CHAT CONNECT – A real-time chat and communication app

By using android development application

Under the guidance of

Mrs.C.Kondalraj,M.C.A., B.Ed., M.Phil., SET.,

S. Ramachandran

M. Karan kumar

M. Kathiravan

P. Kaviyarasan

CONTENTS

8 APPENDIX

INTRODUCTION:

Overview:

- Connect Chat is a real-time messaging tool that enables users to chat with individuals and groups, quickly share files, and collaborate on any record by connecting with the right people instantly.
- Connect Chat animates communication around records, Visual Task Boards, topics of interest, or groups of people.

Preview:

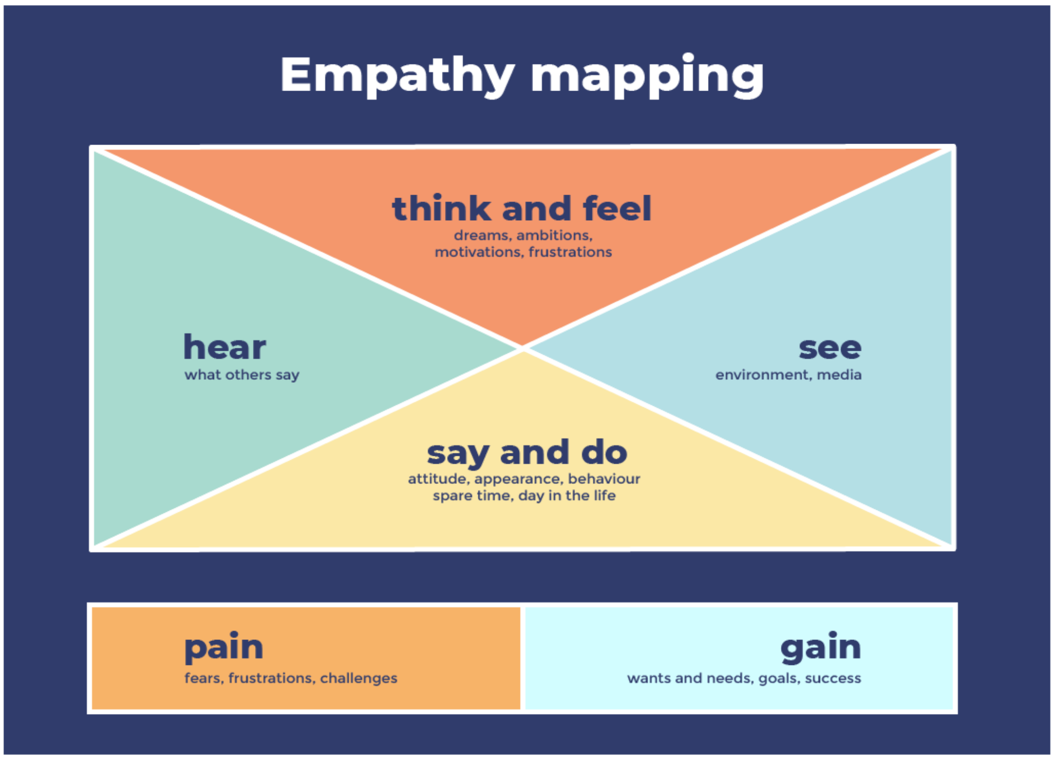
- This guide shows you how to extend an app that displays messages to the user and receives the user's replies, such as a chat app, to hand message display and reply receipt off to an Auto device
- Staying connected through messages is important to many drivers.
- Chat apps can let users know if a child needs to be picked up or if a dinner location has been changed.
- The Android framework lets messaging app,s extend their services into the driving experience using a standard user interface that lets.

Problem Definition & Design Thinking:

Empathy Map:

- Let’s start with the basics. An *empathy map* is a design tool that is particularly useful in UX design. This method helps you step into the shoes of your target users and understand how they think, feel, and act.

