CHAT CONNECT – A REAL TIME CHAT AND COMMUNICATION APP

INTRODUCTION

1.1 Overview

Project Description

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

Chatbots are conversational tools that perform routine tasks efficiently. People like them because they help them get through those tasks quickly so they can focus their attention on high-level, strategic, and engaging activities that require human capabilities that cannot be replicated by machines.

Chatbot use cases in customer service

- Chatbots answer questions and inquiries. ...
- Book tickets to events/shows with chatbots. ...
- Use chatbots to find products, check inventory and recommend items. ...
- Chatbots to build remarkable customer experience. ...
- Chatbots can process return and exchange requests.

Learning Outcomes

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.

PROBLEM DEFINITION & DESIGN THINKING

Problem Definition

Problem Statement of this project Email, newsgroup and messaging applications provide means for communication among people but these are one-way mechanisms and they do not provide an easy way to carry on a real-time conversation or discussion with people involved. Chat room extends the one-way messaging concept to accommodate multi-way communication among a set of people.

2.1 Empathy Map

Figure 2.1 Empathy Map

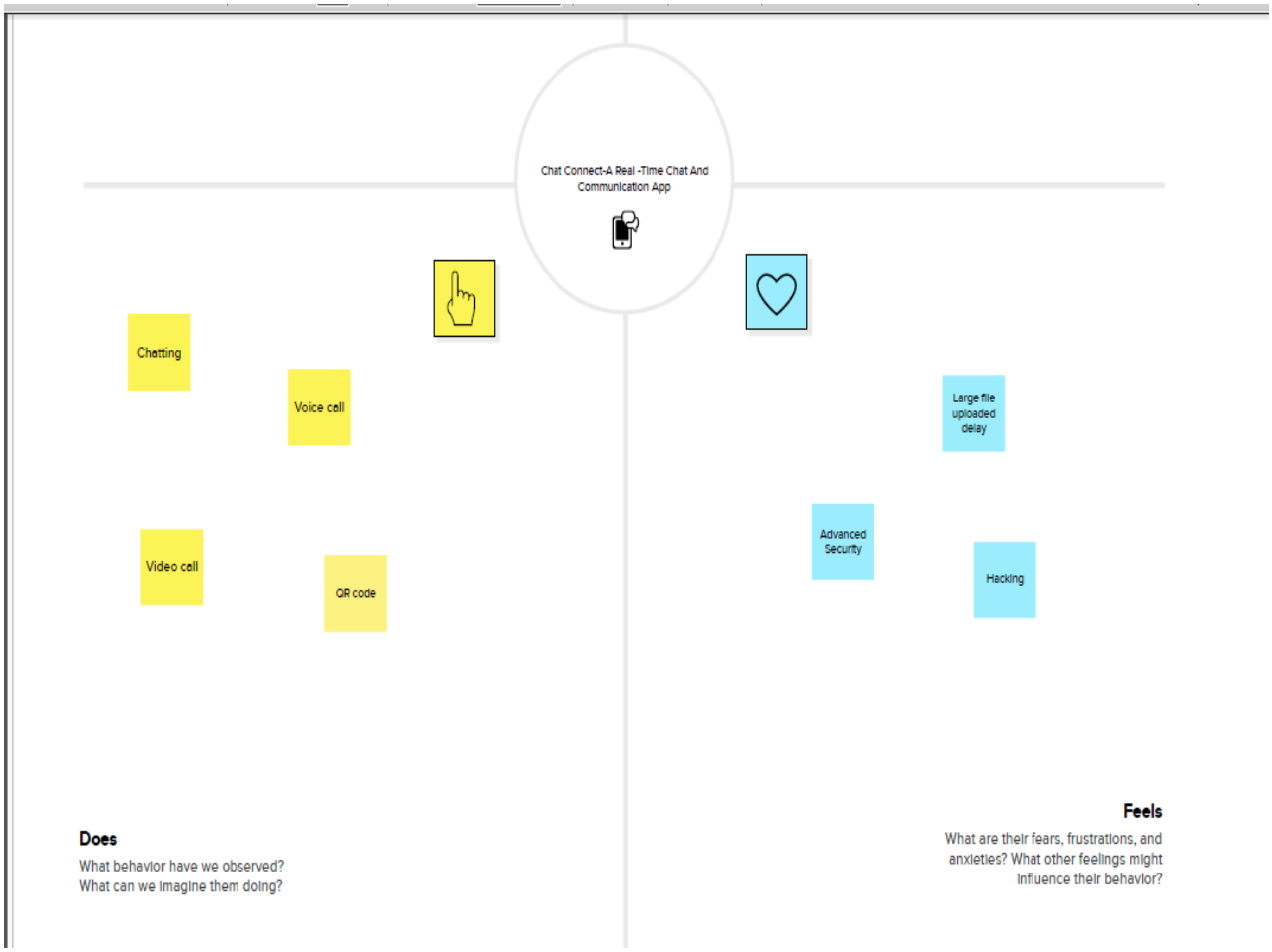

Figure 2.2 Empathy Map

2.2 Brainstorming Map

