CHAT CONNECT – A REAL TIME CHAT AND COMMUNICATION APP

INTRODUCTION

1.1 OVERVIEW

Chat Connect is a sample project built using the Android Compose UI toolkit. It demonstrates how to create a simple chat app using the Compose libraries. The app allows users to send and receive text messages. The project showcases the use of Compose's declarative UI and state management capabilities. It also includes examples of how to handle input and navigation using composable functions and how to use data from a firebase to populate the UI.

1.2 PURPOSE

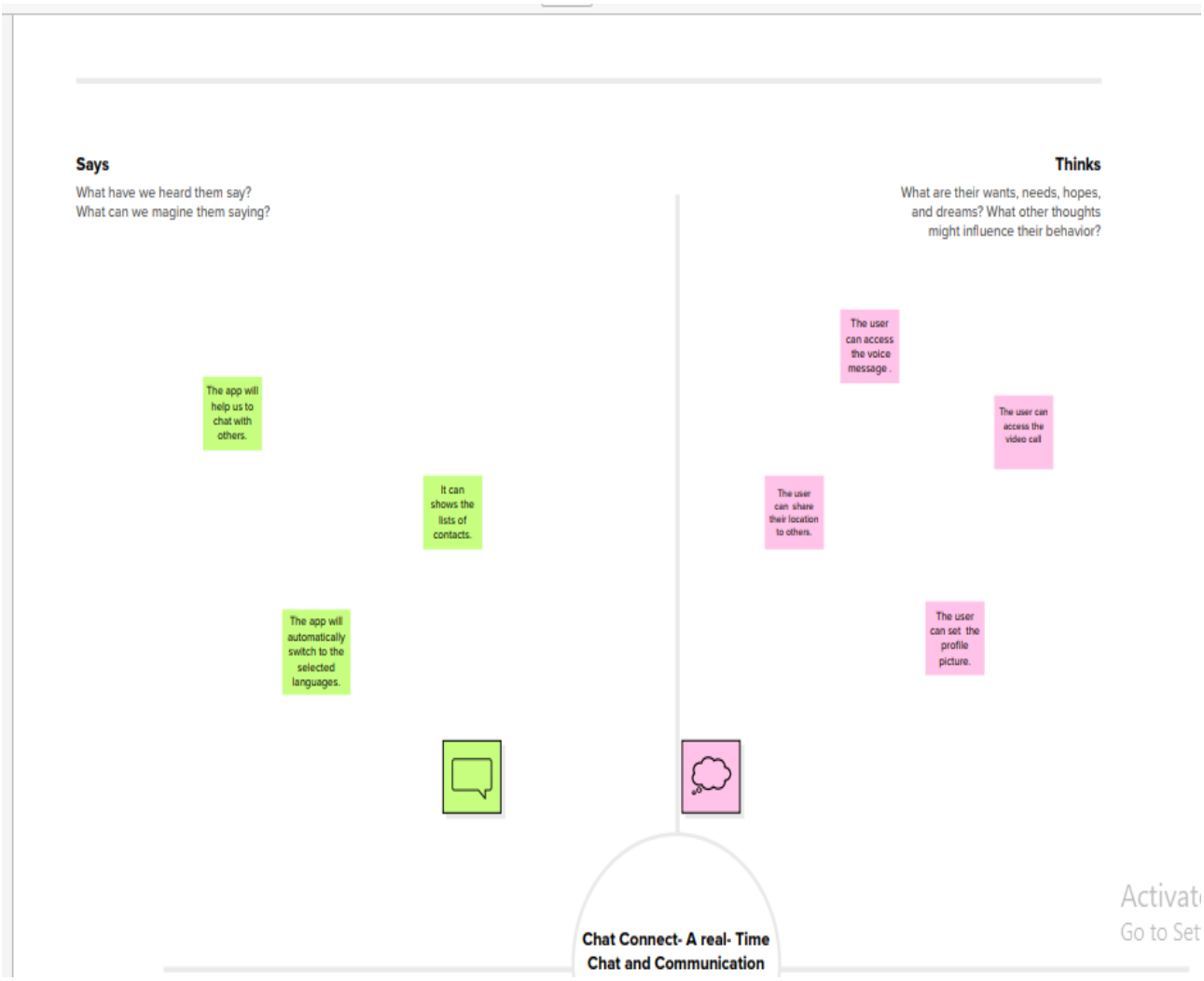
Real-time chat applications allow users to communicate with each other in real time through text, voice, or video. This type of app allows for more immediate messaging than other types of communication such as email or IM.

