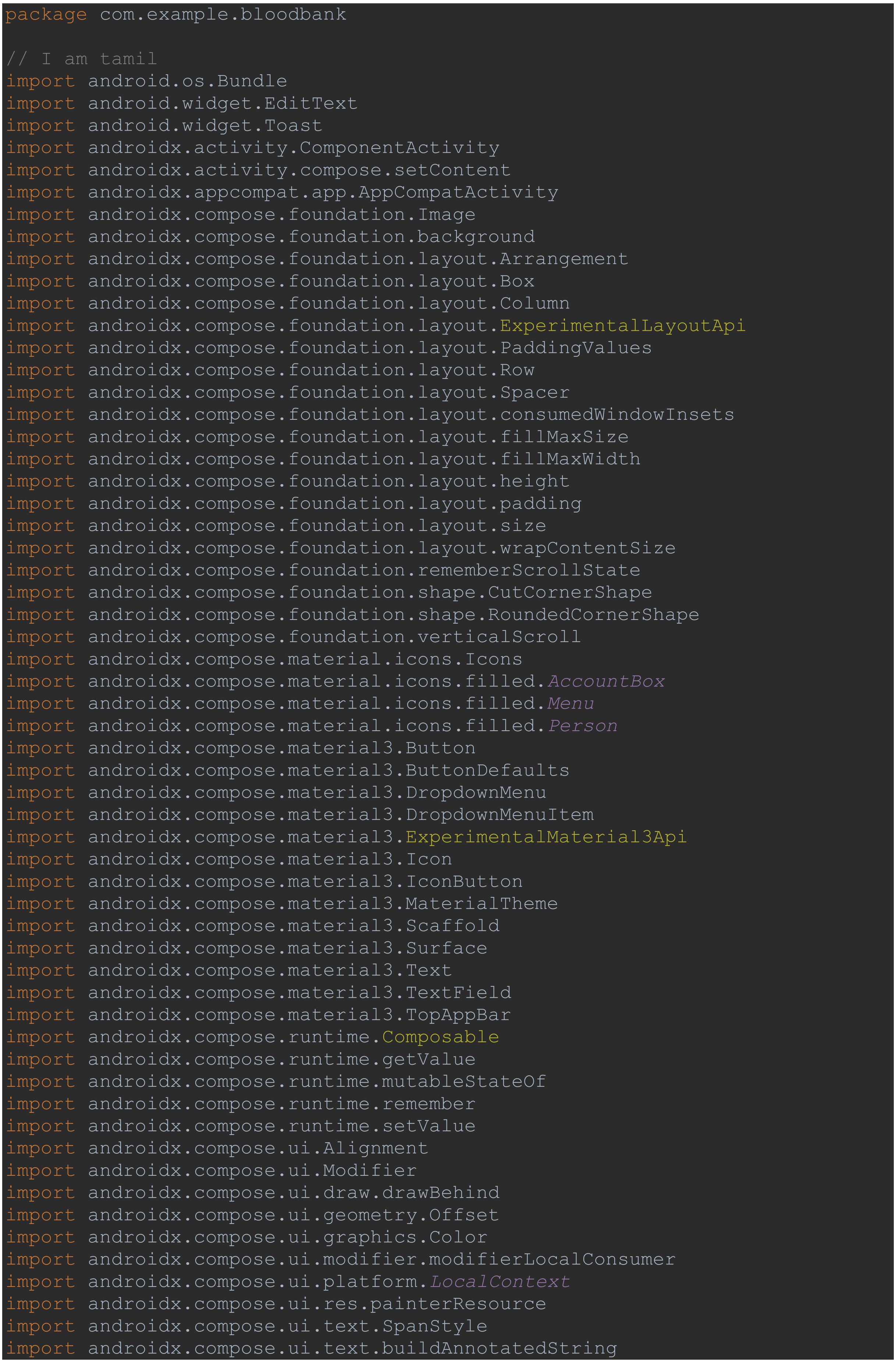
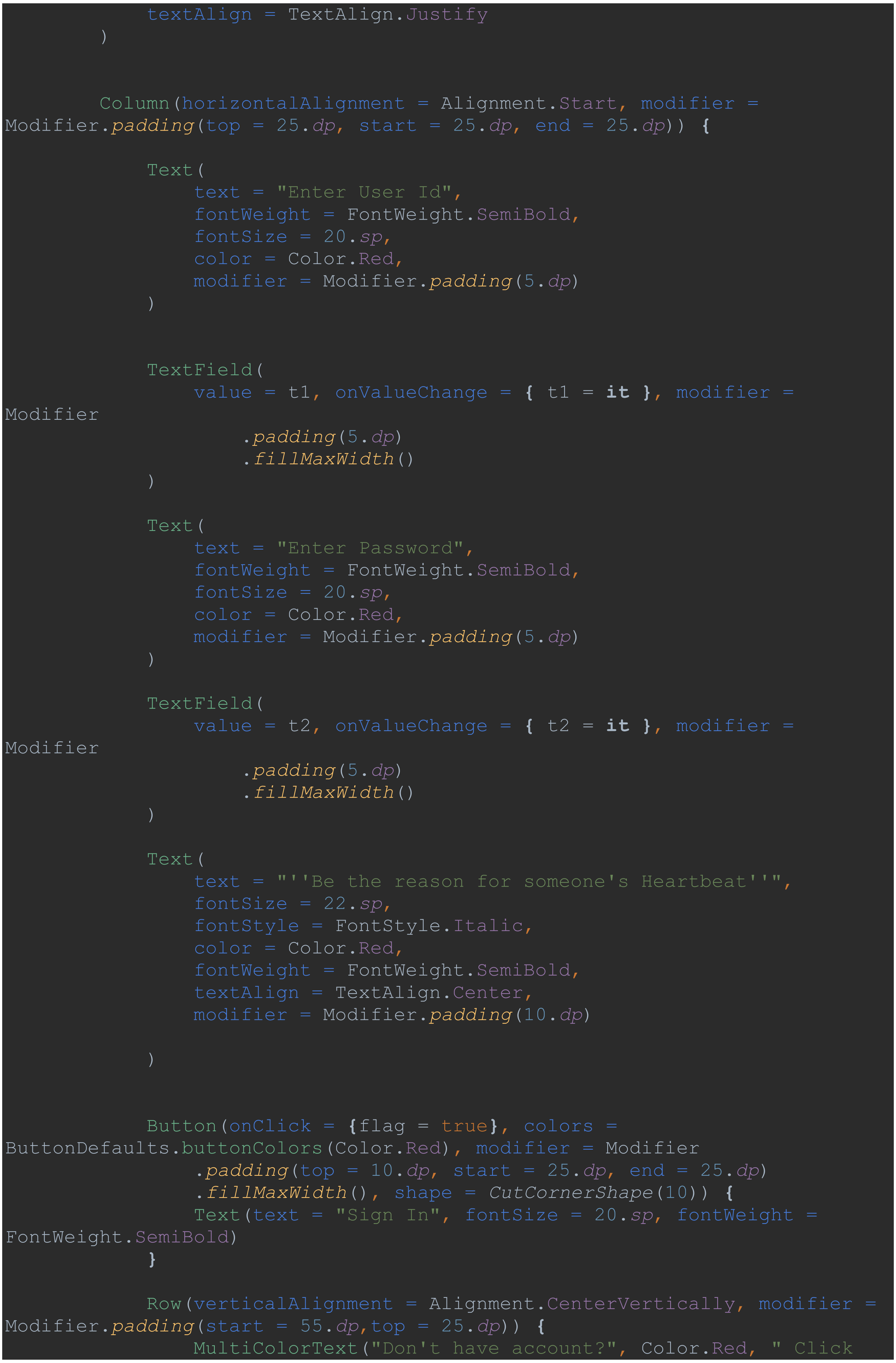
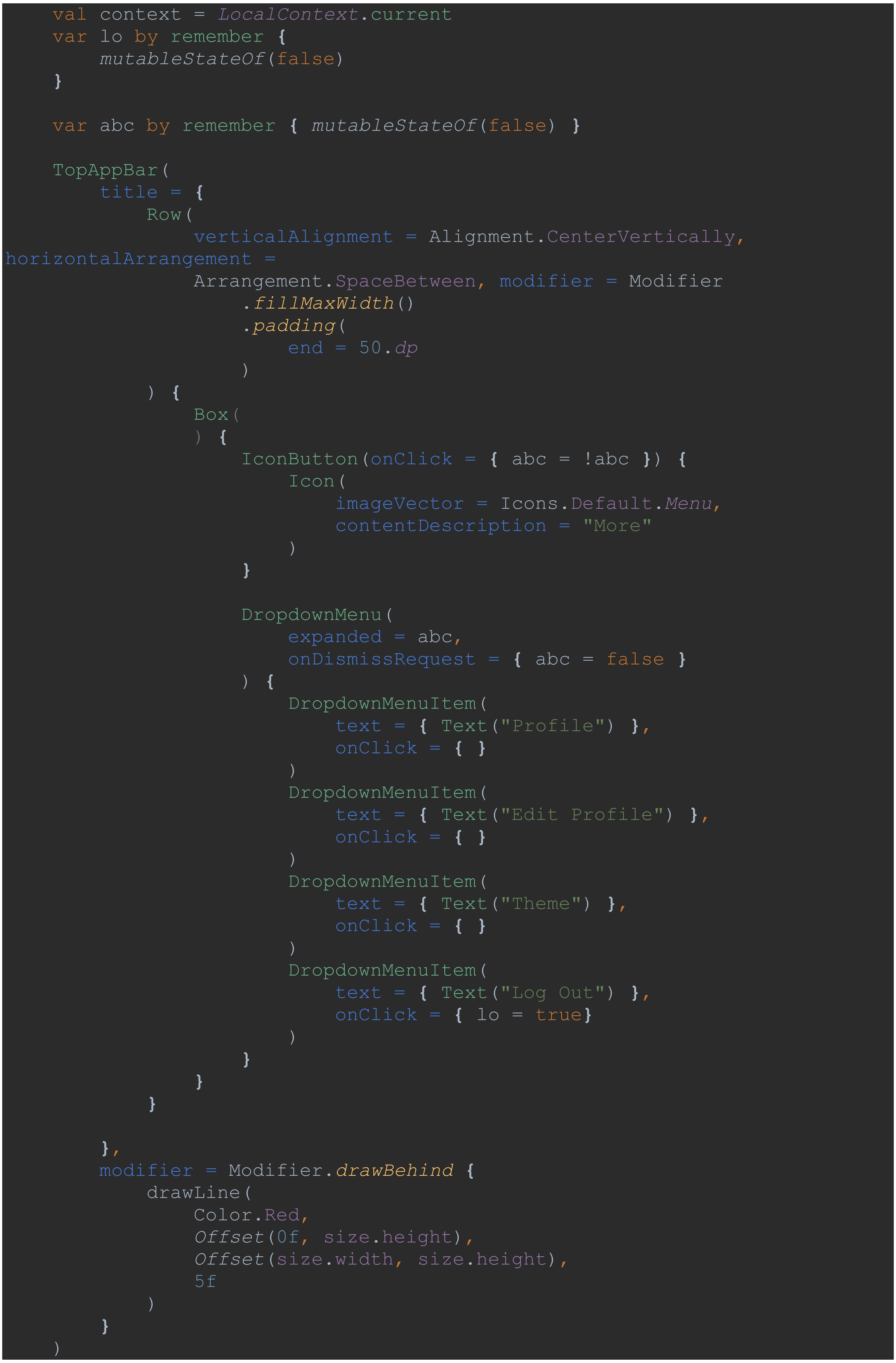


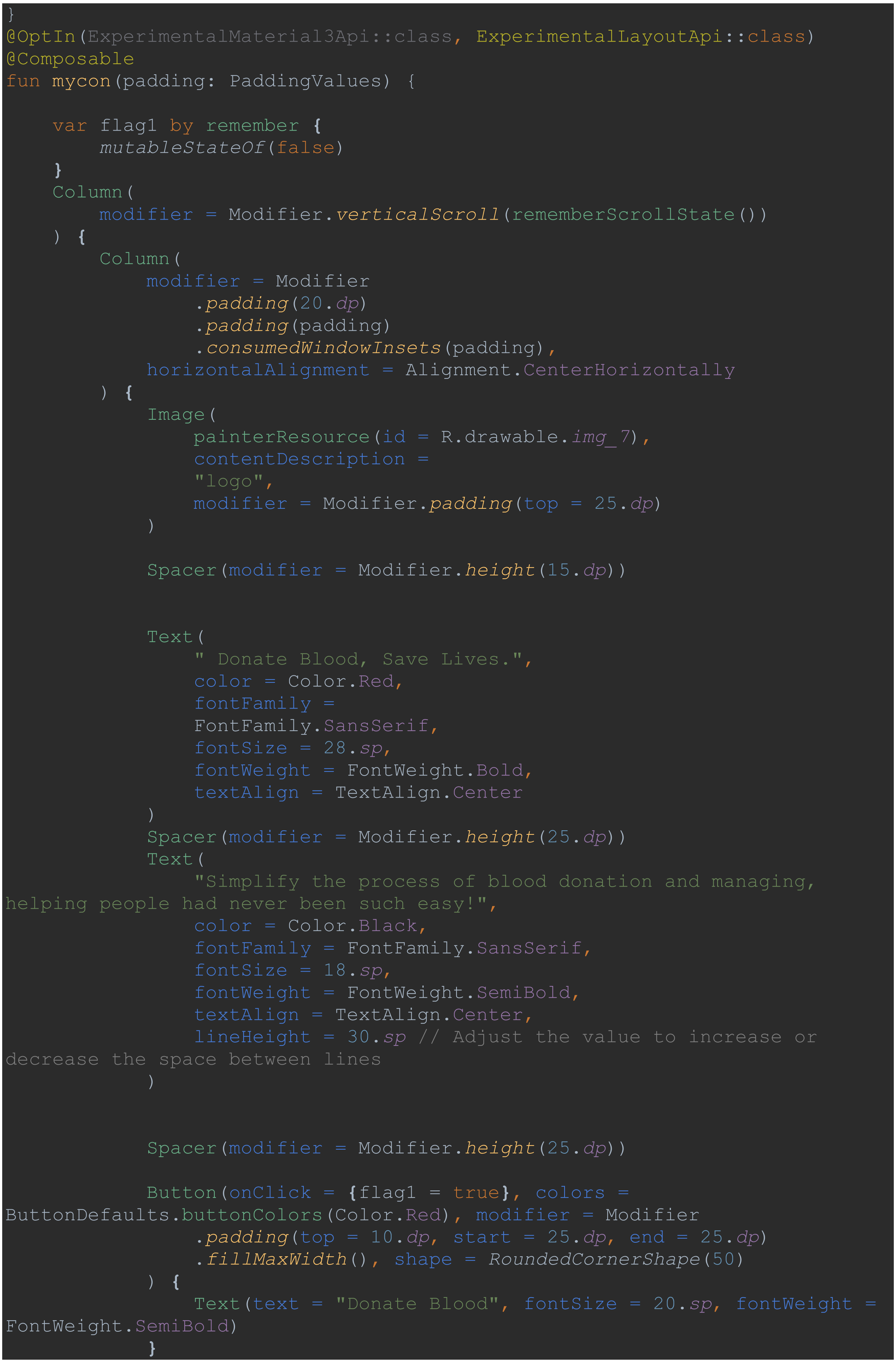
```kotlin
package com.example.bloodbank

// I am tamil
import android.os.Bundle
import android.widget.EditText
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.appcompat.app.AppCompatActivity
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.ExperimentalLayoutApi
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.consumedWindowInsets
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.wrapContentSize
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.CutCornerShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.AccountBox
import androidx.compose.material.icons.filled.Menu
import androidx.compose.material.icons.filled.Person
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.DropdownMenu
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TopAppBar
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.draw.drawBehind
import androidx.compose.ui.geometry.Offset
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.modifier.modifierLocalConsumer
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.SpanStyle
import androidx.compose.ui.text.buildAnnotatedString
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.text.style.TextDecoration
import androidx.compose.ui.text.withStyle
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.bloodbank.ui.theme.BloodBankTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            BloodBankTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    Bloodbank()
