Project Report

Introduction :

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.

Teleconferencing or chatting is a method of using technology to bring people and ideas “together” despite of the geographical barrier. The technology has been available for year by acceptance it was quit recent. Our project is the example of chat connect server . It is made up of 2 applicate the client applications ,which runs through a network. To start chatting client should get connected to server where they can practice two kind of chatting public one and private one(between any 2 user only) and drawing the last one security measure user take.

Purpose:

With multi-tasking mechanism playing the major focus, today’s chat apps are explored globally by billions of users for both personal as well as commercial fulfillment .

At the heart of these chat app innovation lies fascination for mobile technology that was earlier seen as a fleeting fad.

As we progress in this blog, we are going to glance at how chat apps are playing several roles, including enhancing engagements, monetization and user retention and more.

Unique Apps For Prolific Engagement:

Providing users with rich features, these apps deliver incredible live chat experience, allowing users to share text messages, images, emoticons, stickers, audio and video clips and other media content.

Multipurpose Use of Mobile Chat Apps:

WeChat is more than just mere chat and news reading; users can pay bills, make purchases, book transportations, doctor's appointments and reservations, watch traffic updates and report suspicious incidents and also earn loyalty rewards

Problem Definition and Design thinking:

Empathy map:

Template

Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share template feedback

Build empathy

The information you add here should be representative of the observations and research you've done about your users.

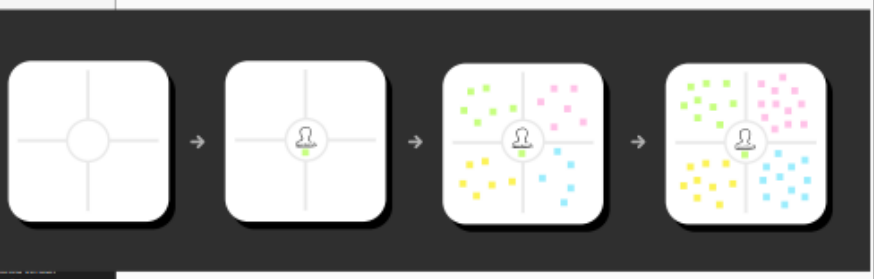
Says
What have we heard them say?
What can we imagine them saying?

Thinks
What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

banker

Does
What behavior have we observed?
What can we imagine them doing?

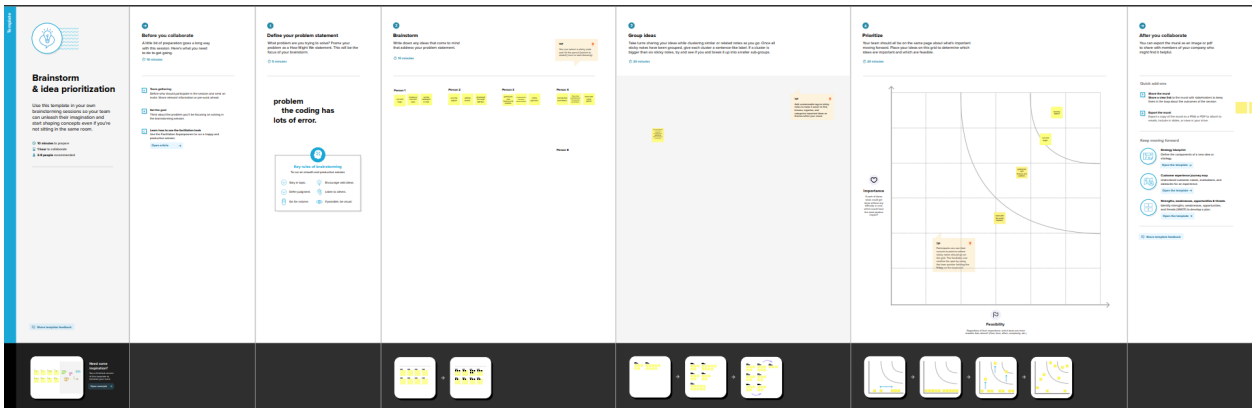
Feels
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?



of this template to kickstart your work.