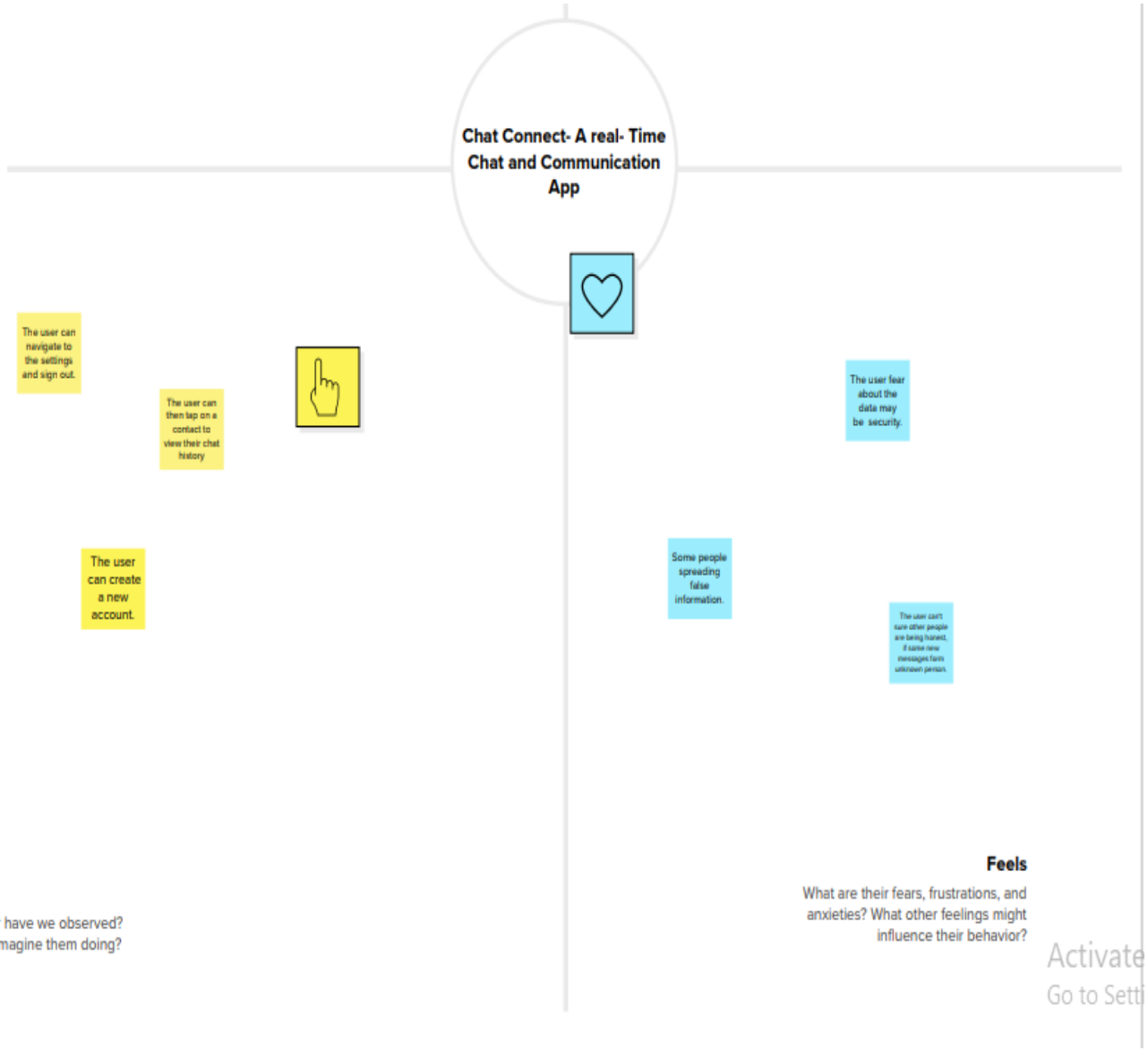
There are several reasons why chat applications must work in real time:

- **Improved performance**: More immediate communication allows for more natural conversation
- **Greater responsiveness**: Real-time functionality results in improved user experience
- **Superior reliability**: With real-time functionality there‘s less opportunity for messages to be lost or delayed

