**ChatConnect – A Real-Time Chat and Communication App**

**1. INTRODUCTION**

* 1. **Overview :-**

Messaging has become a part of our everyday lives in part due to its convenience for[real-time chat communication](https://www.pubnub.com/learn/glossary/what-is-a-chat-api/) 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. 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](https://www.pubnub.com/use-case/in-app-chat/), users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.

* 1. **Purpose :-**

Engage more users with in-app chat

Chat APIs for experiences that keep users coming back.

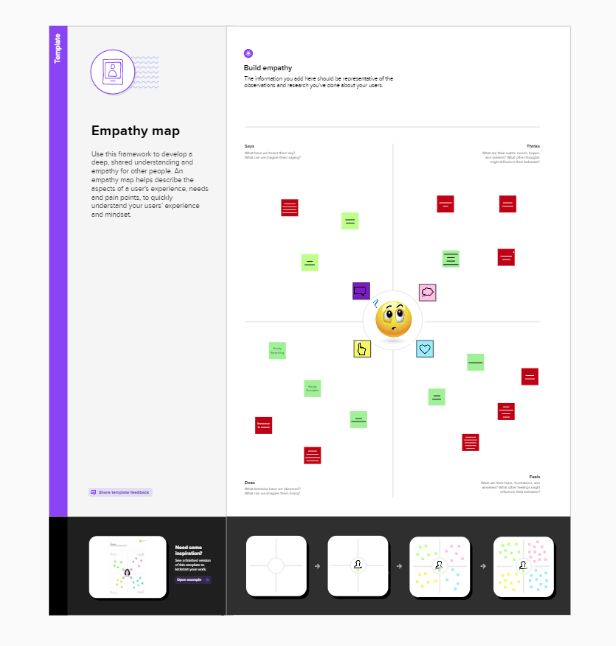
In this chat application guide, we’ll explore:

* How custom in-app chat features enable user engagement
* The long-term factors to consider when building a real-time chat application
* How engaging messaging apps lead to an improved user experience

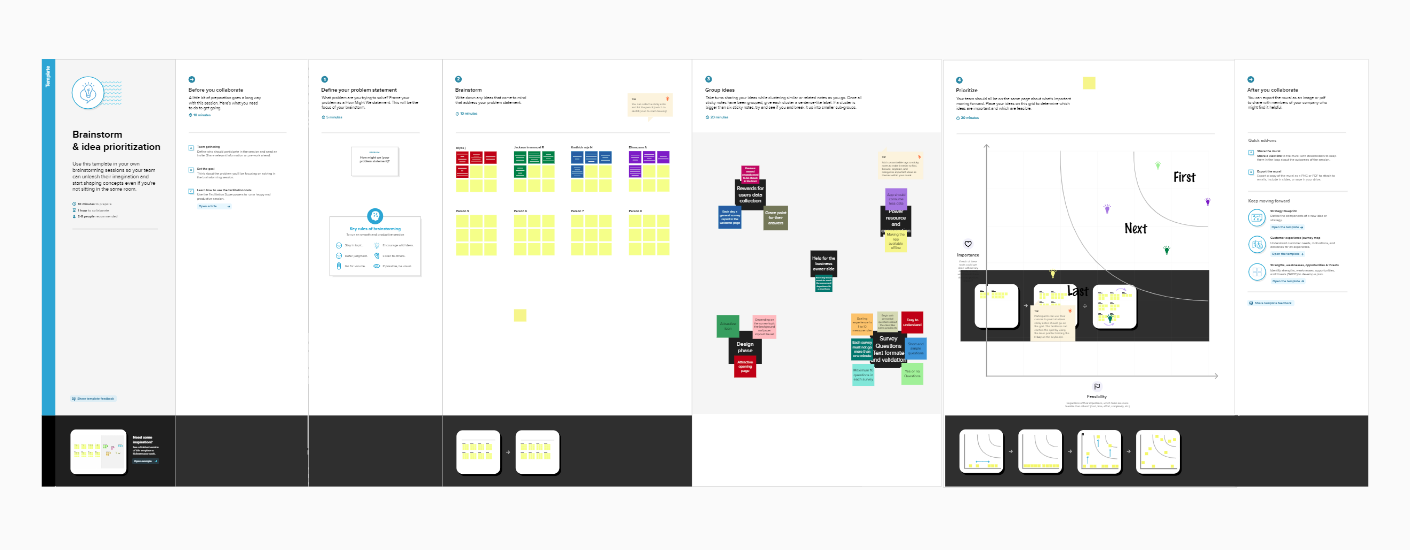
By the end of this guide, you’ll understand the benefits of implementing personalized, real-time chat app features to increase user engagement and retention on your platform.