                }
            }
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Bloodbank() {

    var flag by remember {
        mutableStateOf(false)
    }

    var t1 by remember {
        mutableStateOf("")
    }
    var t2 by remember {
        mutableStateOf("")
    }

    Column(
        verticalArrangement = Arrangement.Top,
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = Modifier
            .fillMaxSize()
    ) {
        Image(
            painterResource(id = R.drawable.img_3),
            contentDescription = "",
            modifier = Modifier.padding(20.dp)
        )

        Text(
            text = "Welcome Donar !",
            fontWeight = FontWeight.Bold,
            fontSize = 35.sp,
            color = Color.Red,
            modifier = Modifier.padding(20.dp),
```



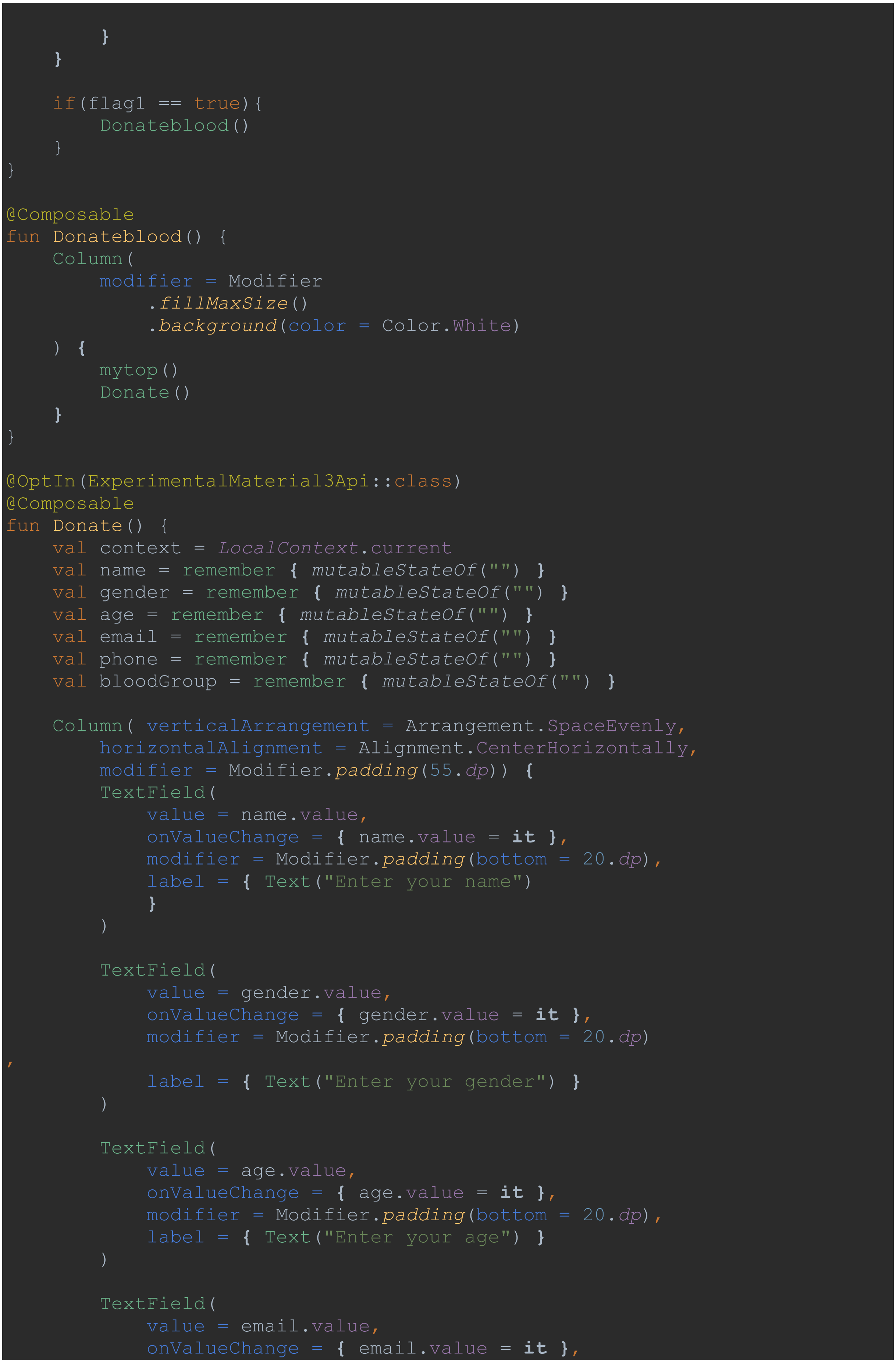
```
            textAlign = TextAlign.Justify
        )


        Column(horizontalAlignment = Alignment.Start, modifier =
Modifier.padding(top = 25.dp, start = 25.dp, end = 25.dp)) {

            Text(
                text = "Enter User Id",
                fontWeight = FontWeight.SemiBold,
                fontSize = 20.sp,
                color = Color.Red,
                modifier = Modifier.padding(5.dp)
            )


            TextField(
                value = t1, onValueChange = { t1 = it }, modifier =
Modifier
                    .padding(5.dp)
                    .fillMaxWidth()
            )