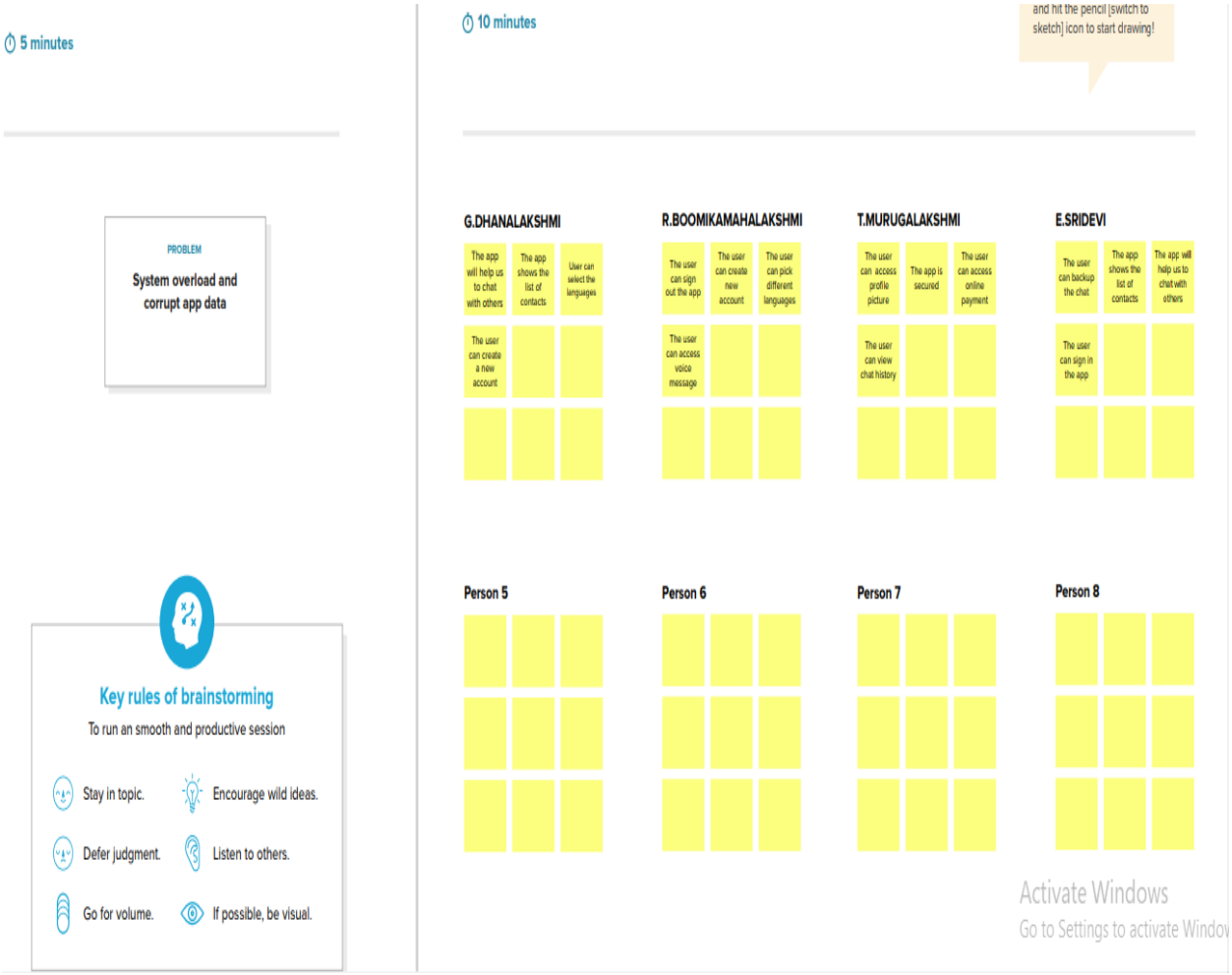
PROBLEM DEFINITION &DESIGN THINKING

2.1 EMPATHY MAP




Chat Connect- A real- Time Chat and Communication App
The user can navigate to the settings and sign out.
The user can then tap on a contact to view their chat history
The user can create a new account.
The user fear about the data may be security.
Some people spreading false information.
The user can't sure other people are being honest, if some new messages form unknown person.
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?
Activate
Go to Sett

2.2 IDEATION AND BRAINSTROM MAP



Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

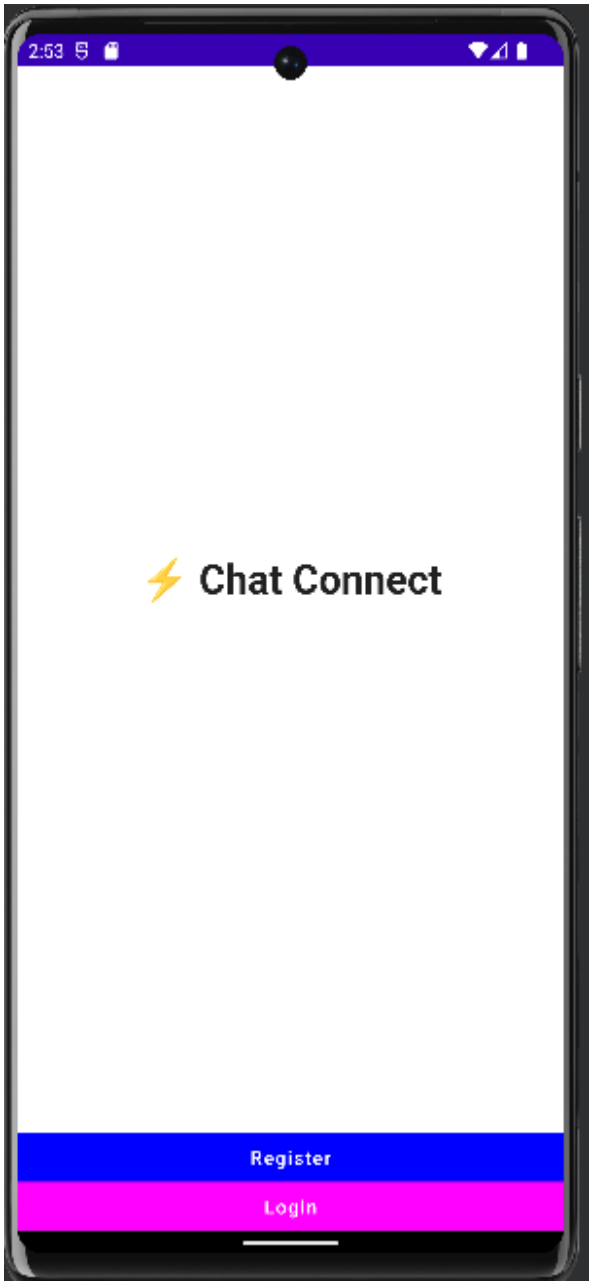
TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

