false)
```



```

Box(
    contentAlignment = Alignment.Center,
    modifier = Modifier.fillMaxSize()
) {
    if (loading) {
        CircularProgressIndicator()
    }

    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Top
    ) {
        AppBar(
            title = "Login",
            action = back
        )

        TextFormField(
            value = email,
            onValueChange = { loginViewModel.updateEmail(it) },
            label = "Email",
            keyboardType = TextInputType.Email,
            visualTransformation = VisualTransformation.None
        )

        TextFormField(
            value = password,
            onValueChange = { loginViewModel.updatePassword(it) },
            label = "Password",
            keyboardType = TextInputType.Password,
            visualTransformation = PasswordVisualTransformation()
        )
    }
}

```

```

        Spacer(modifier = Modifier.height(20.dp))

        Buttons(

            title = "Login",

            onClick = { loginViewModel.loginUser(home = home) },

            backgroundColor = Color.Magenta

        )

    }

}

```

LoginViewModel:

```

package com.berjin.flashchat.view.login

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.lang.IllegalArgumentException

class LoginViewModel : ViewModel() {

    private val auth: FirebaseAuth = Firebase.auth

    private val _email = MutableLiveData("")
    val email: LiveData<String> = _email

    private val _password = MutableLiveData("")
    val password: LiveData<String> = _password

    private val _loading = MutableLiveData(false)

```

```
val loading: LiveData<Boolean> = _loading

// Update email
fun updateEmail(newEmail: String) {
    _email.value = newEmail
}

// Update password
fun updatePassword(newPassword: String) {
    _password.value = newPassword
}

// Register user
fun loginUser(home: () -> Unit) {
    if (_loading.value == false) {
        val email: String = _email.value ?: throw
IllegalArgumentException("email expected")

        val password: String =
            _password.value ?: throw IllegalArgumentException("password
expected")

        _loading.value = true

        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener {
                if (it.isSuccessful) {
                    home()
                }
                _loading.value = false
            }
    }
}
```

```
}  
}
```

Register.kt:

```
package com.berjin.flashchat.view.register  
  
import androidx.compose.foundation.layout.*  
import androidx.compose.material.CircularProgressIndicator  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.getValue  
import androidx.compose.runtime.livedata.observeAsState  
import androidx.compose.ui.Alignment  
import androidx.compose.ui.Modifier  
import androidx.compose.ui.graphics.Color  
import androidx.compose.ui.text.input.KeyboardType  
import androidx.compose.ui.text.input.PasswordVisualTransformation  
import androidx.compose.ui.text.input.VisualTransformation  
import androidx.compose.ui.unit.dp  
import androidx.lifecycle.viewmodel.compose.viewModel  
import com.berjin.flashchat.view.Appbar  
import com.berjin.flashchat.view.Buttons  
import com.berjin.flashchat.view.TextFormField  
  
@Composable  
fun RegisterView(  
    home: () -> Unit,  
    back: () -> Unit,  
    registerViewModel: RegisterViewModel = viewModel()  
) {  
    val email: String by registerViewModel.email.observeAsState("")
```

```
val password: String by registerViewModel.password.observeAsState("")

val loading: Boolean by
registerViewModel.loading.observeAsState(initial = false)

Box(

    contentAlignment = Alignment.Center,

    modifier = Modifier.fillMaxSize()

) {

    if (loading) {

        CircularProgressIndicator()

    }

    Column(

        modifier = Modifier.fillMaxSize(),

        horizontalAlignment = Alignment.CenterHorizontally,

        verticalArrangement = Arrangement.Top

    ) {

        AppBar(

            title = "Register",

            action = back

        )

        TextFormField(

            value = email,

            onValueChange = { registerViewModel.updateEmail(it) },

            label = "Email",

            keyboardType = TextInputType.Email,

            visualTransformation = VisualTransformation.None

        )

        TextFormField(

            value = password,

            onValueChange = { registerViewModel.updatePassword(it) },

            label = "Password",
```

```

        keyboardType = KeyboardType.Password,

        visualTransformation = PasswordVisualTransformation()

    )

    Spacer(modifier = Modifier.height(20.dp))

    Buttons(

        title = "Register",

        onClick = { registerViewModel.registerUser(home = home) },

        backgroundColor = Color.Blue

    )

}

}

}

```

RegisterViewModel:

```

package com.berjin.flashchat.view.register

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.lang.IllegalArgumentException

class RegisterViewModel : ViewModel() {

    private val auth: FirebaseAuth = Firebase.auth

    private val _email = MutableLiveData("")

    val email: LiveData<String> = _email

    private val _password = MutableLiveData("")

```

```

val password: LiveData<String> = _password

private val _loading = MutableLiveData(false)
val loading: LiveData<Boolean> = _loading

// Update email
fun updateEmail(newEmail: String) {
    _email.value = newEmail
}

// Update password
fun updatePassword(newPassword: String) {
    _password.value = newPassword
}

// Register user
fun registerUser(home: () -> Unit) {
    if (_loading.value == false) {
        val email: String = _email.value ?: throw
IllegalArgumentException("email expected")

        val password: String =
            _password.value ?: throw IllegalArgumentException("password
expected")

        _loading.value = true

        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener {
                if (it.isSuccessful) {
                    home()
                }
            }
    }
}

```

```

        _loading.value = false
    }

}

}

}

```

AuthenticationOption.kt:

```

package com.berjin.flashchat.view

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.material.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import com.berjin.flashchat.ui.theme.FlashChatTheme

