



Adaptive mail: A flexible Email Client App

By:
AronRekshi.A
Abilisha.P
Esaiyarasi.T
Abinesh.A
Ashwin.M

- ✓ The Adaptive Mail App is a comprehensive sample project that demonstrates the use of Android's Jetpack Compose UI toolkit to build a modern, conversational messaging interface.
- ✓ Messaging Simulation: The app simulates a messaging platform where users can send and receive messages in a chat-like environment, mimicking real-world communication apps
- ✓ Jetpack Compose UI: Leverages Jetpack Compose, a modern UI toolkit for Android, to create responsive, declarative, and flexible user interfaces with minimal code.
- ✓ State Management: Demonstrates how to effectively manage UI state in a Compose-based app, including handling incoming and outgoing messages, user interactions, and maintaining chat history.
- ✓ Conversation History: Allows users to view a scrollable history of previous messages, showcasing the use of dynamic content rendering and lists within Compose.
- ✓ Data Binding: Highlights how to bind UI components to data models, enabling easy updates to the UI based on changes to the underlying data.

- ✓ **User Interaction:** Includes features like message input, sending, and displaying real-time updates to simulate an interactive chat experience.
- ✓ **UI Customization:** Shows how Compose can be used to customize UI elements such as message bubbles, timestamps, and buttons for a personalized user experience.
- ✓ **Modern UI Components:** Uses advanced UI components like LazyColumn, which is optimized for performance and efficient rendering of long lists, such as the conversation history.
- ✓ **Adaptive Layouts:** Demonstrates responsive design principles by adapting the layout to different screen sizes and orientations, ensuring a seamless experience on phones, tablets, and foldable devices.
- ✓ **Best Practices:** Serves as an example of best practices for building scalable, maintainable Android applications using the latest tools and technologies in the Android ecosystem.

MAIN ACTIVITY JAVA

```
package com.example.emailapplication
```

```
import android.os.Bundle
```

```
import androidx.activity.ComponentActivity
```

```
import androidx.activity.compose.setContent
```

```
import androidx.activity.enableEdgeToEdge
```

```
import androidx.compose.foundation.layout.fillMaxSize
```

```
import androidx.compose.foundation.layout.padding
```

```
import androidx.compose.material3.Scaffold
```

```
import androidx.compose.material3.Text
```

```
import androidx.compose.runtime.Composable
```

```
import androidx.compose.ui.Modifier
```

```
import androidx.compose.ui.tooling.preview.Preview
```

```
import com.example.emailapplication.ui.theme.EmailApplicationTheme
```

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        enableEdgeToEdge()  
        setContent {  
            EmailApplicationTheme {  
                Scaffold(modifier = Modifier.fillMaxSize()) { innerPadding ->  
                    Greeting(  
                        name = "Android",  
                        modifier = Modifier.padding(innerPadding)  
                    )  
                }  
            }  
        }  
    }  
}
```

The background features several large, overlapping geometric shapes in shades of green and yellow. In the top-left, there's a large green triangle pointing downwards. In the top-right, a yellow diamond is partially visible. In the bottom-left, a green triangle points upwards. In the bottom-right, a yellow diamond is partially visible. A small green diamond is located near the bottom center, and a small yellow diamond is near the top center.

@Composable

```
fun Greeting(name: String, modifier: Modifier = Modifier) {  
    Text(  
        text = "Hello $name!",  
        modifier = modifier  
    )  
}
```

@Preview(showBackground = true)

@Composable

```
fun GreetingPreview() {  
    EmailApplicationTheme {  
        Greeting("Android")  
    }  
}
```

REGISTER ACTIVITY.KY

```
class MainActivity : ComponentActivity() {  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContent {  
            // A surface container using the 'background' color from the theme  
            Surface(  
                modifier = Modifier.fillMaxSize().background(Color.White),  
            ) {  
                Email(this)  
            }  
        }  
    }  
}  
  
@Composable  
fun Email(context: Context) {
```

```
Text(
  text = "Home Screen",
  modifier = Modifier.padding(top = 74.dp, start = 100.dp, bottom = 24.dp),
  color = Color.Black,
  fontWeight = FontWeight.Bold,
  fontSize = 32.sp
)
Column(
  horizontalAlignment = Alignment.CenterHorizontally,
  verticalArrangement = Arrangement.Center
) {
  Image(
    painterResource(id = R.drawable.home_screen), contentDescription = ""
  )
}
```