1. **Problem Definition & Design Thinking:-**

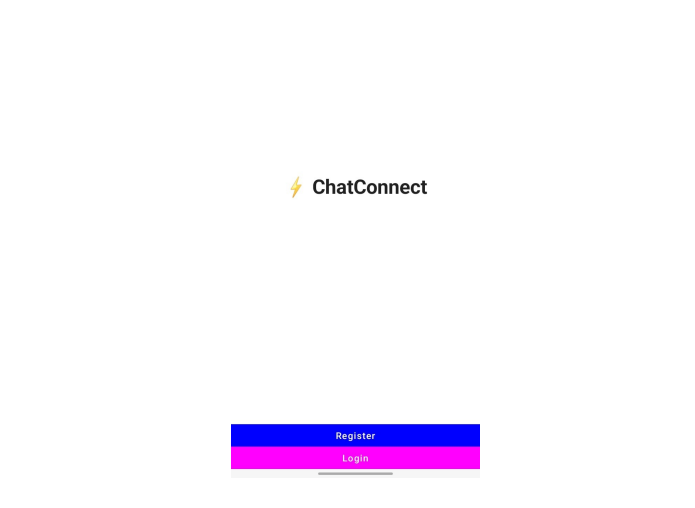
**2.1 Empathy Map:-**

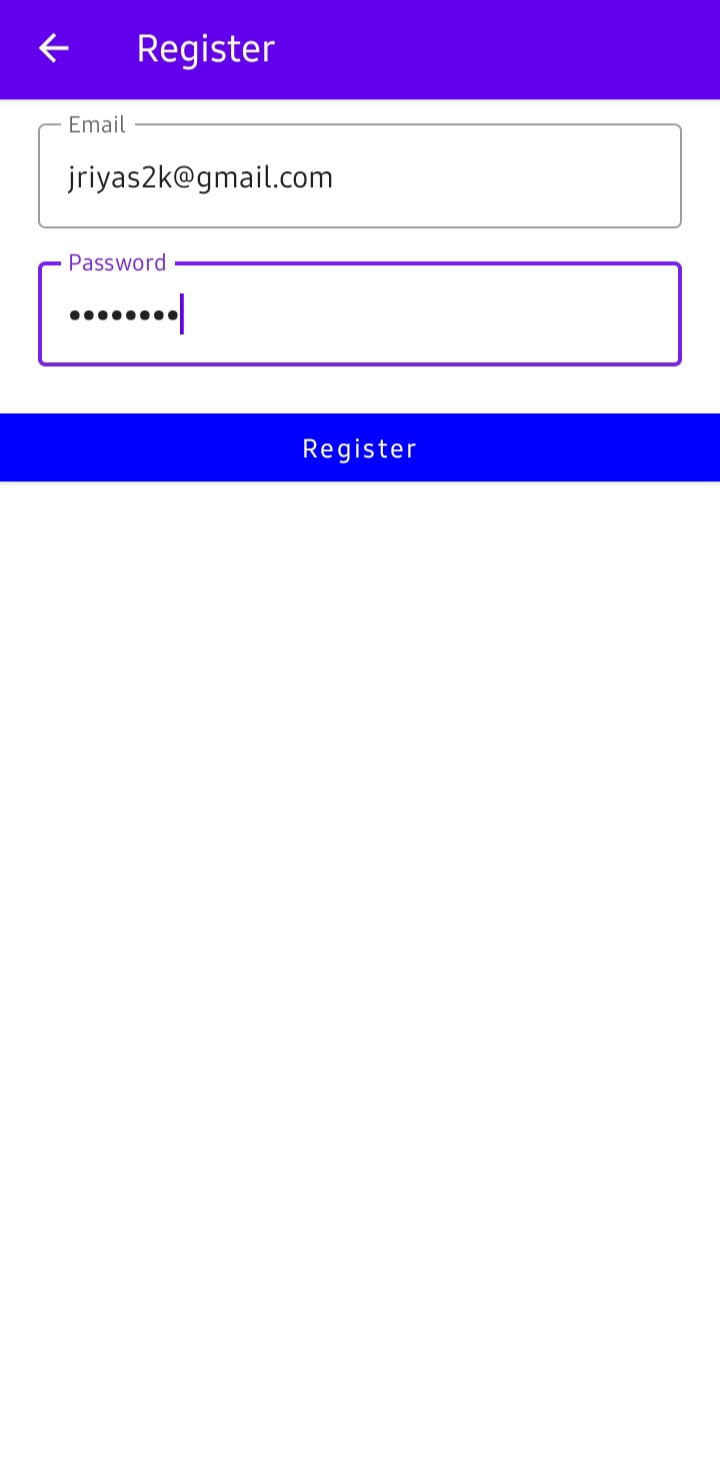


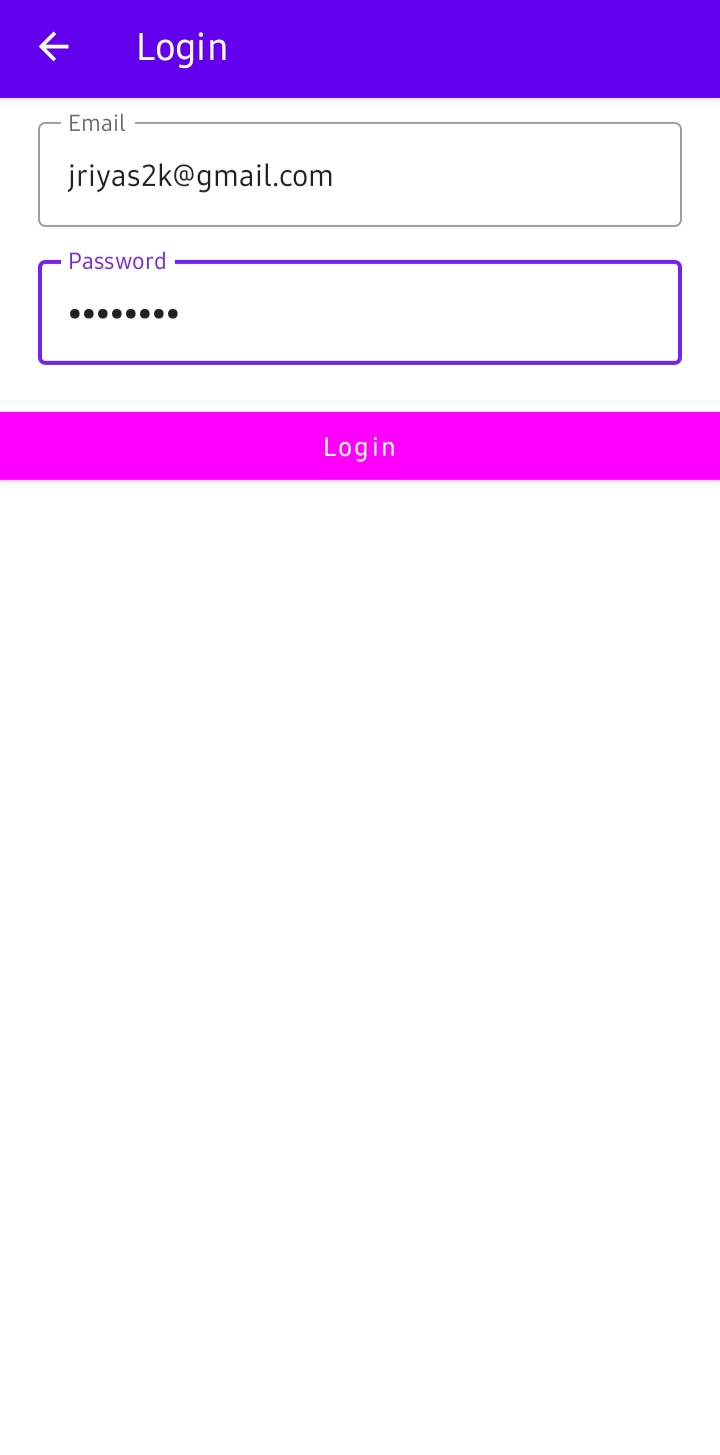
**2.2 Brainstorming Map:-**

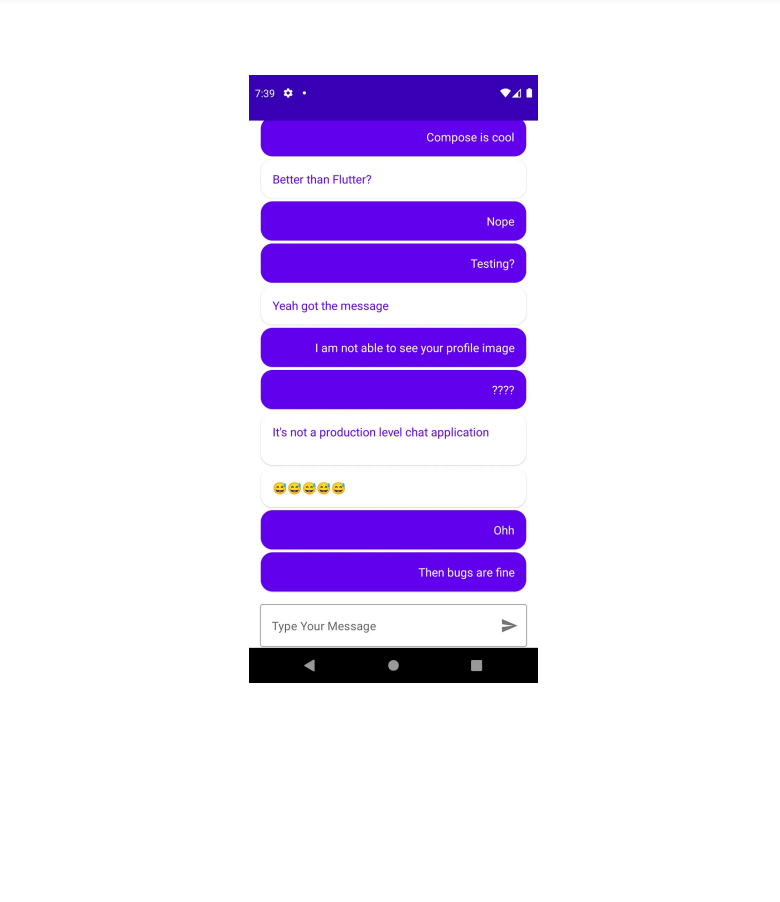


1. **Result :-**









**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.

**Disadvantage :-**

* 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.

1. **Application :-**

The materials and methods section should include **a clear and brief description of your research procedures**. One important purpose of this section is to convince the readers that your work is valid.

Another purpose is for researchers to use your methodology to guide his or her own experiments.

**6. Conclusion :-**

From our research we can conclude that, the invention of computers and internet has led to a technological boom in the world. Internet is being used as a backbone for social media applications to interact with humans living distances apart from each other.

We have learned a lot from the development of this project. We got to know more closely, how networking works and how data transfer takes place

**8.Appendix**

**Source code :-**

<https://github.com/smartinternz02/Chat-Connect-App>

**code :-**

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"  
 )  
 **}  
 }** )  
 **}**}

logint.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.  
 \*/*@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  
 )  
 **}  
 }**}

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  
 )  
 **}  
 }**}

mainactivity.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)  
 **}  
 }  
 }**}