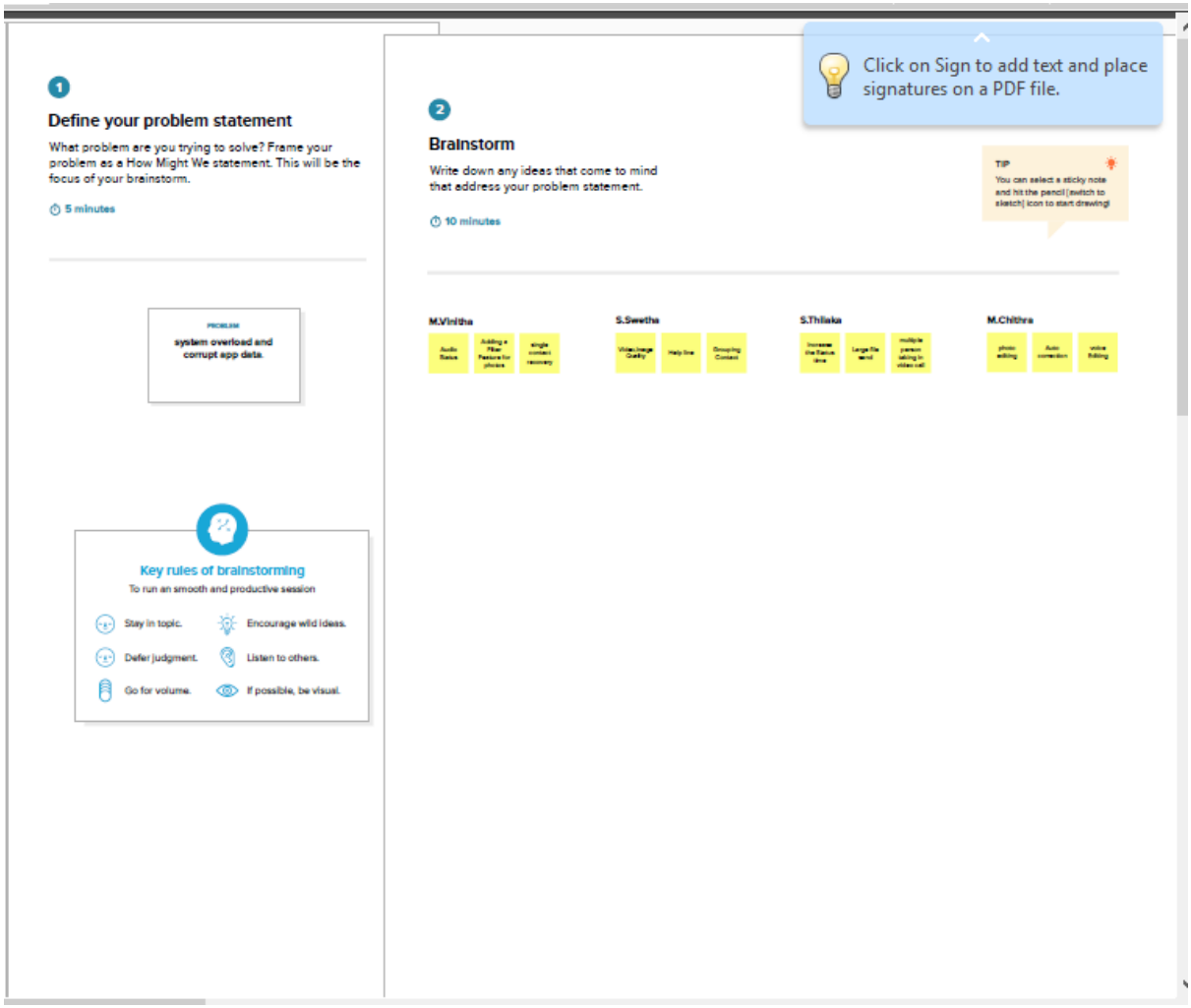

Figure 2.3 Brainstorming Map

CHAPTER 2 PROBLEM DEFINITION & DESIGN THINKING

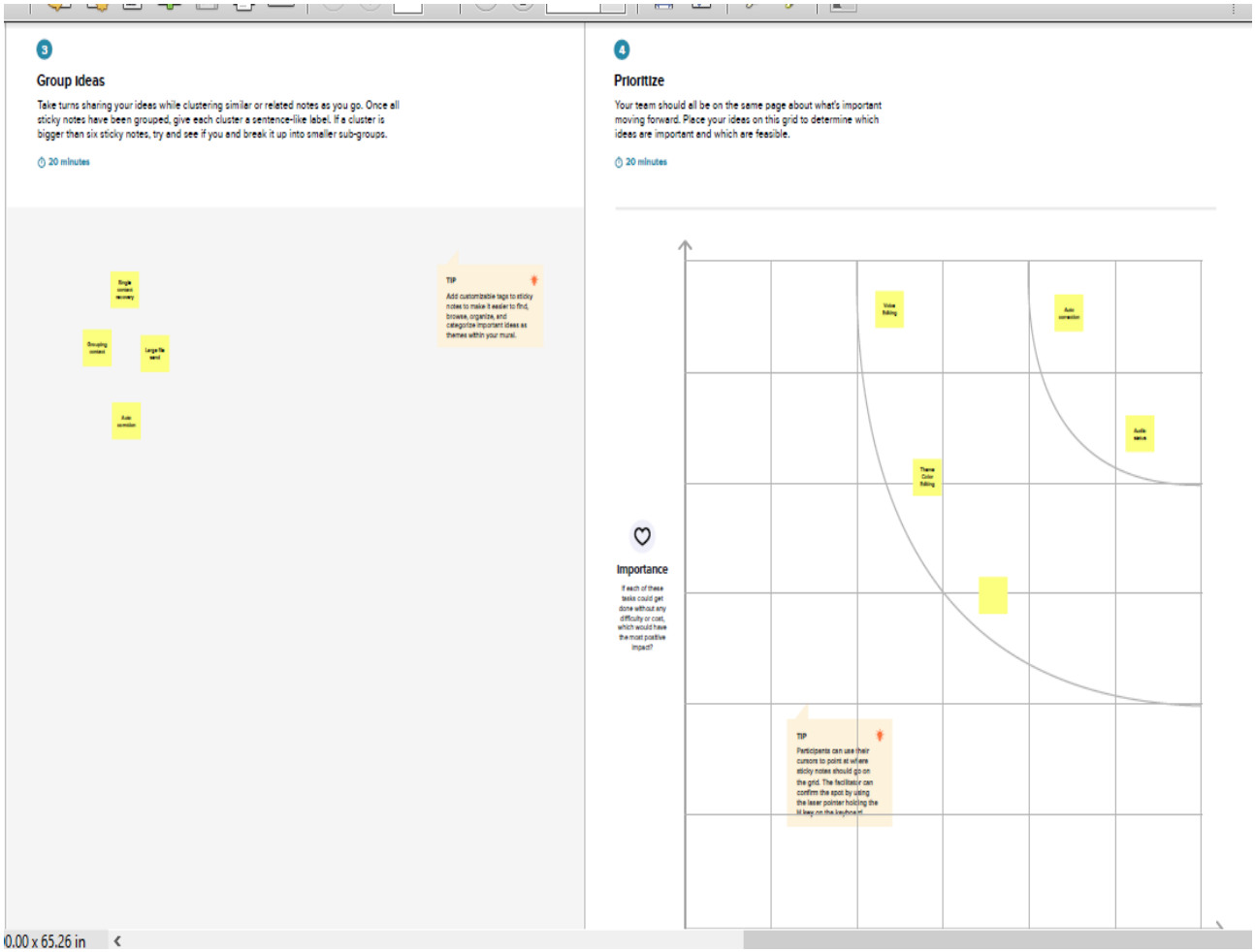

Figure 2.4 Brainstorming Map

SCREENSHOT

Home Page

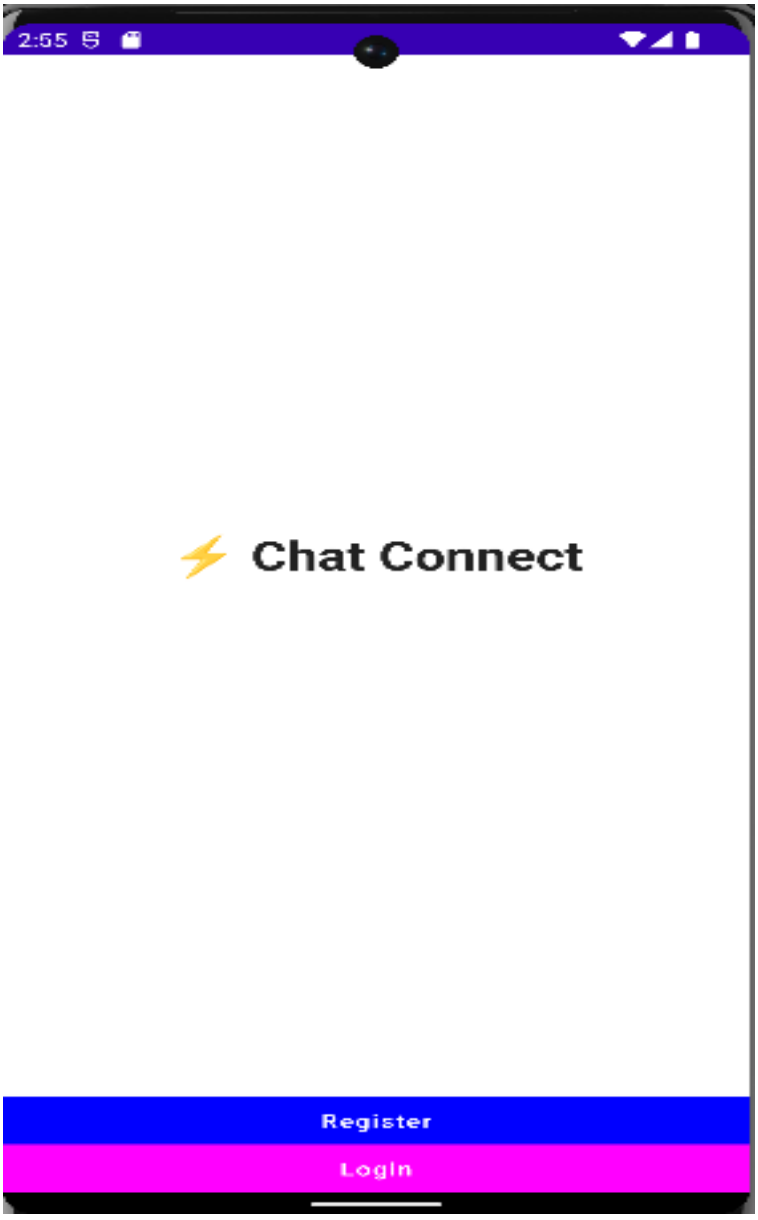

Figure 3.1: Home Page

Register Page

Figure 3.2: Register Page

Login Page

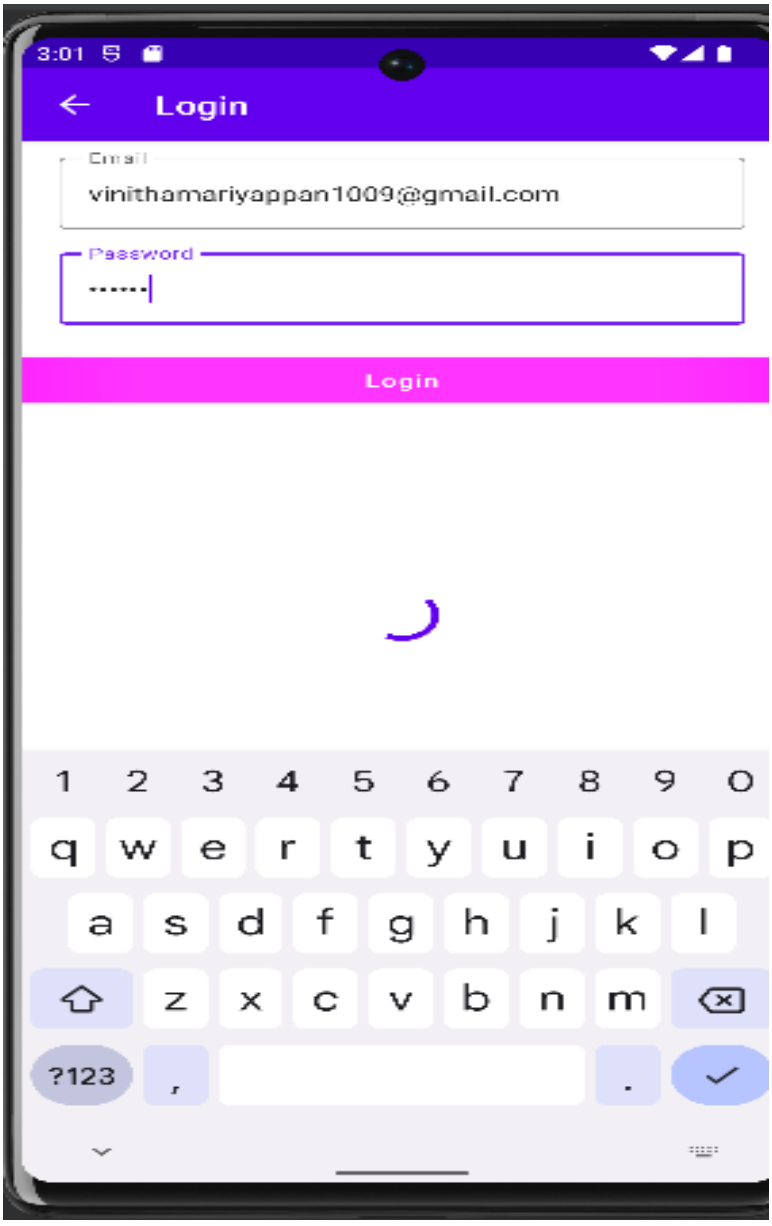