```
Button(onClick = {
    context.startActivity(
        Intent(
            context,
            SendMailActivity::class.java
        )
    )
},

colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))
) {
    Text(
        text = "Send Email",
        modifier = Modifier.padding(10.dp),
        color = Color.Black,
        fontSize = 15.sp
    )
}
```

The background features several large, overlapping geometric shapes in shades of green and yellow. In the top-left, there are green triangles. In the top-right, there is a large yellow triangle with a small yellow diamond inside it. In the bottom-left, there is a large green triangle and a yellow triangle. In the bottom-right, there is a green triangle. A small green diamond is located near the bottom center.

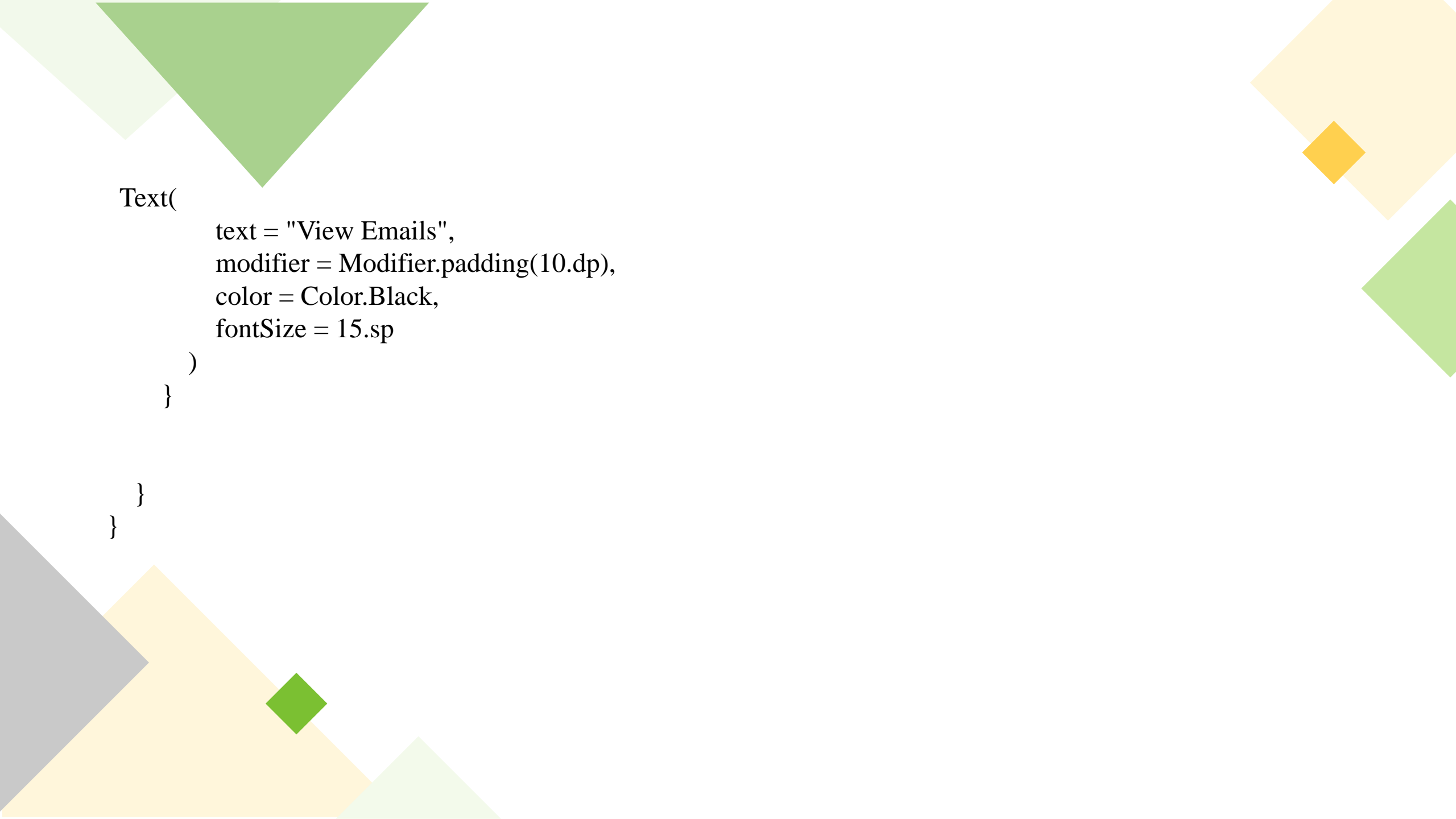
```
Spacer(modifier = Modifier.height(20.dp))
```