            Text(
                text = "Enter Password",
                fontWeight = FontWeight.SemiBold,
                fontSize = 20.sp,
                color = Color.Red,
                modifier = Modifier.padding(5.dp)
            )

            TextField(
                value = t2, onValueChange = { t2 = it }, modifier =
Modifier
                    .padding(5.dp)
                    .fillMaxWidth()
            )

            Text(
                text = "''Be the reason for someone's Heartbeat''",
                fontSize = 22.sp,
                fontStyle = FontStyle.Italic,
                color = Color.Red,
                fontWeight = FontWeight.SemiBold,
                textAlign = TextAlign.Center,
                modifier = Modifier.padding(10.dp)

            )


            Button(onClick = {flag = true}, colors =
ButtonDefaults.buttonColors(Color.Red), modifier = Modifier
                .padding(top = 10.dp, start = 25.dp, end = 25.dp)
                .fillMaxWidth(), shape = CutCornerShape(10)) {
                Text(text = "Sign In", fontSize = 20.sp, fontWeight =
FontWeight.SemiBold)
            }

            Row(verticalAlignment = Alignment.CenterVertically, modifier =
Modifier.padding(start = 55.dp,top = 25.dp)) {
                MultiColorText("Don't have account?", Color.Red, " Click
Here.", Color.Red)
                }



            }

        }

        if(flag == true){
            Second(t1 = t1, t2 =t2)
        }


}


@Composable
fun Second(t1:String, t2:String){

