Chat Connect-A Real-Time Chat and Communication App

1.Introduction:

1.1.Overview:

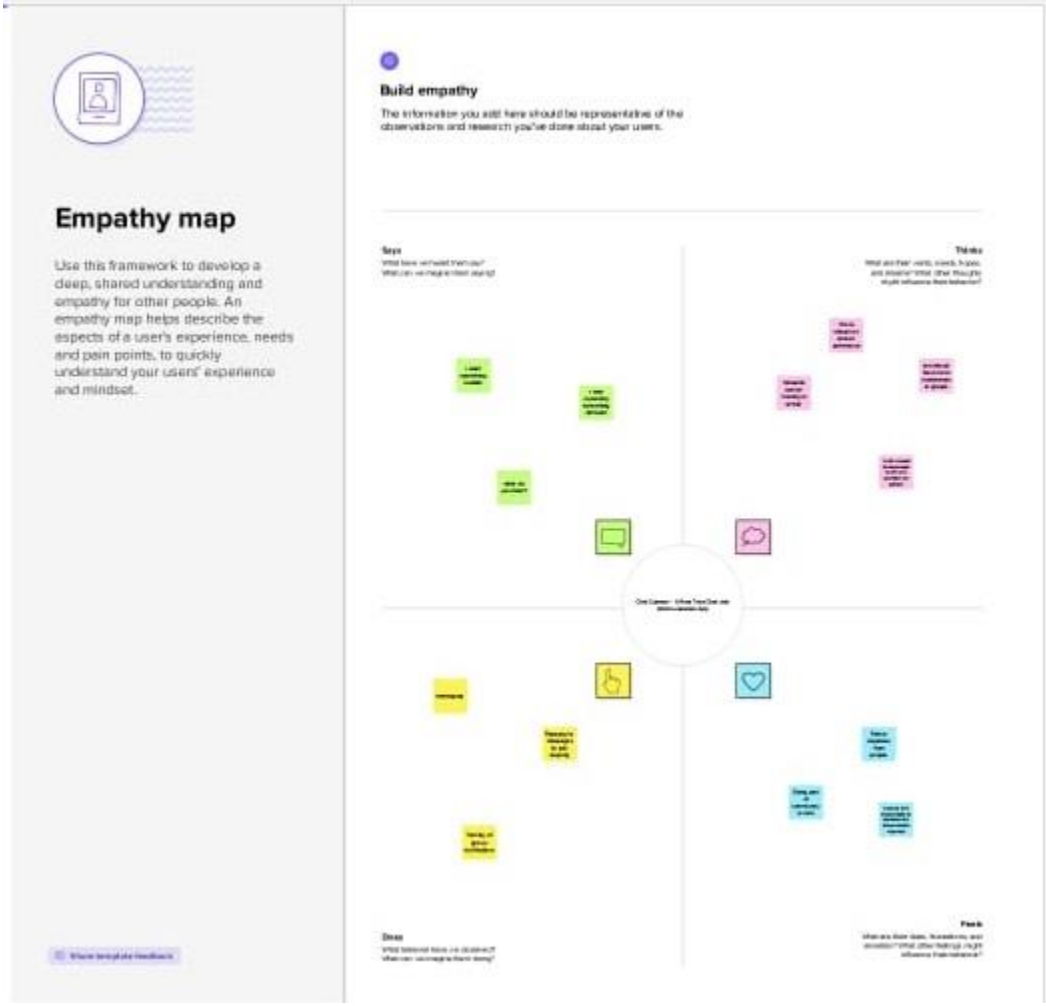
- Chatting app allows you to communicate with your customers in web chat rooms.
- It enables you to send and receive messages. Chatting apps make it easier, simpler, and faster to connect with everyone and it is also easy to use.
- There are many types of chatting apps and every one has its own format, design, and functions. A chat room is an online platform that enables users to communicate with each other in real time.
- Chat rooms are typically hosted on a server with an internet connection, enabling members from around the world to hold conversations about various topics.