```
Button(onClick = {  
    context.startActivity(  
        Intent(  
            context,  
            ViewMailActivity::class.java
```

```
)  
    })  
},
```

```
colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFadbef4))  
)
```


```
{
```



```
Text(  
    text = "View Emails",  
    modifier = Modifier.padding(10.dp),  
    color = Color.Black,  
    fontSize = 15.sp  
)  
}  
}
```

LOGIN ACTIVITY.KT

```
class SendMailActivity : ComponentActivity() {  
    private lateinit var databaseHelper: EmailDatabaseHelper  
    @SuppressWarnings("UnusedMaterialScaffoldPaddingParameter")  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        databaseHelper = EmailDatabaseHelper(this)  
        setContent {  
  
            Scaffold(  
                // in scaffold we are specifying top bar.  
                topBar = {  
                    // inside top bar we are specifying  
                    // background color.  
                    TopAppBar(backgroundColor = Color(0xFFadbef4), modifier = Modifier.height(80.dp),
```



```
// along with that we are specifying
// title for our top bar.
title = {
  // in the top bar we are specifying
  // title as a text
  Text(
    // on below line we are specifying
    // text to display in top app bar.
    text = "Send Mail",
    fontSize = 32.sp,
    color = Color.Black,

    // on below line we are specifying
    // modifier to fill max width.
    modifier = Modifier.fillMaxWidth(),
```