    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(color = Color(240, 161, 170))
    ) {
        Scaff()
    }

}


@Composable
fun MultiColorText(text1: String, color1: Color, text2: String, color2:
Color) {
    Text(buildAnnotatedString {
        withStyle(style = SpanStyle(color = color1)) {
            append(text1)
        }
        withStyle(style = SpanStyle(color = color2,textDecoration =
TextDecoration.Underline)) {
            append(text2)
        }
    })
}


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Scaff() {
    Scaffold(
        topBar = {mytop()},
        bottomBar = {},
        content = {padding -> mycon(padding)}
    )
}


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun mytop() {
```



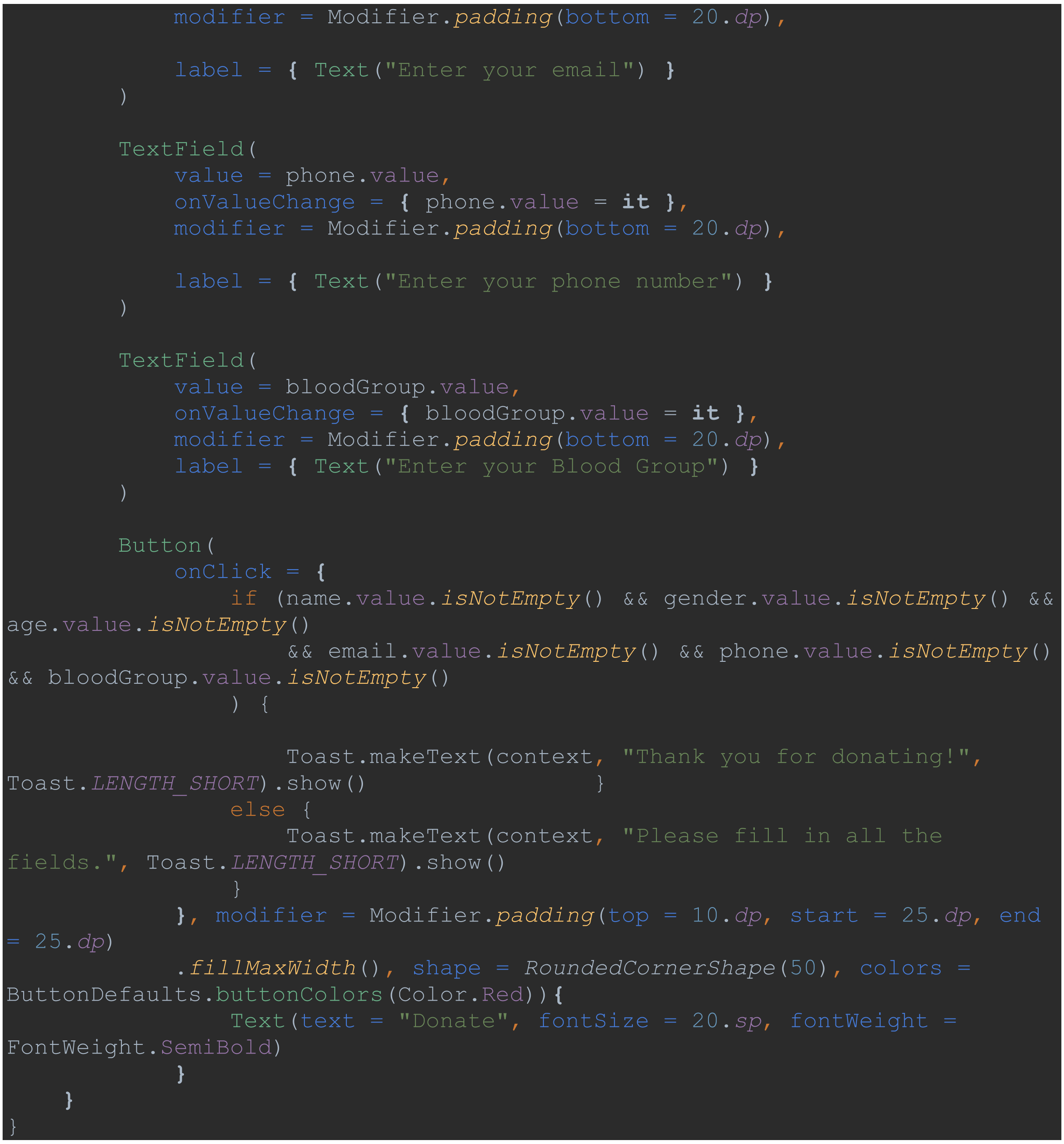
```
    val context = LocalContext.current
    var lo by remember {
        mutableStateOf(false)
    }

    var abc by remember { mutableStateOf(false) }

    TopAppBar(
        title = {
            Row(
                verticalAlignment = Alignment.CenterVertically,
horizontalArrangement =
                Arrangement.SpaceBetween, modifier = Modifier
                    .fillMaxWidth()
                    .padding(
                        end = 50.dp
                    )
            ) {
                Box(
                ) {
                    IconButton(onClick = { abc = !abc }) {
                        Icon(
                            imageVector = Icons.Default.Menu,
                            contentDescription = "More"
                        )