1.2.Purpose:

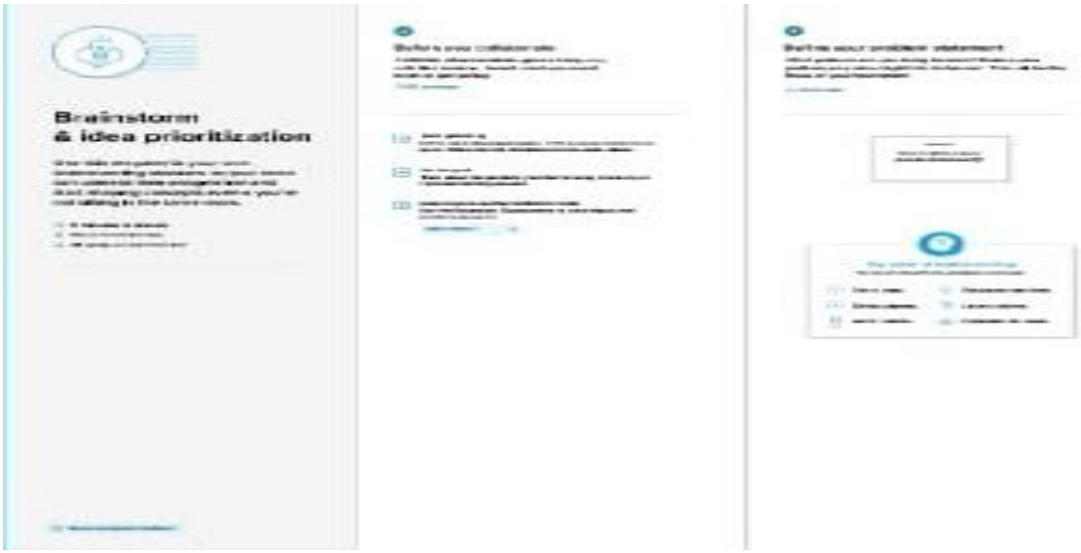
- A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time.
- With a web or mobile chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.
- It enables you to send and receive messages.
- Chatting apps make it easier, simpler, and faster to connect with everyone and it is also easy to use.