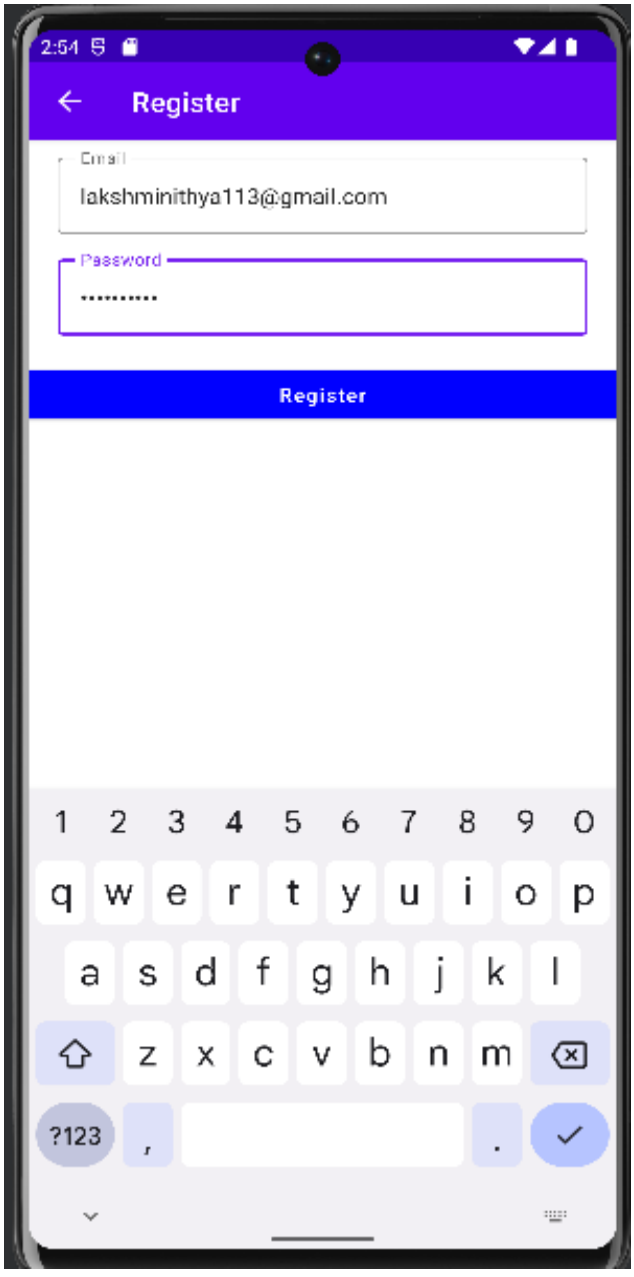
1.The best app
2.Sign in page is available
3.List of contacts
4.Settings added
5.User can select the languages
6.Chat history is available
7.Backup option is provided