- Using an empathy map, you can paint a picture of your users—and how they engage with products, services, and the world around them. You can also map out the perceptions, motivations, and behaviors that drive their decision.


Empathy mapping
think and feel
dreams, ambitions, motivations, frustrations
hear
what others say
see
environment, media
say and do
attitude, appearance, behaviour spare time, day in the life
pain
fears, frustrations, challenges
gain
wants and needs, goals, success

Ideation & Brainstorming Map:

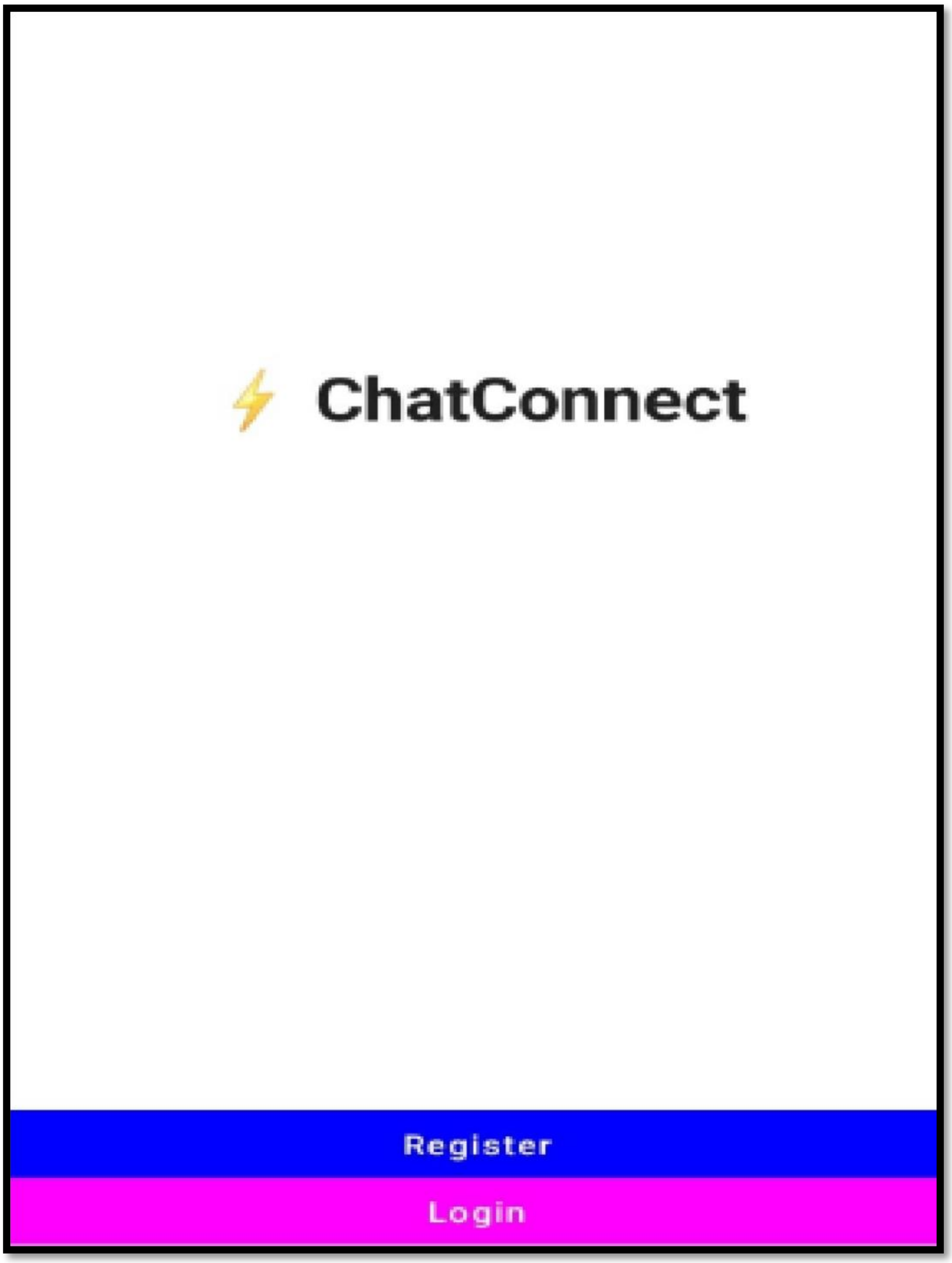
- down into smaller and smaller concepts. The big idea is like the trunk of a tree, each idea a branch, each sub-idea a smaller branch, and so on down until you get to the twigs and leaves. You can always follow any branch back to get to the main idea.