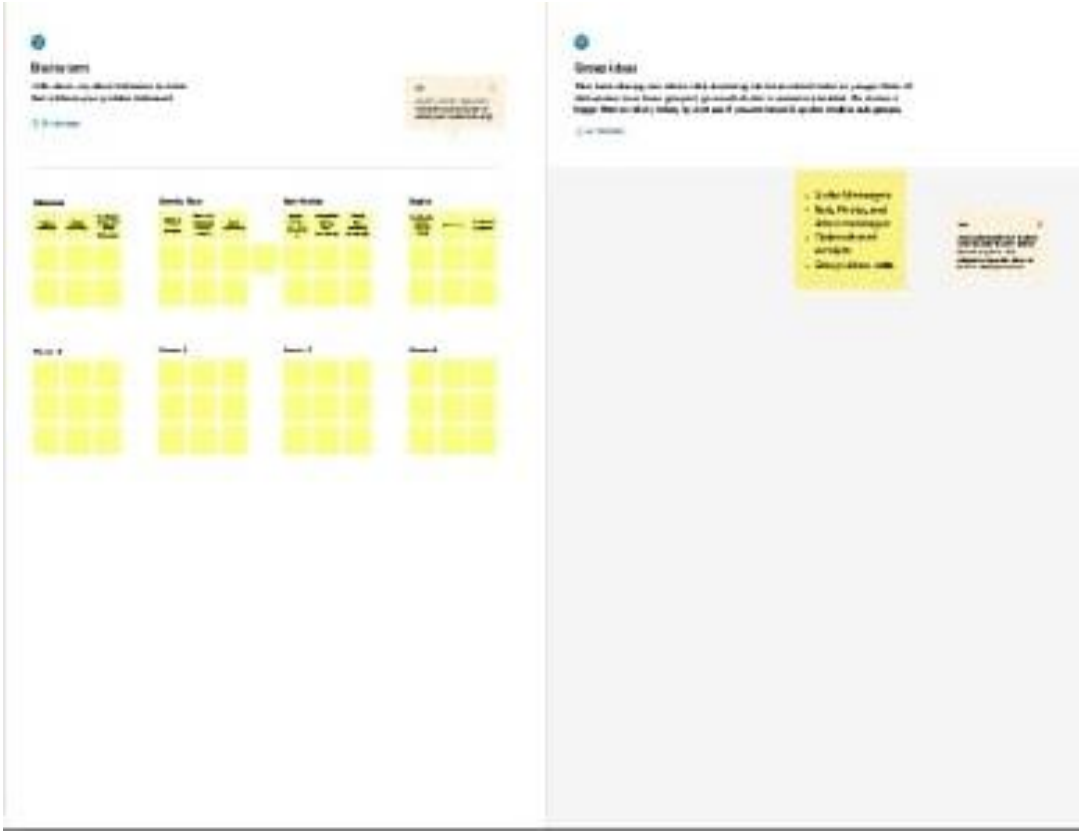
2.Problem definition & design thinking

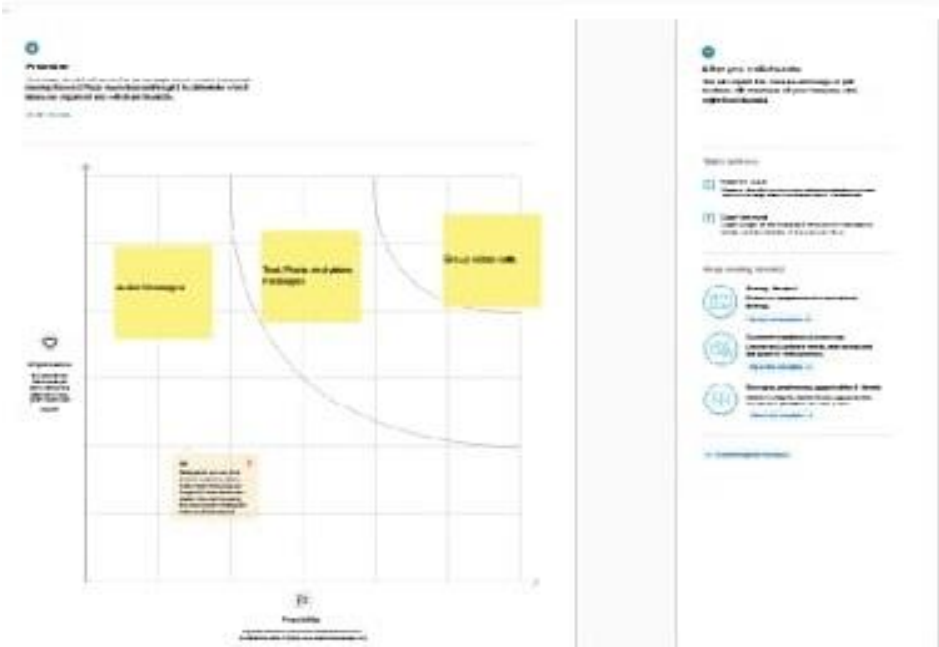
2.1.Empathy map:



2.2.Ideation & Brainstroming map:


Brainstorm
& idea prioritization





3.Result:

Chat Connect

Register

Login


Register
Email
Password
Register


Register
Email
nive@gmail.com
Password
Register

Type Your Message

← Login

Email

Password

Login

Type Your Message

Hi

4. Advantage & Disadvantage

Advantages:

- Voice chat provides your team meaningful, immediate, and efficient communication and collaboration.
- As more and more teams are distributed, more meaningful communication means more successful collaboration. And the key personal touch provided by voice chat allows you to provoke a response.
- A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time.
- With a web or mobile chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.

Disadvantages:

- You can't be sure other people are being honest or that they are who they say they are.
- If you are feeling vulnerable, people online might try to take advantage of you.
- Building relationships online can result in your spending less time with friends and family.
- The possible disadvantages include the “lack of security, Internet addiction, information overload, and loss of social contacts” (Drahošová and Balco 1009). Even though these phenomena are negative, it is possible to state that their effects can be mitigated.
- There is high power consumption in digital communication.
- There is a requirement for synchronization in the case of synchronous modulation.
- There is a sampling error.
- The most common limitation of digital communication is that it requires more transmission bandwidth.