                    }

                    DropdownMenu(
                        expanded = abc,
                        onDismissRequest = { abc = false }
                    ) {
                        DropdownMenuItem(
                            text = { Text("Profile") },
                            onClick = { }
                        )
                        DropdownMenuItem(
                            text = { Text("Edit Profile") },
                            onClick = { }
                        )
                        DropdownMenuItem(
                            text = { Text("Theme") },
                            onClick = { }
                        )
                        DropdownMenuItem(
                            text = { Text("Log Out") },
                            onClick = { lo = true}
                        )
                    }
                }
            }

        },
        modifier = Modifier.drawBehind {
            drawLine(
                Color.Red,
                Offset(0f, size.height),
                Offset(size.width, size.height),
                5f
            )
        }
    )
```



```
}
@OptIn(ExperimentalMaterial3Api::class, ExperimentalLayoutApi::class)
@Composable
fun mycon(padding: PaddingValues) {

    var flag1 by remember {
        mutableStateOf(false)
    }
    Column(
        modifier = Modifier.verticalScroll(rememberScrollState())
    ) {
        Column(
            modifier = Modifier
                .padding(20.dp)
                .padding(padding)
                .consumedWindowInsets(padding),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Image(
                painterResource(id = R.drawable.img_7),
                contentDescription =
                "logo",
                modifier = Modifier.padding(top = 25.dp)
            )

            Spacer(modifier = Modifier.height(15.dp))


            Text(
                " Donate Blood, Save Lives.",
                color = Color.Red,
                fontFamily =
                FontFamily.SansSerif,
                fontSize = 28.sp,
                fontWeight = FontWeight.Bold,
                textAlign = TextAlign.Center
            )
            Spacer(modifier = Modifier.height(25.dp))
            Text(
                "Simplify the process of blood donation and managing,
helping people had never been such easy!",
                color = Color.Black,
                fontFamily = FontFamily.SansSerif,
                fontSize = 18.sp,
                fontWeight = FontWeight.SemiBold,
                textAlign = TextAlign.Center,
                lineHeight = 30.sp // Adjust the value to increase or
decrease the space between lines
            )


            Spacer(modifier = Modifier.height(25.dp))

            Button(onClick = {flag1 = true}, colors =
ButtonDefaults.buttonColors(Color.Red), modifier = Modifier
                .padding(top = 10.dp, start = 25.dp, end = 25.dp)
                .fillMaxWidth(), shape = RoundedCornerShape(50)
            ) {