- We've got a full tutorial on mind mapping, Mind Mapping 101, so if you're not familiar with the concept, you should check it out before continuing. In today's article we're going to look at some of the best mind mapping apps available for both individual users and small teams.

RESULT

First Screen:

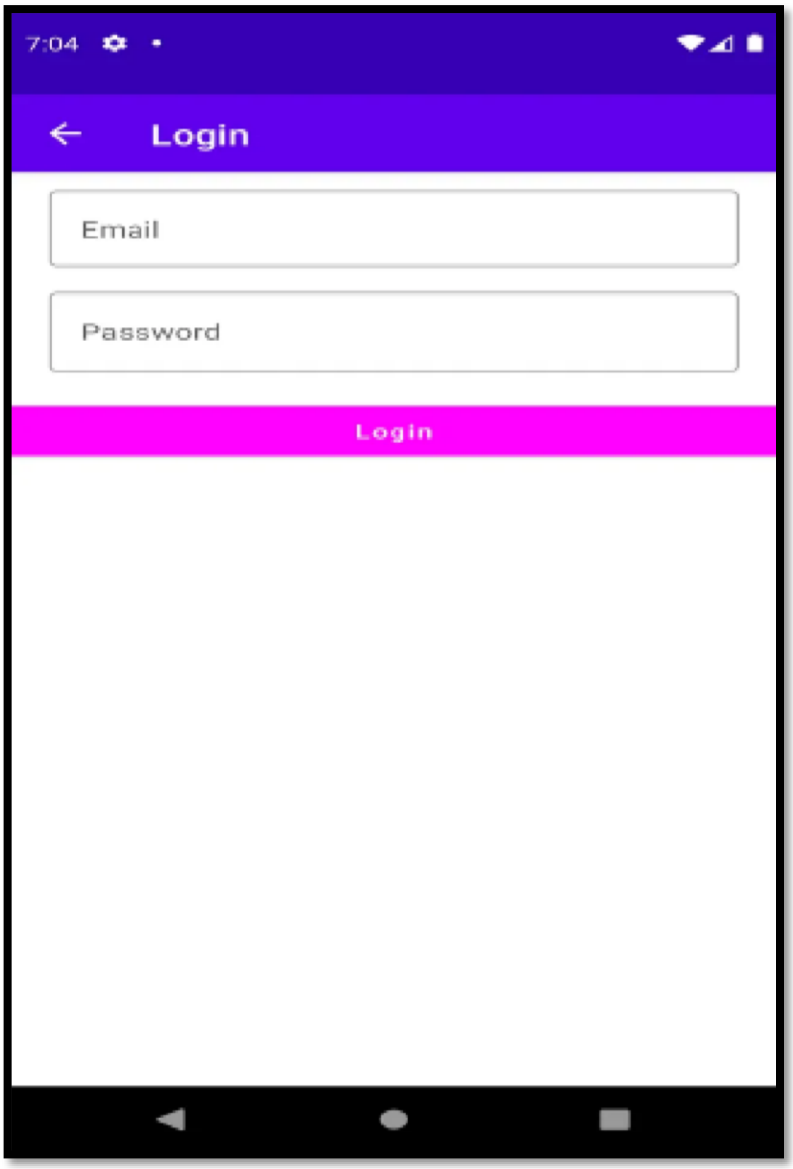
Login Page :