RESULT

HOME PAGE



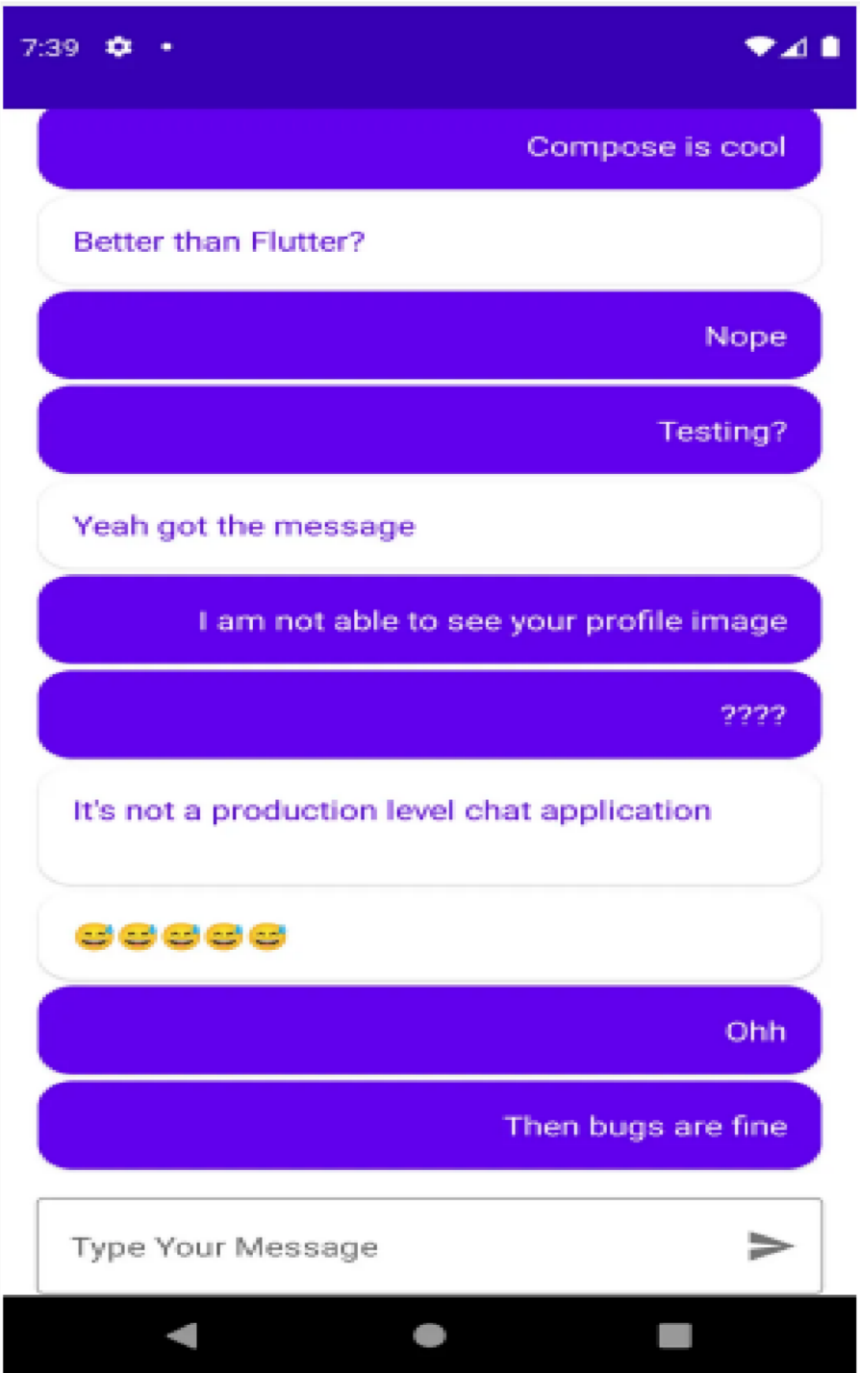
REGISTRATION PAGE



LOGIN PAGE


2:57
Login
Email
lakshminithya113@gmail.com
Password
Login
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123

CHATTING PAGE



ADVANTAGES & DISADVANTAGES

Advantages

- Chat is very easy to reach for our customers.
- It is very easy to use and simple chat app.
- The app lets user's direct attention to online and communicate others.
- The app helps with customers with prospective and communicate with other's very confidence.
- It is very faster and reach the information to others by this app.

Disadvantages

- The user can't sure other people are being honest or that they are whose.
- If the user feeling vulnerable, people online might try to take advantages of some ones.
- Building the relationships online can result in user spending less time with friends and family.
- There is no assurity about the personal data is secured.
- Lots of internet addiction due to chatting.

APPLICATIONS

Chatbot allow businesses to connect with customers in a personal way without the expense of human representatives. For example, many of the questions or issues customers have are common and easily answered.

Chatbot provide a personal alternative to a written FAQ or guide and can even triage questions, including handing off a customer issue to a live person if the issue becomes too complex for the chatbot to resolve. Chatbots have become; popular as a time and money saver for businesses and an added convenience for customers

CONCLUSION

You did it! You learned real time and several techniques you can use to go about doing real time communication. Before we wrap up, I want to talk about a few additional pieces of connections you can do that we didn’t talk about.ws is the other leading implementation of Web Sockets for Node.js and a damn good one. Frankly most of the time I choose it over Socket.IO because it’s more minimal and I don’t need the richness of what Socket.IO offers all the time. But honestly both are great and valid decisions. ws is nice because it doesn’t have a client; you just use the same new WebSocket () call we did on the client because that’s more than enough for just WebSocket usage. However, if you need all that retry logic, it’s hard to beat Socket.IO. It’s good for you to give this one a try too. Think of this as a one-way socket. Your client can connect to a server and the server can push many messages to the client. The difference here is that messages don’t flow the opposite way: your client can't use the same connection to push messages back. In the terms of the app we just built, we could start a long-running HTTP2 connection and use that to get updates on new messages but just a normal RESTful POST back to the API to post a new message. Perfectly great architecture decision.