```
// on below line we are
```

```
// specifying text alignment.
```

```
textAlign = TextAlign.Center,
```

```
)
```

```
}
```

```
)
```

```
}
```

```
) {
```

```
// on below line we are
```

```
// calling method to display UI.
```

```
openEmailer(this, databaseHelper)
```

```
}
```

```
}
```

```
}
```

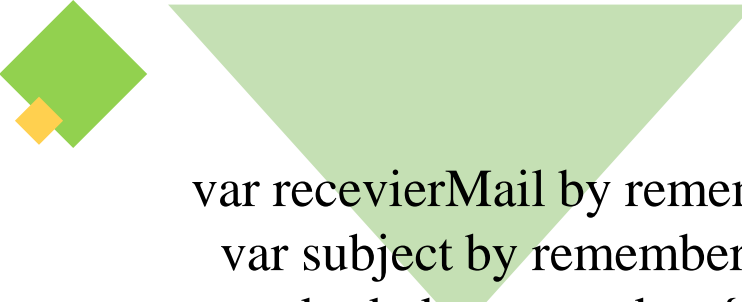
```
}
```

```
@Composable
```

```
fun openEmailer(context: Context, databaseHelper: EmailDatabaseHelper) {
```

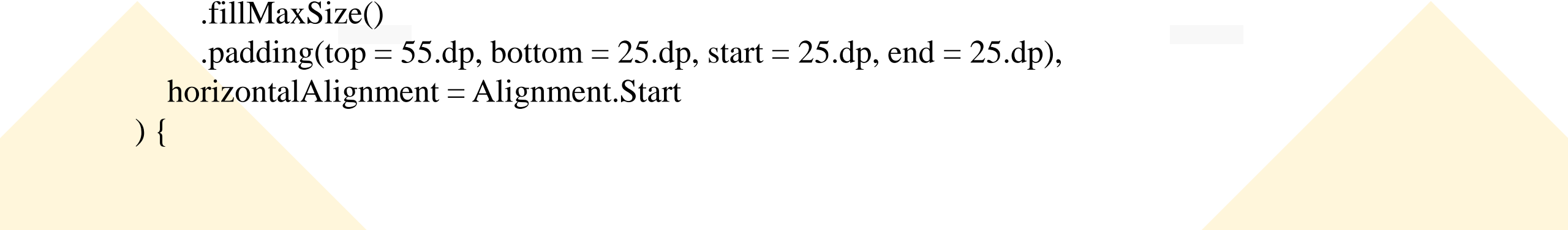
```
// in the below line, we are
```

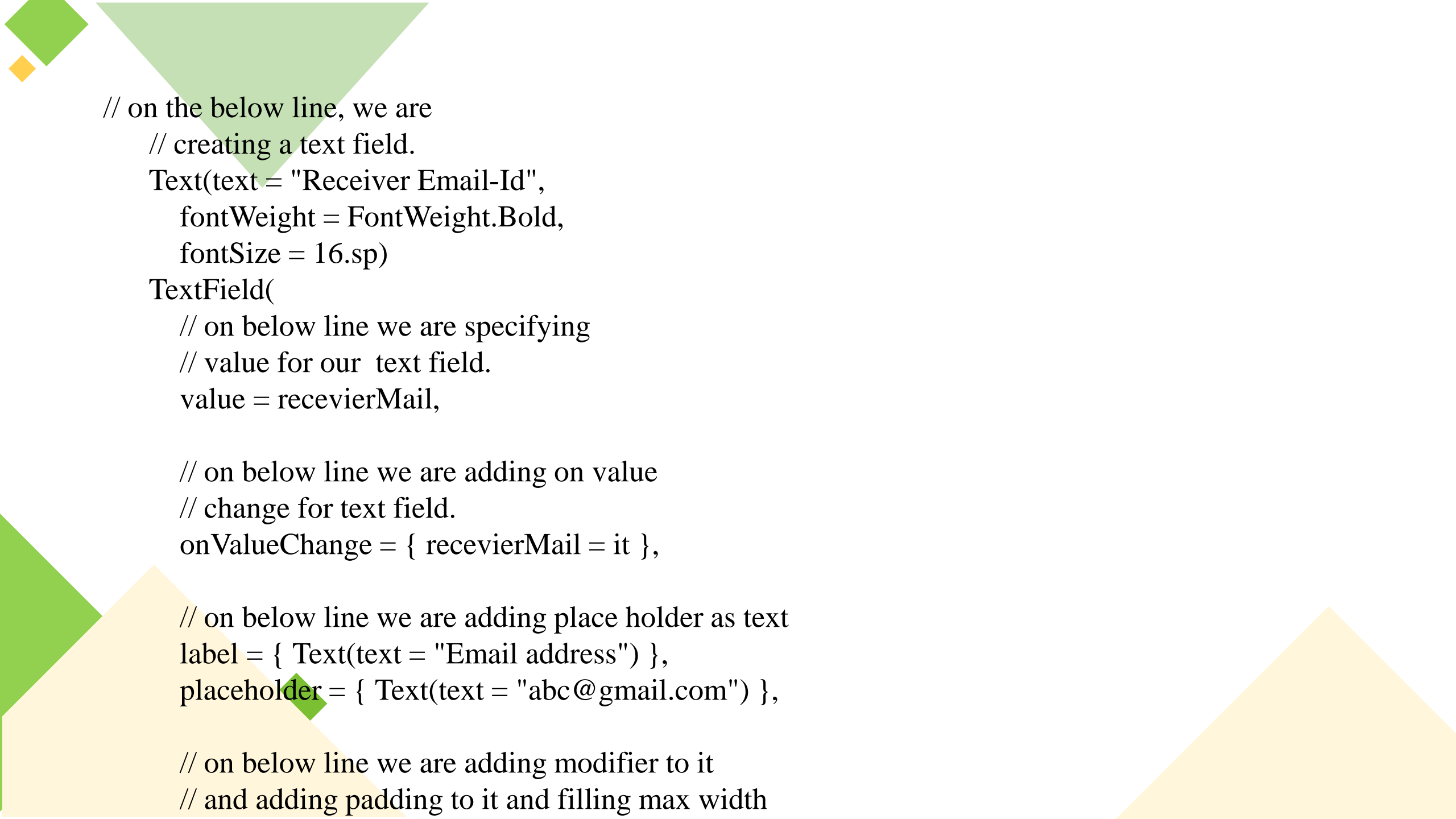
```
// creating variables for URL
```



```
var recevierMail by remember { mutableStateOf("") }
var subject by remember { mutableStateOf("") }
var body by remember { mutableStateOf("") }
var error by remember { mutableStateOf("") }
// on below line we are creating
// a variable for a context
val ctx = LocalContext.current

// on below line we are creating a column
Column(
    // on below line we are specifying modifier
    // and setting max height and max width
    // for our column
    modifier = Modifier
        .fillMaxSize()
        .padding(top = 55.dp, bottom = 25.dp, start = 25.dp, end = 25.dp),
    horizontalAlignment = Alignment.Start
) {
```