@Composable
fun AuthenticationView(register: () -> Unit, login: () -> Unit) {
    FlashChatTheme {
        // A surface container using the 'background' color from the theme
        Surface(color = MaterialTheme.colors.background) {
            Column(
                modifier = Modifier
                    .fillMaxWidth()
                    .fillMaxHeight(),
                horizontalAlignment = Alignment.CenterHorizontally,
                verticalArrangement = Arrangement.Bottom
            )
        }
    }
}

```



```

        ) {

            Title(title = "Air Bird \uD83E\uDD9C")

            Buttons(title = "Register", onClick = register,
backgroundColor = Color.Blue)

            Buttons(title = "Login", onClick = login, backgroundColor =
Color.Magenta)

        }

    }

}

}

```

Widgets.kt:

```

package com.berjin.flashchat.view

import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.ArrowBack
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

```

```

@Composable
fun Title(title: String) {
    Text(
        text = title,
        fontSize = 30.sp,
        fontWeight = FontWeight.Bold,
        modifier = Modifier.fillMaxHeight(0.5f)
    )
}

// Different set of buttons in this page

@Composable
fun Buttons(title: String, onClick: () -> Unit, backgroundColor: Color) {
    Button(
        onClick = onClick,
        colors = ButtonDefaults.buttonColors(
            backgroundColor = backgroundColor,
            contentColor = Color.White
        ),
        modifier = Modifier.fillMaxWidth(),
        shape = RoundedCornerShape(0),
    ) {
        Text(
            text = title
        )
    }
}

@Composable

```

```

fun AppBar(title: String, action: () -> Unit) {

    TopAppBar(

        title = {

            Text(text = title)

        },

        navigationIcon = {

            IconButton(

                onClick = action

            ) {

                Icon(

                    imageVector = Icons.Filled.ArrowBack,

                    contentDescription = "Back button"

                )

            }

        }

    )

}

@Composable

fun TextFormField(value: String, onValueChange: (String) -> Unit, label:
String, keyboardType: KeyboardType, visualTransformation:
VisualTransformation) {

    OutlinedTextField(

        value = value,

        onValueChange = onValueChange,

        label = {

            Text(

                label

            )

        },

        maxLines = 1,

```

```

        modifier = Modifier
            .padding(horizontal = 20.dp, vertical = 5.dp)
            .fillMaxWidth(),
        keyboardOptions = KeyboardOptions(
            keyboardType = keyboardType
        ),
        singleLine = true,
        visualTransformation = visualTransformation
    )
}

@Composable
fun SingleMessage(message: String, isCurrentUser: Boolean) {
    Card(
        shape = RoundedCornerShape(16.dp),
        backgroundColor = if (isCurrentUser) MaterialTheme.colors.primary
    else Color.White
    ) {
        Text(
            text = message,
            textAlign =
                if (isCurrentUser)
                    TextAlign.End
                else
                    TextAlign.Start,
            modifier = Modifier.fillMaxWidth().padding(16.dp),
            color = if (!isCurrentUser) MaterialTheme.colors.primary else
Color.White
        )
    }
}

```

Constants:

```
package com.berjin.flashchat

object Constants {

    const val TAG = "flash-chat"

    const val MESSAGES = "messages"

    const val MESSAGE = "message"

    const val SENT_BY = "sent_by"

    const val SENT_ON = "sent_on"

    const val IS_CURRENT_USER = "is_current_user"

}
```

MainActivity.kt:

```
package com.berjin.flashchat

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import com.google.firebase.FirebaseApp

class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        FirebaseApp.initializeApp(this)

        setContent {

            NavComposeApp()

        }

    }

}
```

NavComposeApp.kt:

```
package com.berjin.flashchat

import androidx.compose.runtime.Composable
import androidx.compose.runtime.remember
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import com.google.firebase.auth.FirebaseAuth
import com.berjin.flashchat.nav.Action
import com.berjin.flashchat.nav.Destination.AuthenticationOption
import com.berjin.flashchat.nav.Destination.Home
import com.berjin.flashchat.nav.Destination.Login
import com.berjin.flashchat.nav.Destination.Register
import com.berjin.flashchat.ui.theme.FlashChatTheme
import com.berjin.flashchat.view.AuthenticationView
import com.berjin.flashchat.view.home.HomeView
import com.berjin.flashchat.view.login.LoginView
import com.berjin.flashchat.view.register.RegisterView

@Composable
fun NavComposeApp() {
    val navController = rememberNavController()
    val actions = remember(navController) { Action(navController) }

    FlashChatTheme {
        NavHost(
            navController = navController,
            startDestination =
                if (FirebaseAuth.getInstance().currentUser != null)

```

```

        Home

    else

        AuthenticationOption

    ) {

        composable(AuthenticationOption) {

            AuthenticationView(

                register = actions.register,

                login = actions.login

            )

        }

        composable(Register) {

            RegisterView(

                home = actions.home,

                back = actions.navigateBack

            )

        }

        composable(Login) {

            LoginView(

                home = actions.home,

                back = actions.navigateBack

            )

        }

        composable(Home) {

            HomeView()

        }

    }

}

```

AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.berjin.flashchat">

    <uses-permission android:name="android.permission.INTERNET"/>

    <application

        android:allowBackup="true"

        android:icon="@mipmap/ic_launcher"

        android:label="@string/app_name"

        android:roundIcon="@mipmap/ic_launcher_round"

        android:supportsRtl="true"

        android:theme="@style/Theme.FlashChat">

        <activity

            android:name=".MainActivity"

            android:exported="true"

            android:label="@string/app_name"

            android:theme="@style/Theme.FlashChat.NoActionBar">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>

            </intent-filter>

        </activity>

    </application>

</manifest>
```