FUTURE SCOPE

- Extending this application by providing Authorisation service.
- Creating Database and maintain users.
- Increasing the effectiveness of the application by providing voice chat.
- Extending it to Web Support.
- A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time.

A. SOURCE CODE

```
import android.os.Bundle import
androidx.activity.ComponentActivity import
androidx.activity.compose.setContent import
androidx.compose.foundation.layout.Arrangement import
androidx.compose.foundation.layout.Column import
androidx.compose.foundation.layout.Row import
androidx.compose.foundation.layout.fillMaxSize import
androidx.compose.foundation.layout.padding import
androidx.compose.material3.Button import
androidx.compose.material3.ElevatedButton import
androidx.compose.material3.MaterialTheme import
androidx.compose.material3.Surface import
androidx.compose.material3.Text import
androidx.compose.runtime.Composable import
androidx.compose.runtime.getValue import
androidx.compose.runtime.mutableStateOf import
androidx.compose.runtime.remember import
androidx.compose.runtime.setValue import
androidx.compose.ui.Alignment import
androidx.compose.ui.Modifier import
androidx.compose.ui.tooling.preview.Preview import
androidx.compose.ui.unit.dp import
com.codelab.basics.ui.theme.BasicsCodelabTheme
```

```
class MainActivity : ComponentActivity() {     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)        setContent {
          BasicsCodelabTheme {
             MyApp(modifier = Modifier.fillMaxSize())
          }
       }
   }
}

@Composable
fun MyApp(modifier: Modifier = Modifier) {

   var shouldShowOnboarding by remember { mutableStateOf(true) }

   Surface(modifier) {        if
(shouldShowOnboarding) {
          OnboardingScreen(onContinueClicked = { shouldShowOnboarding = false })
       } else {
          Greetings()
       }
   }
}

@Composable fun
OnboardingScreen(
```

```
onContinueClicked: () -> Unit,
modifier: Modifier = Modifier
) {

   Column(         modifier = modifier.fillMaxSize(),
verticalArrangement = Arrangement.Center,
horizontalAlignment = Alignment.CenterHorizontally
   ) {
      Text("Welcome to the Basics Codelab!")
Button(             modifier = Modifier.padding(vertical
= 24.dp),            onClick = onContinueClicked
      ) {
         Text("Continue")
      }
   }
}

@Composable
private fun Greetings(     modifier: Modifier =
Modifier,     names: List<String> = listOf("World",
"Compose")
) {
   Column(modifier = modifier.padding(vertical = 4.dp)) {
for (name in names) {
         Greeting(name = name)
      }
```

```
    }
}

@Preview(showBackground = true, widthDp = 320, heightDp = 320)
@Composable fun
OnboardingPreview() {
    BasicsCodelabTheme {
        OnboardingScreen(onContinueClicked = {})
    }
}

@Composable
private fun Greeting(name: String) {

    val expanded = remember { mutableStateOf(false) }

    val extraPadding = if (expanded.value) 48.dp else 0.dp

    Surface(        color = MaterialTheme.colorScheme.primary,
modifier = Modifier.padding(vertical = 4.dp, horizontal = 8.dp)
    ) {
        Row(modifier = Modifier.padding(24.dp)) {
            Column(modifier = Modifier
                .weight(1f)
                .padding(bottom = extraPadding)
            ) {
```

```
            Text(text = "Hello, ")
            Text(text = name)
        }
        ElevatedButton(
            onClick = { expanded.value = !expanded.value }
        ) {
            Text(if (expanded.value) "Show less" else "Show more")
        }
    }
  }
}

@Preview(showBackground = true, widthDp = 320)
@Composable
fun DefaultPreview() {
  BasicsCodelabTheme {
    Greetings()
  }
}

@Preview
@Composable fun
MyAppPreview() {
  BasicsCodelabTheme {
    MyApp(Modifier.fillMaxSize())
  }
```

```
}
import android.os.Bundle import
androidx.activity.ComponentActivity import
androidx.activity.compose.setContent import
androidx.compose.animation.core.Spring import
androidx.compose.animation.core.animateDpAsState import
androidx.compose.animation.core.spring import
androidx.compose.foundation.layout.Arrangement import
androidx.compose.foundation.layout.Column import
androidx.compose.foundation.layout.Row import
androidx.compose.foundation.layout.fillMaxSize import
androidx.compose.foundation.layout.padding import
androidx.compose.foundation.lazy.LazyColumn import
androidx.compose.foundation.lazy.items import
androidx.compose.material3.Button import
androidx.compose.material3.ElevatedButton import
androidx.compose.material3.MaterialTheme import
androidx.compose.material3.Surface import
androidx.compose.material3.Text import
androidx.compose.runtime.Composable import
androidx.compose.runtime.getValue import
androidx.compose.runtime.mutableStateOf import
androidx.compose.runtime.remember import
androidx.compose.runtime.saveable.rememberSaveable
import androidx.compose.runtime.setValue import
```

```
androidx.compose.ui.Alignment import