5.Applications

- Telegram.
- Viber.
- Signal.
- WhatsApp.
- Facebook Messenger.
- Line.
- WeChat.
- Skype.tions:
- WhatsApp was the first application for mobile chat. Afterwards, many other applications came but still.

6.Conclusion

- The main objective of the project is to develop a Secure Chat Application.
- I had taken a wide range of literature review in order to achieve all the tasks, where I came to know about some of the products that are existing in the market.
- I made a detailed research in that path to cover the loop holes that existing systems are facing and to eradicate them in our application.
- In the process of research I came to know about the latest technologies and different algorithms.

- The chat app provides a better and more flexible chat system. Developed with the latest technology in the way of providing a reliable system.
- The main advantage of the system is instant messaging, real-world communication, added security, group chat, etc.

7.Future scope:

- With the knowledge I have gained by developing this application, I am confident that in the future I can make the application more effectively by adding this services.
- Extending this application by providing Authorisation service.
- Creating Database and maintaining users.
- Increasing the effectiveness of the application by providing Voice Chat.
- Extending it to Web Support.

8. Appendix

A.Source code:

Navigation.kt

```
package com.project.pradyotprakash.flashchat.nav

import androidx.navigation.NavHostController

import com.project.pradyotprakash.flashchat.nav.Destination.Home

import com.project.pradyotprakash.flashchat.nav.Destination.Login

import com.project.pradyotprakash.flashchat.nav.Destination.Register


/**

 * A set of destination used in the whole application

 */

object Destination {

   const val AuthenticationOption = "authenticationOption"
```

```
    const val Register = "register"
    const val Login = "login"
    const val Home = "home"
}

/**
 * Set of routes which will be passed to different composable so that
 * the routes which are required can be taken.
 */
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() }
}
```

Color.kt:

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)

val Purple500 = Color(0xFF6200EE)

val Purple700 = Color(0xFF3700B3)

val Teal200 = Color(0xFF03DAC5)
```

shape.kt:

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.Shapes

import androidx.compose.ui.unit.dp


val Shapes = Shapes(

    small = RoundedCornerShape(4.dp),

    medium = RoundedCornerShape(4.dp),

    large = RoundedCornerShape(0.dp)

)
```

Theme.kt:

```
package com.project.pradyotprakash.flashchat.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme

import androidx.compose.material.MaterialTheme

import androidx.compose.material.darkColors

import androidx.compose.material.lightColors

import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(

 primary = Purple200,

 primaryVariant = Purple700,

  secondary = Teal200

)


private val LightColorPalette = lightColors(

    primary = Purple500,

    primaryVariant = Purple700,

    secondary = Teal200

)


@Composable
```

```
fun FlashChatTheme(darkTheme: Boolean = isSystemInDarkTheme(), content:
@Composable() () -> Unit) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

Type.kt:

```
package com.project.pradyotprakash.flashchat.ui.theme
import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
```

```
import androidx.compose.ui.unit.sp


/**
 * Set of Material typography styles to start with
 */
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
)
```

Home.kt:

```
package com.project.pradyotprakash.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.project.pradyotprakash.flashchat.Constants
import com.project.pradyotprakash.flashchat.view.SingleMessage

/**
 * The home view which will contain all the code related to the view for HOME.
 *
 * Here we will show the list of chat messages sent by user.
 * And also give an option to send a message and logout.
 */

@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 ->
        val isCurrentUser = message[Constants.IS_CURRENT_USER] as Boolean


        SingleMessage(
            message = message[Constants.MESSAGE].toString(),
            isCurrentUser = isCurrentUser
        )
    }
}
OutlinedTextField(
    value = message,
    onValueChange = {
        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.addMessage()
                }
            ) {
                Icon(
                    imageVector = Icons.Default.Send,
                    contentDescription = "Send Button"
                )
            }
        }
    )
  }
}
```

HomeViewModel.kt:

```
package com.project.pradyotprakash.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.project.pradyotprakash.flashchat.Constants

import java.lang.IllegalArgumentException


/**

 * Home view model which will handle all the logic related to HomeView

 */

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


    /**

     * Update the message value as user types

     */

    fun updateMessage(message: String) {

        _message.value = message

    }


    /**

     * Send 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 = ""
    }
  }
}

/**
 * Get the messages
 */
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)
        }
    }

    /**
     * Update the list after getting the details from firestore
     */
    private fun updateMessages(list: MutableList<Map<String, Any>>) {
        _messages.value = list.asReversed()
    }
```

```
}
```

Login.kt:

```
package com.project.pradyotprakash.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.project.pradyotprakash.flashchat.view.Appbar

import com.project.pradyotprakash.flashchat.view.Buttons

import com.project.pradyotprakash.flashchat.view.TextFormField


/**
```

```
 * The login view which will help the user to authenticate themselves and go to the
 * home screen to show and send messages to others.
 */

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 = KeyboardType.Email,
        visualTransformation = VisualTransformation.None
    )
    TextFormField(
        value = password,
        onValueChange = { loginViewModel.updatePassword(it) },
        label = "Password",
        keyboardType = KeyboardType.Password,
        visualTransformation = PasswordVisualTransformation()
    )
    Spacer(modifier = Modifier.height(20.dp))
```

```
            Buttons(
                title = "Login",
                onClick = { loginViewModel.loginUser(home = home) },
                backgroundColor = Color.Magenta
            )
        }
    }
}
```

LoginViewModel.kt:

```
package com.project.pradyotprakash.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
/**
 * View model for the login view.
 */
```

```
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.project.pradyotprakash.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.project.pradyotprakash.flashchat.view.Appbar

import com.project.pradyotprakash.flashchat.view.Buttons

import com.project.pradyotprakash.flashchat.view.TextFormField


/**

 * The Register view which will be helpful for the user to register themselves into

 * our database and go to the home screen to see and send messages.
```

```
 */

@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 = KeyboardType.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.kt:

```
package com.project.pradyotprakash.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

/**
 * View model for the login view.
 */
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.project.pradyotprakash.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.foundation.shape.RoundedCornerShape
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.project.pradyotprakash.flashchat.ui.theme.FlashChatTheme

/**
 * The authentication view which will give the user an option to choose between
 * login and register.
 */

@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 = "⚡ Chat Connect")
            Buttons(title = "Register", onClick = register, backgroundColor =
Color.Blue)
            Buttons(title = "Login", onClick = login, backgroundColor =
Color.Magenta)
        }
    }
  }
}
```

Constants.kt:

```
package com.project.pradyotprakash.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.project.pradyotprakash.flashchat
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import com.google.firebase.FirebaseApp
/**
 * The initial point of the application from where it gets started.
 *
 * Here we do all the initialization and other things which will be required
 * thought out the application.
 */
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
        FirebaseApp.initializeApp(this)
        setContent {
            NavComposeApp()
        }
    }
}
```

NavComposeApp.kt:

```
package com.project.pradyotprakash.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.project.pradyotprakash.flashchat.nav.Action
import
com.project.pradyotprakash.flashchat.nav.Destination.AuthenticationOption
import com.project.pradyotprakash.flashchat.nav.Destination.Home
import com.project.pradyotprakash.flashchat.nav.Destination.Login
import com.project.pradyotprakash.flashchat.nav.Destination.Register
import com.project.pradyotprakash.flashchat.ui.theme.FlashChatTheme
```

```
import com.project.pradyotprakash.flashchat.view.AuthenticationView
import com.project.pradyotprakash.flashchat.view.home.HomeView
import com.project.pradyotprakash.flashchat.view.login.LoginView
import com.project.pradyotprakash.flashchat.view.register.RegisterView
/**
 * The main Navigation composable which will handle all the navigation stack.
 */
@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()
    }
  }
}
}
```