                Text(text = "Donate Blood", fontSize = 20.sp, fontWeight =
FontWeight.SemiBold)
            }
```



```
        }
    }

    if(flag1 == true){
        Donateblood()
    }
}

@Composable
fun Donateblood() {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .background(color = Color.White)
    ) {
        mytop()
        Donate()
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun Donate() {
    val context = LocalContext.current
    val name = remember { mutableStateOf("") }
    val gender = remember { mutableStateOf("") }
    val age = remember { mutableStateOf("") }
    val email = remember { mutableStateOf("") }
    val phone = remember { mutableStateOf("") }
    val bloodGroup = remember { mutableStateOf("") }

    Column( verticalArrangement = Arrangement.SpaceEvenly,
        horizontalAlignment = Alignment.CenterHorizontally,
        modifier = Modifier.padding(55.dp)) {
        TextField(
            value = name.value,
            onValueChange = { name.value = it },
            modifier = Modifier.padding(bottom = 20.dp),
            label = { Text("Enter your name")
            }
        )

        TextField(
            value = gender.value,
            onValueChange = { gender.value = it },
            modifier = Modifier.padding(bottom = 20.dp)
,
            label = { Text("Enter your gender") }
        )

        TextField(
            value = age.value,
            onValueChange = { age.value = it },
            modifier = Modifier.padding(bottom = 20.dp),
            label = { Text("Enter your age") }
