Assignment – 2

Name: MD SHABREZ

Register Number: 20BCE1690

Blood Bank App

MainActivity.kt

```
package com.zach.vit_20bce1690_bloodbankapp

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import
com.zach.vit_20bce1690_bloodbankapp.ui.theme.VIT_20BCE1690_BloodBankAppThe
me

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            VIT_20BCE1690_BloodBankAppTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    ActivityNavigation()
```

```
                }
            }
        }
    }
}
```

ActivityNavigation.kt

```
package com.zach.vit_20bce1690_bloodbankapp

import android.app.DatePickerDialog
import android.widget.DatePicker
import androidx.compose.foundation.BorderStroke
import androidx.compose.foundation.Image
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
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.width
import androidx.compose.foundation.rememberScrollState
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.Check
import androidx.compose.material.icons.filled.KeyboardArrowDown
import androidx.compose.material.icons.filled.KeyboardArrowUp
import androidx.compose.material.icons.filled.Send
import androidx.compose.material.icons.outlined.AccountBox
import androidx.compose.material.icons.outlined.Add
```

```
import androidx.compose.material.icons.outlined.DateRange
import androidx.compose.material.icons.outlined.Email
import androidx.compose.material.icons.outlined.Place
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.Divider
import androidx.compose.material3.DropdownMenu
import androidx.compose.material3.DropdownMenuItem
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.material3.TopAppBar
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
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.geometry.Size
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.onGloballyPositioned
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalDensity
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.input.TextFieldValue
import androidx.compose.ui.text.input.VisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
```

```
import androidx.compose.ui.unit.toSize
import androidx.navigation.NavController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.navArgument
import androidx.navigation.compose.rememberNavController
import java.util.Calendar
import java.util.Date

@Composable
fun ActivityNavigation() {
    val navController = rememberNavController()
    NavHost(navController = navController, startDestination =
Activities.Home.route) {

        composable(Activities.Home.route) {
            Home(navController = navController)
        }

        composable(
            route = Activities.Destination.route +

"/{name}/{email}/{address}/{mobile}/{donationDate}/{appointmentDate}/{bloo
dGroup}",
            arguments = listOf(
                navArgument(name = "name") {
                    type = NavType.StringType
                },
                navArgument(name = "email") {
                    type = NavType.StringType
                },
                navArgument(name = "address") {
                    type = NavType.StringType
                },
                navArgument(name = "mobile") {
                    type = NavType.StringType
                },
                navArgument(name = "donationDate") {
                    type = NavType.StringType
                },
                navArgument(name = "appointmentDate") {
```

```
                    type = NavType.StringType
                },
                navArgument(name = "bloodGroup") {
                    type = NavType.StringType
                },
            )
        ) { entry ->
            Box(
                modifier = Modifier
                    .fillMaxSize()
                    .padding(40.dp, 20.dp)
            ) {
                DonationSchedule(
                    name = entry.arguments?.getString("name"),
                    email = entry.arguments?.getString("email"),
                    address = entry.arguments?.getString("address"),
                    mobile = entry.arguments?.getString("mobile"),
                    appointmentDate =
entry.arguments?.getString("appointmentDate"),
                    bloodGroup = entry.arguments?.getString("bloodGroup")
                )
            }


        }
    }
}

@Composable
fun Home(navController: NavController) {
    CustomScaffold(navController)
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CustomScaffold(navController: NavController) {
    LocalContext.current
    Scaffold(topBar = { CustomTopBar() },
        content = { pad -> MainContent(pad, navController) }
    )
}
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CustomTopBar() {
    TopAppBar(
        title = {
            Row(
                verticalAlignment = Alignment.CenterVertically,
horizontalArrangement =
                Arrangement.SpaceBetween, modifier = Modifier
                    .fillMaxWidth()
                    .padding(
                        end = 50
                            .dp
                    )
            ) {
                Icon(
                    Icons.Filled.AccountBox, contentDescription = "",
modifier = Modifier
                        .size(30.dp)
                )
                Image(
                    painterResource(id = R.drawable.banklogo1),
                    contentDescription = "",
                    modifier = Modifier.size(50.dp)
                )
                Spacer(modifier = Modifier)

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

```
@OptIn(ExperimentalLayoutApi::class, ExperimentalMaterial3Api::class)
@Composable
fun MainContent(padding: PaddingValues, navController: NavController) {
    val primaryTextColor = remember {
        mutableStateOf(Color(115, 115, 115))
    }

    val tertiaryTextColor = remember {
        mutableStateOf(Color.Black)
    }

    val textFieldColor = remember {
        mutableStateOf(Color(250, 250, 250))
    }

    val mobile = remember {
        mutableStateOf(TextFieldValue())
    }

    val fullName = remember {
        mutableStateOf(TextFieldValue())
    }

    val email = remember {
        mutableStateOf(TextFieldValue())
    }

    val address = remember {
        mutableStateOf(TextFieldValue())
    }

    val date = remember { mutableStateOf("") }

    val date2 = remember { mutableStateOf("") }


    var mSelectedText by remember { mutableStateOf("") }

    Column(
        modifier = Modifier.verticalScroll(rememberScrollState())
    ) {
        Column(
```