ChatConnect

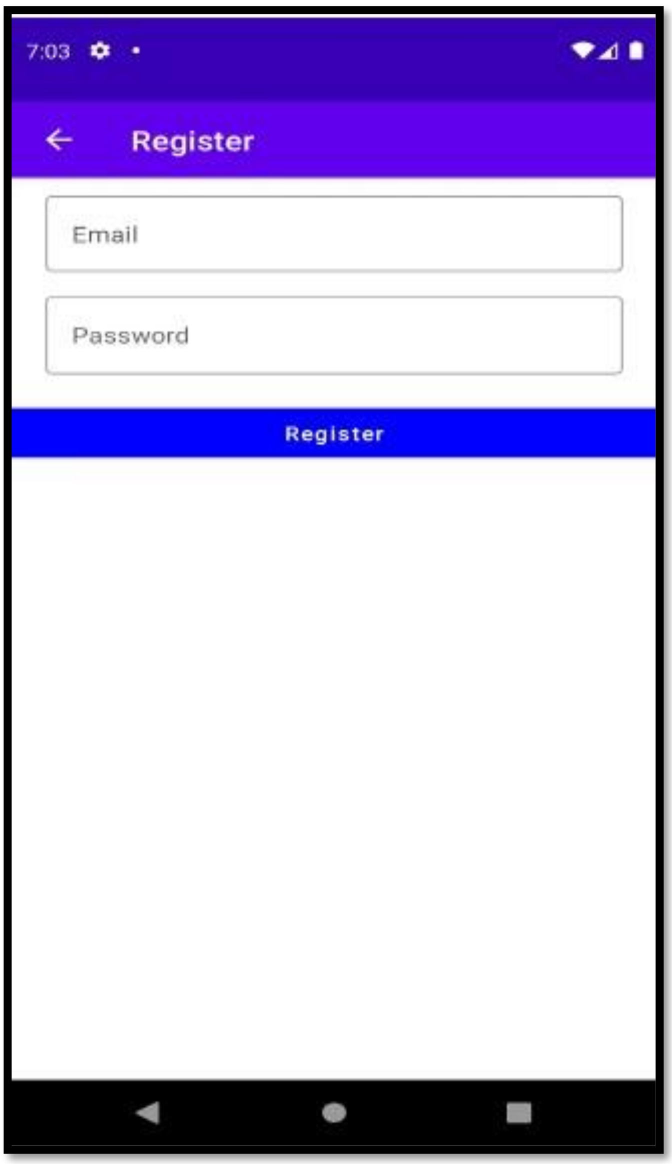
Register

Login

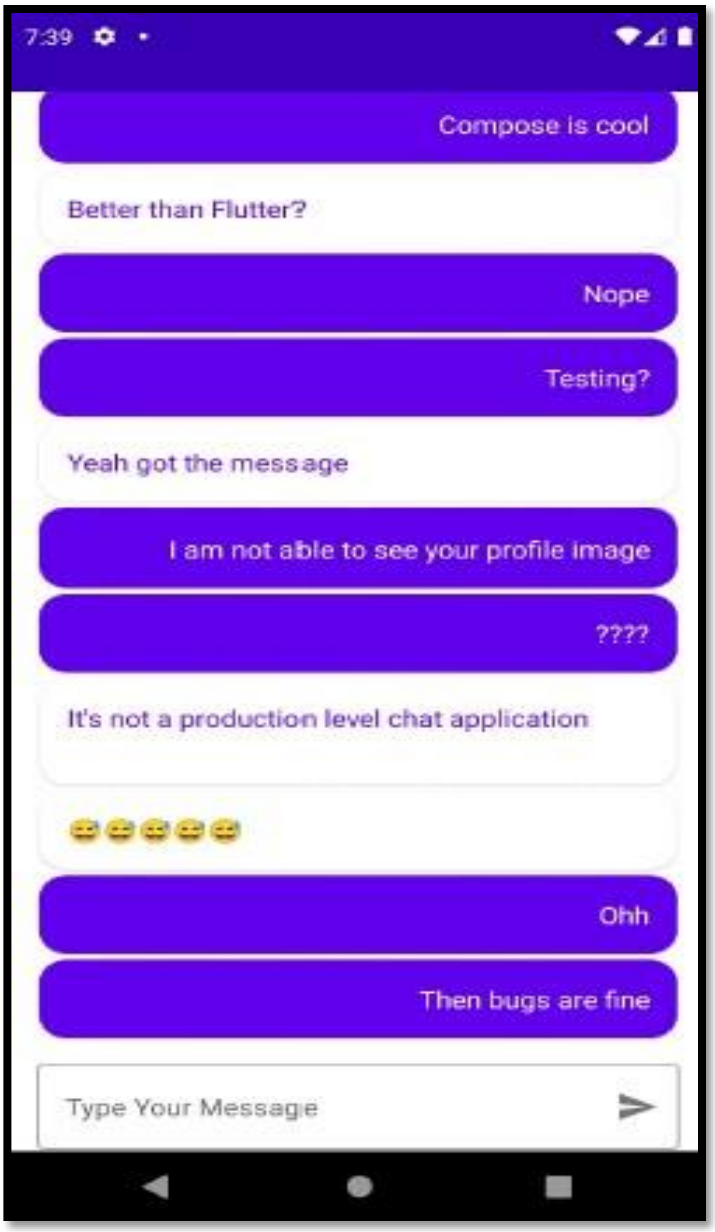
Login Page :



Register Page :



Home Screen:



7:39

Compose is cool

Better than Flutter?

Nope

Testing?

Yeah got the message

I am not able to see your profile image

????

It's not a production level chat application

😅😅😅😅😅

Ohh

Then bugs are fine

Type Your Message

ADVANTAGES &

DISADVANTAGES

ADVANTAGES:

1. Real-time text preview
2. Clear context
3. File transfer

4. Multilingual support
5. Mobile Messaging Integrations
6. 24/7 support
7. Personal
8. Data security

<u>DISADVANTAGES:</u>

1. The need to be online to offer support
2. Online Trolls
3. Doesn’t work well for older demographics

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.

- As a result, the product has been successfully developed in terms of extendability, portability, and maintainability and tested in order to meet all requirements that are Authentication , Integrity and Confidentiality

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 Authorization service.
    - Creating Database and maintaining users.
    - Increasing the effectiveness of the application by providing Voice Chat.
    - Extending it to Web Support.

APPENDIX

Source code:

<u>MainActivity.kt file:</u>

```
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()

        }
```

```
    }
  }
```

Navigation.kt file:

```
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

importcom.project.pradyotprakash.flashchat.nav.Destination.Regist
er


   /**
   * 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()
}
}
```

AuthenticationOption.kt file 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)

            }

        }

    }

}
```

Widgets.kt:

```
package com.project.pradyotprakash.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

import com.project.pradyotprakash.flashchat.Constants


/**
```

* Set of widgets/views which will be used throughout the application.

* This is used to increase the code usability.

```
 */

@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
```

```
        )
    }
}
```

Home.kt file:

```
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.addMessag
e()

                    }
                ) {
                    Icon(
                        imageVector = Icons.Default.Send,
contentDescription = "Send Button"

                    )
                }
            }
        )
    }
}
```

HomeViewModel class:

```
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().s
e t(
                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 file:

```
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.

 */
```

```
@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 = 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 class:

```
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
                }
        }
    }
}
```

<u>Register.kt file</u> 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 class:

```
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

                }

        }

    }

}
```