Figure 3.3: Login Page

Chatting Screen

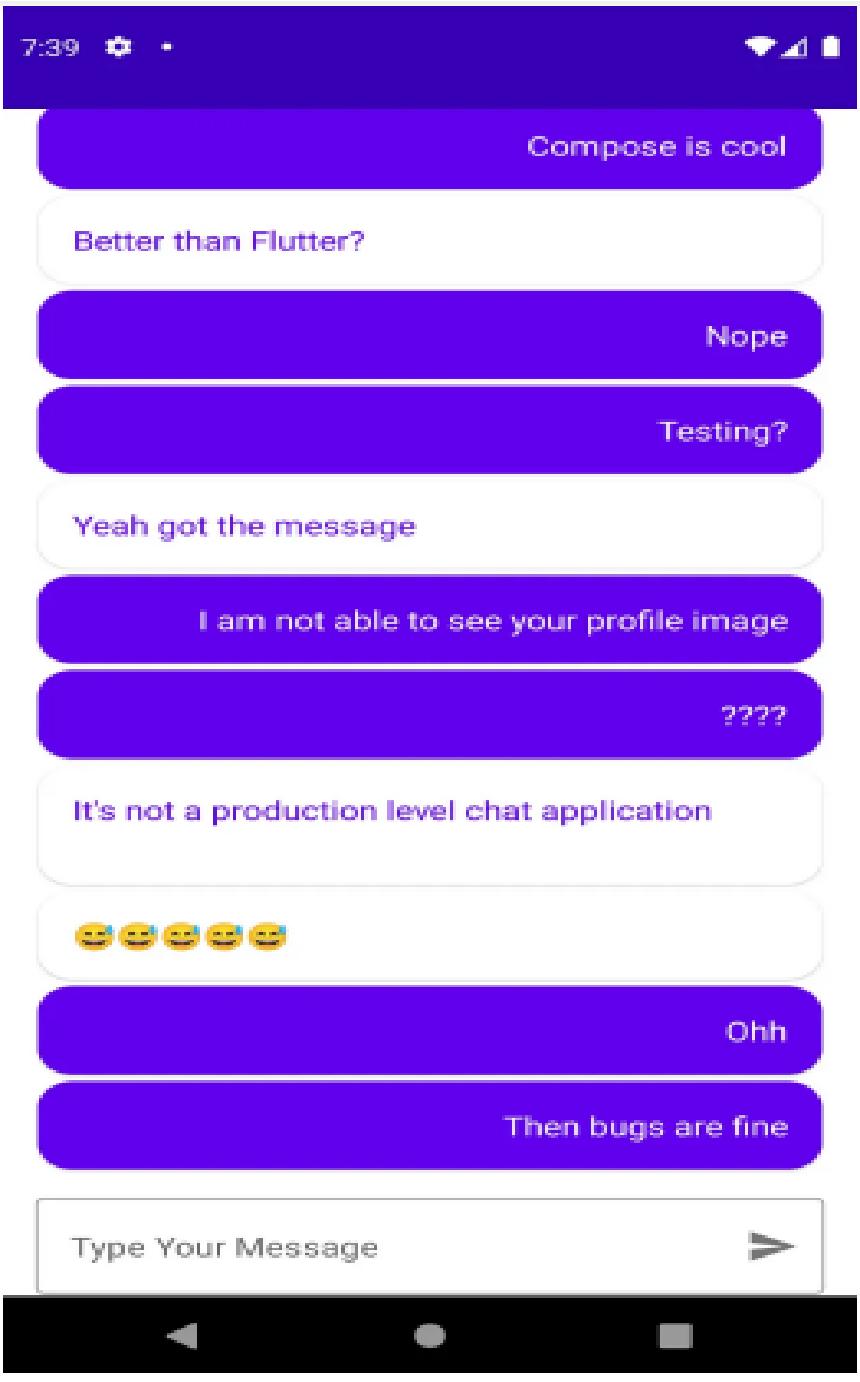

Figure 3.4: Chatting Screen

ADVANTAGES & DISADVANTAGES

ADVANTAGES

Faster support

Obviously chat is easy to reach for your customers, but what's more is that the average resolution time is significantly lower than with traditional service channels