androidx.compose.ui.Modifier import
androidx.compose.ui.tooling.preview.Preview import
androidx.compose.ui.unit.dp import
com.codelab.basics.ui.theme.BasicsCodelabTheme

class MainActivity : ComponentActivity() {     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)

      setContent {
         BasicsCodelabTheme {
            MyApp(modifier = Modifier.fillMaxSize())
         }
      }
   }
}

@Composable
fun MyApp(modifier: Modifier = Modifier) {

   var shouldShowOnboarding by rememberSaveable { mutableStateOf(true) }

   Surface(modifier) {        if
(shouldShowOnboarding) {
         OnboardingScreen(onContinueClicked = { shouldShowOnboarding = false })
```

```
        } else {
            Greetings()
        }
    }
}

@Composable fun
OnboardingScreen(
onContinueClicked: () -> Unit,
modifier: Modifier = Modifier
) {


    Column(         modifier = modifier.fillMaxSize(),
verticalArrangement = Arrangement.Center,
horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text("Welcome to the Basics Codelab!")
Button(             modifier = Modifier.padding(vertical
= 24.dp),            onClick = onContinueClicked
        ) {
            Text("Continue")
        }
    }
}
```

```
@Composable
private fun Greetings(     modifier: Modifier
= Modifier,     names: List<String> =
List(1000) { "$it" }
) {
   LazyColumn(modifier = modifier.padding(vertical = 4.dp)) {
items(items = names) { name ->
          Greeting(name = name)
       }
   }
}

@Preview(showBackground = true, widthDp = 320, heightDp = 320)
@Composable fun
OnboardingPreview() {
   BasicsCodelabTheme {
      OnboardingScreen(onContinueClicked = {})
   }
}

@Composable
private fun Greeting(name: String) {

   var expanded by remember { mutableStateOf(false) }
```

```
    val extraPadding by animateDpAsState(        if (expanded)
48.dp else 0.dp,        animationSpec = spring(
dampingRatio = Spring.DampingRatioMediumBouncy,
stiffness = Spring.StiffnessLow
        )
    )
    Surface(        color = MaterialTheme.colorScheme.primary,
modifier = Modifier.padding(vertical = 4.dp, horizontal = 8.dp)
    ) {
        Row(modifier = Modifier.padding(24.dp)) {
            Column(modifier = Modifier
                .weight(1f)
                .padding(bottom = extraPadding.coerceAtLeast(0.dp))
            ) {
                Text(text = "Hello, ")
                Text(text = name)
            }
            ElevatedButton(                onClick = {
expanded = !expanded }
            ) {
                Text(if (expanded) "Show less" else "Show more")
            }
        }
    }
}
```

```
@Preview(showBackground = true, widthDp = 320)
@Composable
fun DefaultPreview() {
    BasicsCodelabTheme {
        Greetings()
    }
}

@Preview
@Composable fun
MyAppPreview() {
    BasicsCodelabTheme {
        MyApp(Modifier.fillMaxSize())
    }
}
import android.app.Activity import
android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme import
androidx.compose.material3.darkColorScheme import
androidx.compose.material3.dynamicDarkColorScheme import
androidx.compose.material3.dynamicLightColorScheme import
androidx.compose.material3.lightColorScheme import
androidx.compose.runtime.Composable import
androidx.compose.runtime.SideEffect import
androidx.compose.ui.graphics.Color import
```

```
androidx.compose.ui.graphics.toArgb import
androidx.compose.ui.platform.LocalContext import
androidx.compose.ui.platform.LocalView import
androidx.core.view.ViewCompat


private val DarkColorScheme = darkColorScheme(
surface = Blue,     onSurface = Navy,     primary =
Navy,     onPrimary = Chartreuse
)


private val LightColorScheme = lightColorScheme(
surface = Blue,     onSurface = Color.White,
primary = LightBlue,

   onPrimary = Navy
)
@Composable fun BasicsCodelabTheme(
darkTheme: Boolean = isSystemInDarkTheme(),
// Dynamic color is available on Android 12+
dynamicColor: Boolean = true,     content:
@Composable () -> Unit
) {     val colorScheme = when {         dynamicColor && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme) dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }
```

```
        darkTheme -> DarkColorScheme
else -> LightColorScheme
    }
    val view = LocalView.current
if (!view.isInEditMode) {
        SideEffect {
            (view.context as Activity).window.statusBarColor =
colorScheme.primary.toArgb()
            ViewCompat.getWindowInsetsController(view)?.isAppearanceLightStatusBars =
darkTheme
        }
    }
    MaterialTheme(
        colorScheme = colorScheme,
typography = Typography,
content = content
    )
}
import android.content.res.Configuration.UI_MODE_NIGHT_YES
import android.os.Bundle import
androidx.activity.ComponentActivity import
androidx.activity.compose.setContent import
androidx.compose.animation.animateContentSize import
androidx.compose.animation.core.Spring import
androidx.compose.animation.core.spring import
androidx.compose.foundation.layout.Arrangement import
```