```
// on the below line, we are
// creating a text field.
Text(text = "Receiver Email-Id",
    fontWeight = FontWeight.Bold,
    fontSize = 16.sp)
TextField(
    // on below line we are specifying
    // value for our text field.
    value = recevierMail,

    // on below line we are adding on value
    // change for text field.
    onValueChange = { recevierMail = it },

    // on below line we are adding place holder as text
    label = { Text(text = "Email address") },
    placeholder = { Text(text = "abc@gmail.com") },

    // on below line we are adding modifier to it
    // and adding padding to it and filling max width
```



```
modifier = Modifier
    .padding(16.dp)
    .fillMaxWidth(),
// on below line we are adding text style
// specifying color and font size to it.
textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),

// on below line we are
// adding single line to it.
singleLine = true,
)
// on below line adding a spacer.
Spacer(modifier = Modifier.height(10.dp))

Text(text = "Mail Subject",
    fontWeight = FontWeight.Bold,
    fontSize = 16.sp)
```

// on the below line, we are creating a text field.

```
TextField(
```

```
    // on below line we are specifying
```

```
    // value for our text field.
```

```
    value = subject,
```

```
    // on below line we are adding on value change
```

```
    // for text field.
```

```
    onChange = { subject = it },
```

```
    // on below line we are adding place holder as text
```

```
    placeholder = { Text(text = "Subject") },
```

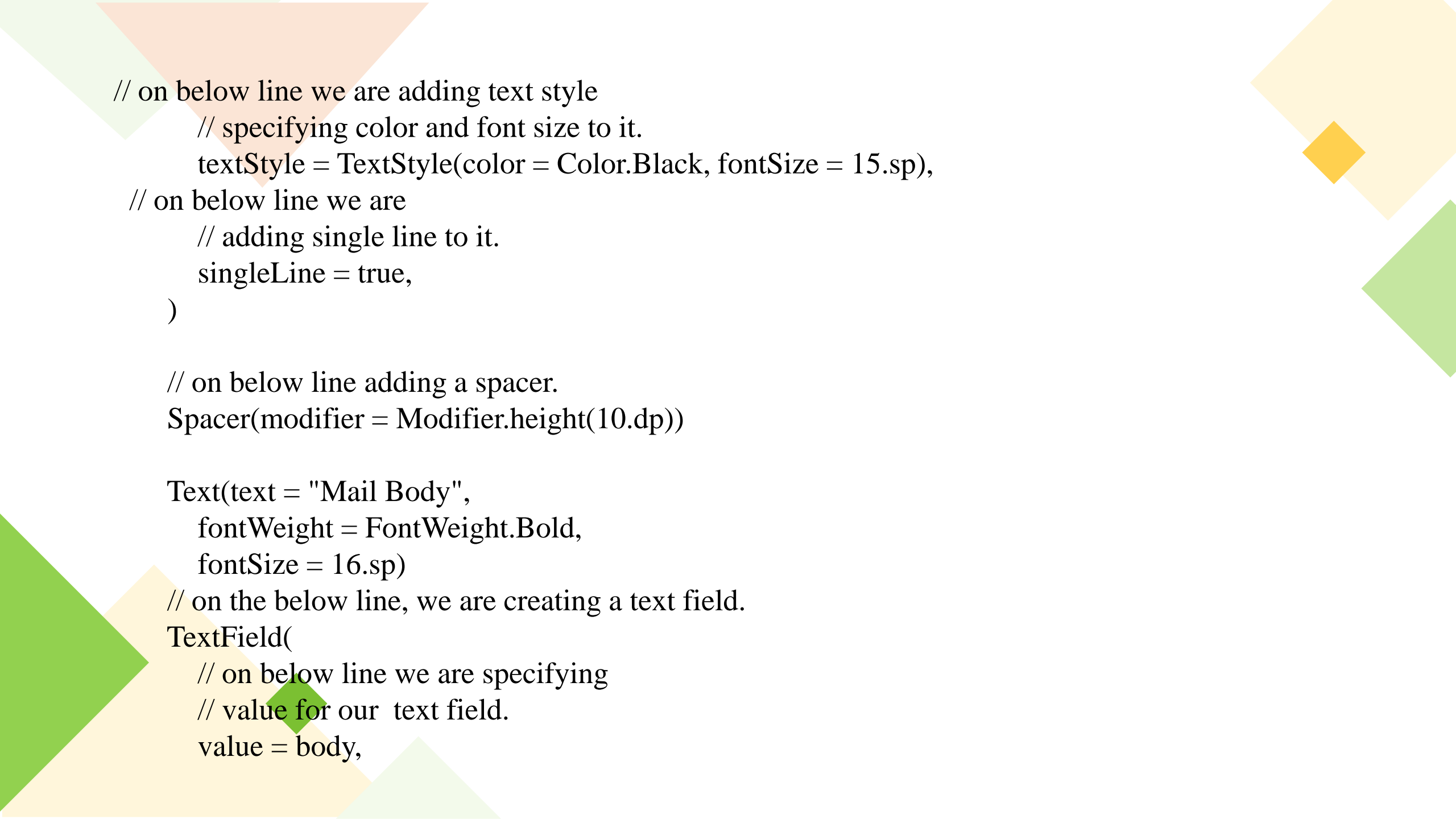
```
    // on below line we are adding modifier to it
```