        )

        TextField(
            value = email.value,
            onValueChange = { email.value = it },
```



```
            modifier = Modifier.padding(bottom = 20.dp),

            label = { Text("Enter your email") }
        )

        TextField(
            value = phone.value,
            onValueChange = { phone.value = it },
            modifier = Modifier.padding(bottom = 20.dp),

            label = { Text("Enter your phone number") }
        )

        TextField(
            value = bloodGroup.value,
            onValueChange = { bloodGroup.value = it },
            modifier = Modifier.padding(bottom = 20.dp),
            label = { Text("Enter your Blood Group") }
        )

        Button(
            onClick = {
                if (name.value.isNotEmpty() && gender.value.isNotEmpty() &&
age.value.isNotEmpty()
                    && email.value.isNotEmpty() && phone.value.isNotEmpty()
&& bloodGroup.value.isNotEmpty()
                ) {

                    Toast.makeText(context, "Thank you for donating!",
Toast.LENGTH_SHORT).show()                }
                else {
                    Toast.makeText(context, "Please fill in all the
fields.", Toast.LENGTH_SHORT).show()
                }
            }, modifier = Modifier.padding(top = 10.dp, start = 25.dp, end
= 25.dp)
            .fillMaxWidth(), shape = RoundedCornerShape(50), colors =
ButtonDefaults.buttonColors(Color.Red)){
                Text(text = "Donate", fontSize = 20.sp, fontWeight =
FontWeight.SemiBold)
            }
    }
}
```