Real-time text preview

One of the handy advantages of live chat is the option to see a real-time preview of what the customer is typing before she hits enter. It gives your chat agent the chance to think about a solution, research and impress users with prompt, customized answers

Instant customer feedback

Feedback is easily collectable. Users can rate your chat service right after their interaction with you. This is a benefit for your service agents too because they get instant feedback on their performance. This makes it easy for them to connect the dots.

Less drama

Chat agents experience less emotional exhaustion after difficult service interactions. It's easier to distance yourself from a mean text than a shouting customer hurling complaints.

Prevents agent fatigue

Avoid one of the main burnout reasons in customer service: boredom. By preparing templates (Chat Macros) for frequently asked questions, chat agents have more time to focus on complex issues and personal customer interactions.

No waiting queues

Customers reach you quicker. According to Zopim research, in less than 30 seconds. This is possible even for small service teams because agents can handle multiple requests at the same time. Research by Software Advice shows that the extinction of waiting queues is the primary reason why people prefer live chat.

Non-intrusive

Customers can go about their day while being helped. Live chat allows users to continue browsing, posting and working – your agent solves the case along the way. Over 51% of customers like live chat because it allows them to multitask.

On-site

Because the conversation takes place on the website, the context is mostly clear. During a chat, the service agent can see in the Message Center what page the visitor is currently looking at, and easily guide the user around the site with direct links or screenshots.

Low barrier

Customers find you right away. There's no need for them to waste time searching for contact details or email addresses. They simply land on your website and launch a session.

It reduces the risk of page bouncing because people can clear doubts and questions without hurdles.

Prevents cart abandonment

Customers can ask purchase-related questions just in time. According to a Forrester study, 44% of consumers say that having the ability to talk to an agent in the middle of an online purchase is one of the greatest live chat advantages. Bold Software came to similar conclusions in their study: Chatters are 1.8 times more likely to make online purchases in comparison.

Quick resolution of site errors

One of the less obvious benefits of live chat is its effectiveness in driving down user fails and site errors. Customers with forgotten passwords, or who are stumbling over 404 pages, are helped immediately. Annoying errors can be fixed before they cause greater damage.

Doesn’t work well for older demographics

While texting is second nature to most demographics, people 55+ are more unlikely to contact a company via chat. This is suggested by a 2015 research study by Software Advice. So, depending on your target group, an additional channel might be mandatory (phone, email).

The need to be online to offer support

Same as for phone, synchronous channels require people to be online. Having enough staff during peak hours can be costly. Though there is still an advantage to live chat compared to phone: the live chat window automatically turns into a contact form so visitors can easily contact you on every page of your website, even after hours.

First response time expectations are high

Ideally, a message in live chat is answered within 30-60 seconds. That requires the right amount of staffing, so that the first response time can be kept at an acceptable level.

This disadvantage is reduced since chat agents can take care of 10 customers simultaneously. For instance, if an agent is already serving two customers, he can easily send short greetings via chat macros to new customers. Alternatively a simple chatbot can take over the first greeting and ask for the issue.

Identity verification

In some cases like online banking, access to personal data is only possible by secure identity check. This might require telephone contact.

Online Trolls

Because you can chat quite anonymously, internet trolls are a phenomenon. That's why Userlike offers features like blocking and ignoring. To protect the privacy of operators, it is possible to use operator aliases.

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

The chat app provides a better and more flexible chat system. Developed with the latest technology in the way of providing a reliable system. The main advantage of the system is instant messaging, real-world communication, added security, group chat, etc. This application may find the best demand in the market for most organizations that aim to have independent applications.

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.

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())
    }
}
```