androidx.compose.foundation.layout.Column import

androidx.compose.foundation.layout.Row import

androidx.compose.foundation.layout.fillMaxSize import

androidx.compose.foundation.layout.padding import

androidx.compose.foundation.lazy.LazyColumn import

androidx.compose.foundation.lazy.items import

androidx.compose.material.icons.Icons.Filled import

androidx.compose.material.icons.filled.ExpandLess import

androidx.compose.material.icons.filled.ExpandMore import

androidx.compose.material3.Button import

androidx.compose.material3.Card import

androidx.compose.material3.CardDefaults import

androidx.compose.material3.Icon import

androidx.compose.material3.IconButton import

androidx.compose.material3.MaterialTheme

import androidx.compose.material3.Surface import

androidx.compose.material3.Text import

androidx.compose.runtime.Composable import

androidx.compose.runtime.getValue import

androidx.compose.runtime.mutableStateOf import

androidx.compose.runtime.remember import

androidx.compose.runtime.saveable.rememberSaveable import

androidx.compose.runtime.setValue import

androidx.compose.ui.Alignment import

androidx.compose.ui.Modifier import

```
androidx.compose.ui.res.stringResource import
androidx.compose.ui.text.font.FontWeight import
androidx.compose.ui.tooling.preview.Preview import
androidx.compose.ui.unit.dp import
com.codelab.basics.ui.theme.BasicsCodelabTheme

class MainActivity : ComponentActivity() {     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)        setContent {
          BasicsCodelabTheme {
             MyApp(modifier = Modifier.fillMaxSize())
          }
       }
   }
}

@Composable
fun MyApp(modifier: Modifier = Modifier) {     var shouldShowOnboarding
by rememberSaveable { mutableStateOf(true) }

   Surface(modifier, color = MaterialTheme.colorScheme.background) {
if (shouldShowOnboarding) {
          OnboardingScreen(onContinueClicked = { shouldShowOnboarding = false })
       } else {
          Greetings()
       }
```

```
    }
}

@Composable fun
OnboardingScreen(
onContinueClicked: () -> Unit,
modifier: Modifier = Modifier
) {
    Column(         modifier = modifier.fillMaxSize(),
verticalArrangement = Arrangement.Center,
horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text("Welcome to the Basics Codelab!")
Button(             modifier = Modifier.padding(vertical
= 24.dp),            onClick = onContinueClicked
        ) {
            Text("Continue")
        }
    }
}

@Composable
private fun Greetings(      modifier: Modifier
= Modifier,      names: List<String> =
List(1000) { "$it" }
) {
```

```
    LazyColumn(modifier = modifier.padding(vertical = 4.dp)) {
items(items = names) { name ->
            Greeting(name = name)
        }
    }
}

@Composable
private fun Greeting(name: String) {     Card(         colors =
CardDefaults.cardColors(             containerColor =
MaterialTheme.colorScheme.primary
        ),
        modifier = Modifier.padding(vertical = 4.dp, horizontal = 8.dp)
    ) {
        CardContent(name)
    }
}

@Composable
private fun CardContent(name: String) {     var
expanded by remember { mutableStateOf(false) }

    Row(         modifier = Modifier             .padding(12.dp)
.animateContentSize(                 animationSpec = spring(
dampingRatio = Spring.DampingRatioMediumBouncy,
stiffness = Spring.StiffnessLow
```

```
            )
        )
    ) {
        Column(
modifier = Modifier
.weight(1f)
            .padding(12.dp)
        ) {
            Text(text = "Hello, ")
Text(
            text = name, style = MaterialTheme.typography.headlineMedium.copy(
fontWeight = FontWeight.ExtraBold
            )
        )
        if (expanded) {                Text(                    text
= ("Composem ipsum color sit lazy, " +
"padding theme elit, sed do bouncy. ").repeat(4),
            )
        }
    }
    IconButton(onClick = { expanded = !expanded }) {
        Icon(
            imageVector = if (expanded) Filled.ExpandLess else
Filled.ExpandMore,                contentDescription = if (expanded) {
stringResource(R.string.show_less)
```

```
            } else {
stringResource(R.string.show_more)
            }
        )
    }
  }
}

@Preview(
showBackground = true,
widthDp = 320,
  uiMode = UI_MODE_NIGHT_YES,
name = "DefaultPreviewDark"
)
@Preview(showBackground = true, widthDp = 320)
@Composable
fun DefaultPreview() {
  BasicsCodelabTheme {
    Greetings()
  }
}

@Preview(showBackground = true, widthDp = 320, heightDp = 320)
@Composable fun
OnboardingPreview() {
  BasicsCodelabTheme {
```

```
        OnboardingScreen(onContinueClicked = {})
    }
}

@Preview
@Composable fun
MyAppPreview() {
    BasicsCodelabTheme {
        MyApp(Modifier.fillMaxSize())
    }
}
```