Open example

Ideation & Brainstorming map:

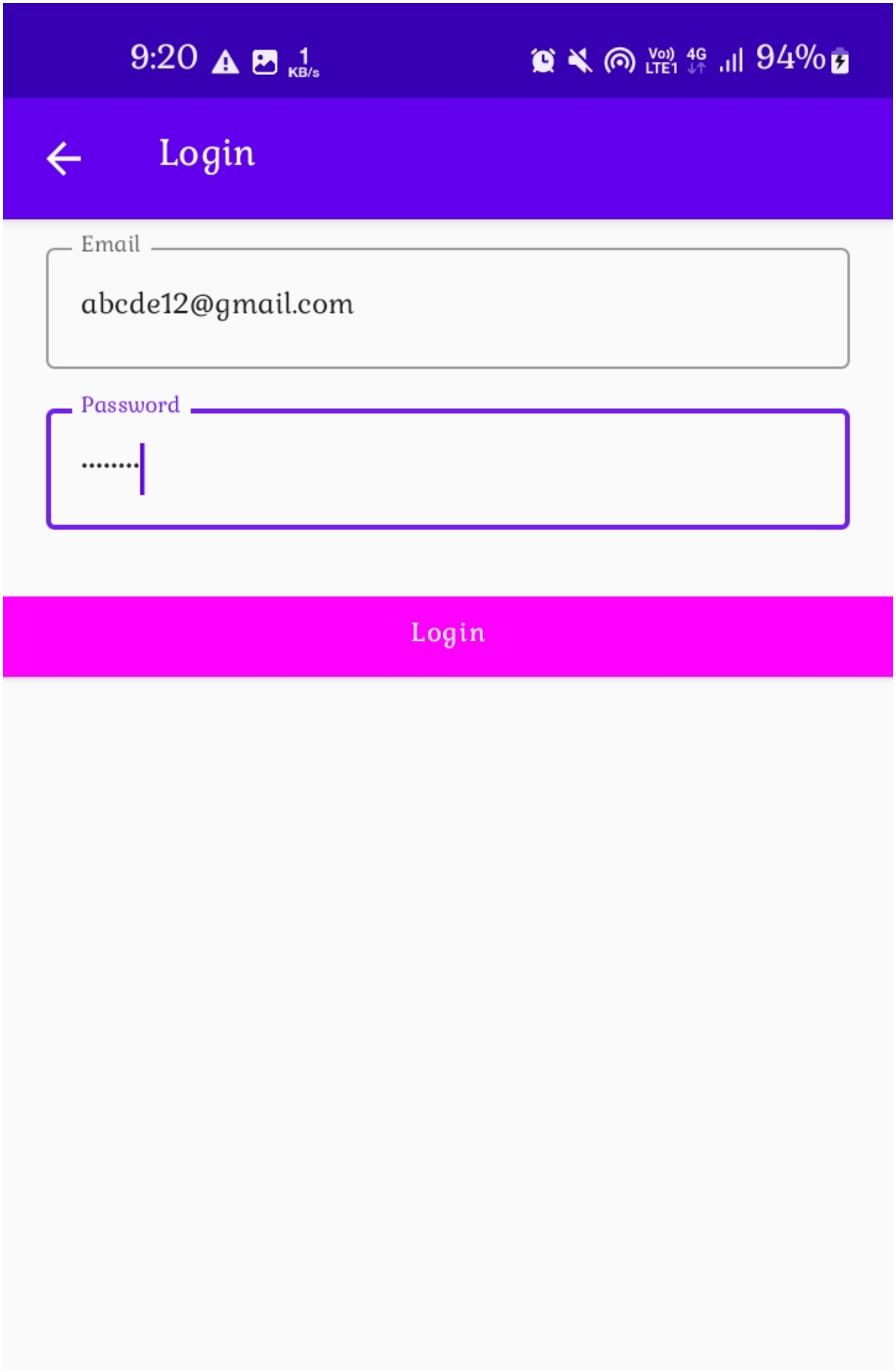


Result:

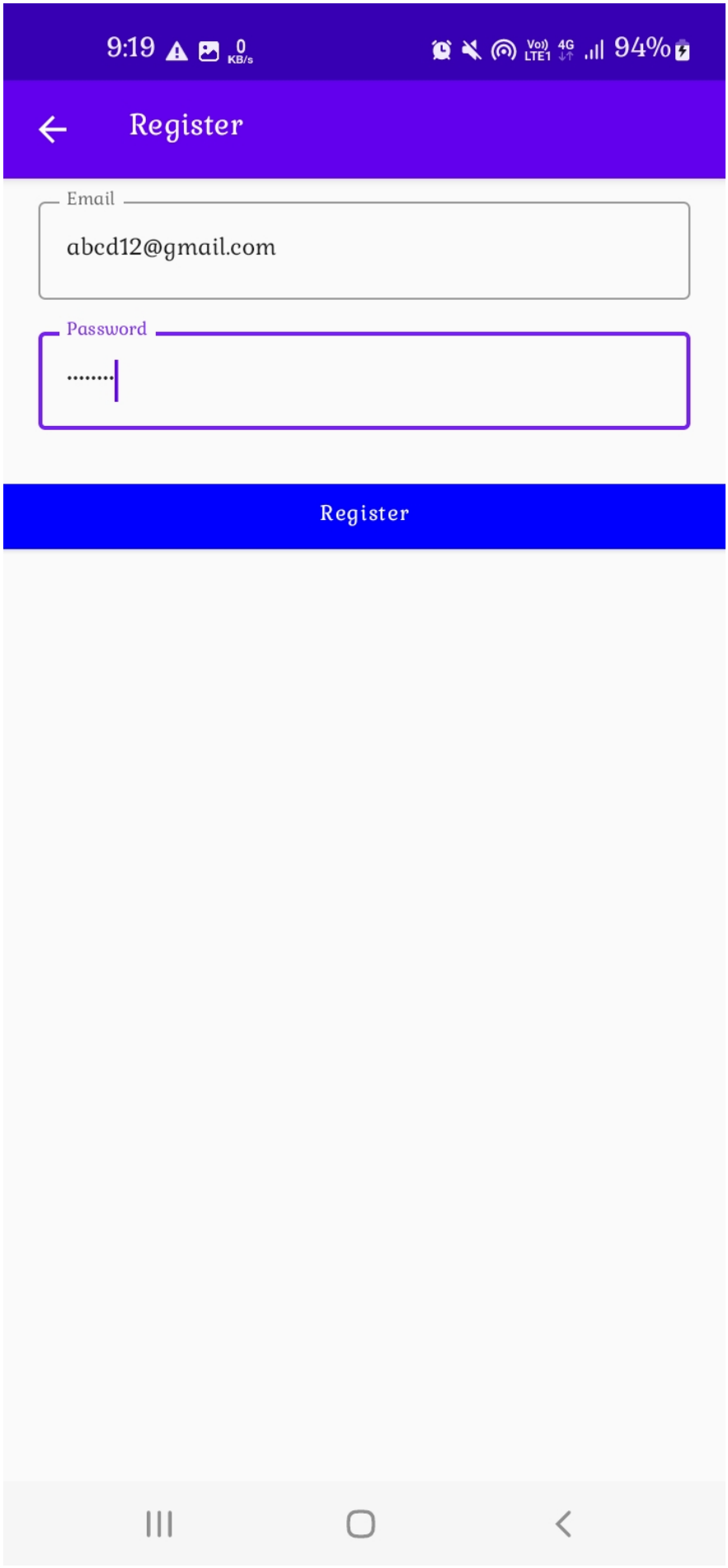
First page:

⚡ Chat Connect

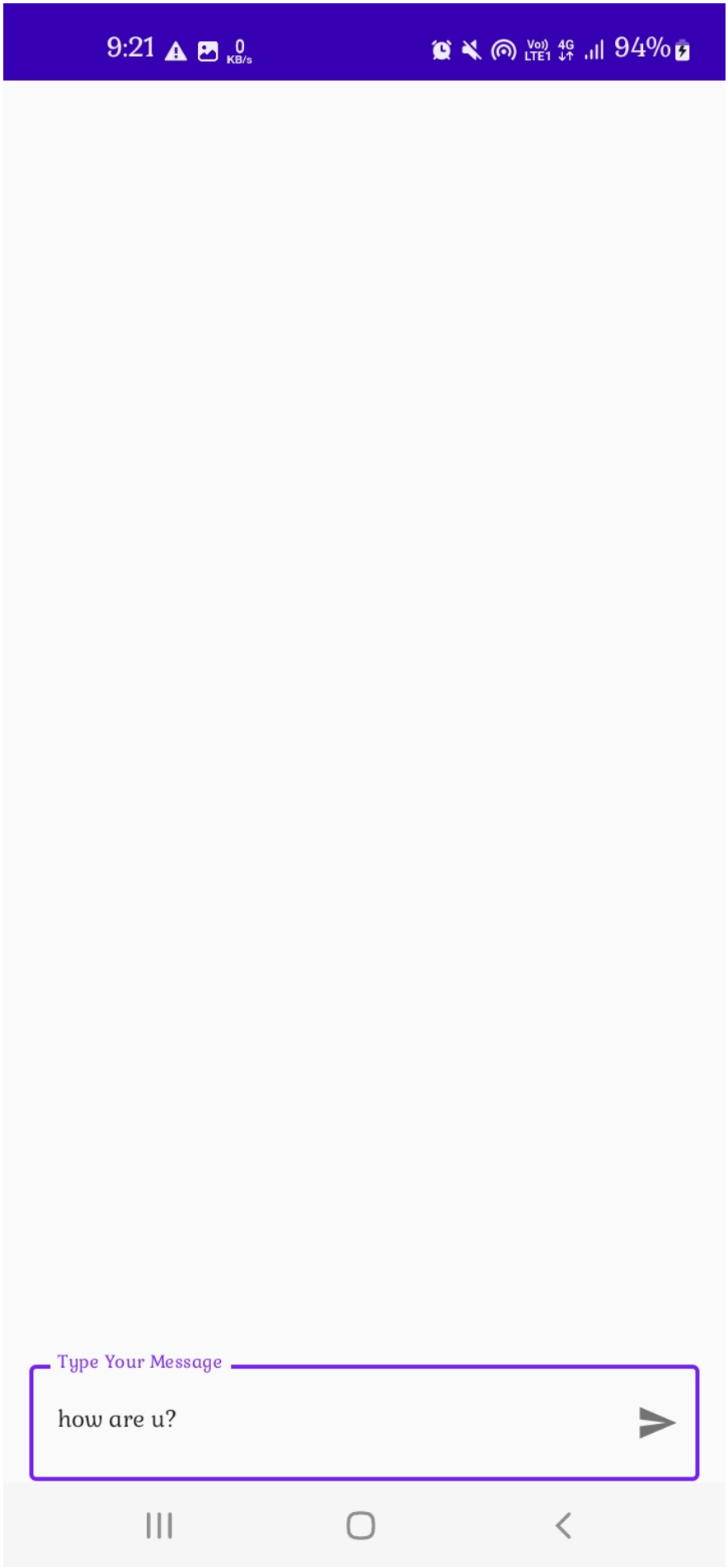
Login page:


9:20
1 KB/s
Vo)) LTE1
4G
94%
Login
Email
abcde12@gmail.com
Password
Login

Register Page:


9:19
94%
Register
Email
abcd12@gmail.com
Password
Register

Home page:


9:21
94%
Type Your Message
how are u?

Advantage:

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. This also keeps users conversing on your platform instead of looking elsewhere for a messaging solution. Whether it's private chat, group chat, or large-scale chat, adding personalized chat features to your app can help ensure that your users have a memorable experience.

Messaging has become a part of our everyday lives in part due to its convenience for real-time chat communication and simple-to-use functionality. For instance, an iOS or text message on an iPhone or Android device from a friend, an email from a co-worker on Microsoft or Gmail, a team chat in a Slack or Microsoft Teams workspace, or even instant messaging through social media. These messaging and real-time chat applications play an important role in how the world interacts today, due to their immediacy and vast capabilities.

Disadvantage:

When you chat with someone in chat rooms, you may have things in common, but you don't know the person intimately like you do your friends and family. While this might sound like a drawback, it can be a benefit as well, as you can sometimes get a better perspective from a stranger than you can from someone who knows you well . Those who love you may be

biased in your favor, and they might not be able to see your situation objectively.

In addition, if you are someone who struggles with sharing your emotions, you might feel more comfortable with written communication. There is some safety and comfort to not having to share deep feelings face to face. This can be especially true if you struggle with anxiety and low self-esteem. And anonymity is another perk of online chat.

Finally, if you are feeling isolated, it's better to have someone to chat with online than no one at all - that is, unless you chat with someone who is toxic. Keeping our emotions inside can be damaging, so even if you do not have a one-on-one chat with someone, sharing on a message board and receiving feedback from others can make all the difference.

Application:

- **Direct conversations between two users.**
- **Group conversations between three or more users.**
- **Conversations linked to records. Comments and work notes appear in conversations in real time and users can update the record directly from the conversation.**
- **Drag-and-drop sharing of links, files, and records.**

Conclusion:

From above discussion, it is evident how chat applications are purposed to play various roles in addition to acting as a chat communication medium.

As days pass by, chat messengers are going to be even more advanced in its capability. While app publishers exhibit immense joy, optimism and excitement for their success so far, they all almost agree that chat app industry is still its experimental and exploratory phase

After spending loads of time perfecting user experience, publishers now gravitate to media-powered partnerships. Hope you have learned enough from chat apps success stories to initiate your own experiment.

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 each web support.

Appendix:

Adding MainActivity.kt file:

```
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()
```

```
 }


 }

}
```

Adding NavComposeApp.kt file:

```
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()
        }
        }
        }
        }
```

Adding Constants object:

```
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"
```

Adding Navigation.kt in nav package:

```
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() }
}
```

Adding Home View:

```
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()
        }
        }
```

Adding login package in view package:

```
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
        )
        }
        }
        }
```

Adding LoginViewModel.kt file :

```
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
        }
        }
        }
        }
```

Adding register package in view package:

```
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
        )
        }
```

```
        }
        }
```

Adding RegisterViewModel.kt file :

```
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
        }
        }
        }
        }
```

Adding Home:

```
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"
)
}
}
)
}
}
```

Adding 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
        )
        }
        }
```

Adding 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)

        }

        }

        }

        }
```