```
    // and adding padding to it and filling max width
```

```
    modifier = Modifier
```

```
        .padding(16.dp)
```

```
        .fillMaxWidth(),
```



```
// on below line we are adding text style
    // specifying color and font size to it.
    textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),
// on below line we are
    // adding single line to it.
    singleLine = true,
)
```

```
// on below line adding a spacer.
Spacer(modifier = Modifier.height(10.dp))
```

```
Text(text = "Mail Body",
    fontWeight = FontWeight.Bold,
    fontSize = 16.sp)
// on the below line, we are creating a text field.
TextField(
    // on below line we are specifying
    // value for our text field.
    value = body,
```

// on below line we are adding on value

// change for text field.

onValueChange = { body = it },

// on below line we are adding place holder as text

placeholder = { Text(text = "Body") },

// on below line we are adding modifier to it

// and adding padding to it and filling max width

modifier = Modifier

.padding(16.dp)

.fillMaxWidth(),

// on below line we are adding text style

// specifying color and font size to it.

textStyle = TextStyle(color = Color.Black, fontSize = 15.sp),

// on below line we are

// adding single line to it.

singleLine = true,

)

// on below line adding a spacer.

```
Spacer(modifier = Modifier.height(20.dp))
```

// on below line adding a

// button to send an email

```
Button(onClick = {
```

```
    if( recevierMail.isNotEmpty() && subject.isNotEmpty() && body.isNotEmpty()) {
```

```
        val email = Email(
```

```
            id = null,
```

```
            recevierMail = recevierMail,
```

```
            subject = subject,
```

```
            body = body
```

```
        )
```

```
        databaseHelper.insertEmail(email)
```

```
        error = "Mail Saved"
```

```
    } else {
```

```
        error = "Please fill all fields"
```

```
    }
```

```
// on below line we are creating
// an intent to send an email
val i = Intent(Intent.ACTION_SEND)

// on below line we are passing email address,
// email subject and email body
val emailAddress = arrayOf(recevierMail)
i.putExtra(Intent.EXTRA_EMAIL,emailAddress)
i.putExtra(Intent.EXTRA_SUBJECT,subject)
i.putExtra(Intent.EXTRA_TEXT,body)

// on below line we are
// setting type of intent
i.setType("message/rfc822")
// on the below line we are starting our activity to open email application.
ctx.startActivity(Intent.createChooser(i,"Choose an Email client : "))

},
```

```
colors = ButtonDefaults.buttonColors(backgroundColor = Color(0xFFd3e5ef))
    ) {
        // on the below line creating a text for our button.
        Text(
            // on below line adding a text ,
            // padding, color and font size.
            text = "Send Email",
            modifier = Modifier.padding(10.dp),
            color = Color.Black,
            fontSize = 15.sp
        )
    }
}
```



Login

Login

[Sign up](#)

[Forget password?](#)

10:52



Register

Username

Email

Password

Register

Have an account? [Log in](#)

The background features decorative geometric shapes in the corners: a cluster of yellow, green, and orange diamonds in the top-left; a large light green triangle in the top-right; a large light green triangle in the bottom-left; and a cluster of green, yellow, and orange diamonds in the bottom-right.

THANK YOU