```
            modifier = Modifier
                .padding(20.dp)
                .padding(padding)
                .consumedWindowInsets(padding),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            Image(
                painterResource(id = R.drawable.banklogo),
                contentDescription =
                "blood bank logo",
                modifier = Modifier.size(width = 300.dp, height = 200.dp)
            )
            Spacer(modifier = Modifier.height(30.dp))
            Text(
                "Give Life, Give Blood: Be a Hero, Get Appointment and
Donate Today!",
                color = Color(202, 5, 37, 255),
                fontFamily =
                FontFamily.SansSerif,
                fontSize = 20.sp,
                fontWeight = FontWeight.SemiBold,
                textAlign = TextAlign.Center
            )

            Spacer(modifier = Modifier.height(40.dp))
            CustomTextField(
                modifier = Modifier.fillMaxWidth(),
                mutableValue =
                fullName,
                label = "Full Name",
                focusedColor = primaryTextColor.value,
                textColor = tertiaryTextColor.value,
                conColor = textFieldColor.value

            )
            Spacer(modifier = Modifier.height(10.dp))

            CustomTextField(
                modifier = Modifier.fillMaxWidth(),
                mutableValue =
                email,
                label = "Email",
```

```
            focusedColor = primaryTextColor.value,
            textColor = tertiaryTextColor.value,
            conColor = textFieldColor.value
        )
        Spacer(modifier = Modifier.height(10.dp))

        CustomTextField(
            modifier = Modifier
                .fillMaxWidth()
                .height(100.dp),
            mutableValue =
            address,
            label = "Address",
            focusedColor = primaryTextColor.value,
            textColor = tertiaryTextColor.value,
            conColor = textFieldColor.value
        )
        Spacer(modifier = Modifier.height(10.dp))

        CustomTextField(
            modifier = Modifier.fillMaxWidth(),
            mutableValue =
            mobile,
            label = "Mobile Number",
            focusedColor = primaryTextColor.value,
            textColor = tertiaryTextColor.value,
            conColor = textFieldColor.value
        )

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

        val year: Int
        val month: Int
        val day: Int

        val calendar = Calendar.getInstance()
        year = calendar.get(Calendar.YEAR)
        month = calendar.get(Calendar.MONTH)
        day = calendar.get(Calendar.DAY_OF_MONTH)
        calendar.time = Date()

        val datePickerDialog = DatePickerDialog(
```

```
                LocalContext.current,
                { _: DatePicker, year: Int, month: Int, dayOfMonth: Int ->
                    date.value = "$dayOfMonth/$month/$year"
                }, year, month, day
            )

            val datePickerDialog2 = DatePickerDialog(
                LocalContext.current,
                { _: DatePicker, year: Int, month: Int, dayOfMonth: Int ->
                    date2.value = "$dayOfMonth/$month/$year"
                }, year, month, day
            )


            CustomButton(buttonText = "Select Last Donation Date", onClick
= {
                datePickerDialog.show()
            })
            Spacer(modifier = Modifier.height(5.dp))
            Text(
                "Last Date of Donation: ${date.value}",
                color = Color(202, 5, 37, 255),
                fontFamily =
                FontFamily.SansSerif,
                fontSize = 16.sp,
                textAlign = TextAlign.Start,
                modifier = Modifier.align(Alignment.Start)
            )

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

            CustomButton(buttonText = "Select Preferred Date of
Appointment", onClick = {
                datePickerDialog2.show()
            })
            Spacer(modifier = Modifier.height(5.dp))
            Text(
                "Preferred Date of Appointment: ${date2.value}",
                color = Color(202, 5, 37, 255),
                fontFamily =
                FontFamily.SansSerif,
                fontSize = 16.sp,
```

```
                textAlign = TextAlign.Start,
                modifier = Modifier.align(Alignment.Start)
            )



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

            var mExpanded by remember { mutableStateOf(false) }

            val bloodGroups = listOf("A+", "B+", "AB+", "O+", "A-", "B-",
"AB-", "O-")

            var mTextFieldSize by remember { mutableStateOf(Size.Zero) }

            val icon = if (mExpanded)
                Icons.Filled.KeyboardArrowUp
            else
                Icons.Filled.KeyboardArrowDown

            Column {
                OutlinedTextField(
                    value = mSelectedText,
                    onValueChange = { mSelectedText = it },
                    modifier = Modifier
                        .fillMaxWidth()
                        .onGloballyPositioned { coordinates ->
                            mTextFieldSize = coordinates.size.toSize()
                        },
                    label = { Text("Blood Group") },
                    trailingIcon = {
                        Icon(icon, "contentDescription",
                            Modifier.clickable { mExpanded = !mExpanded })
                    }
                )

                DropdownMenu(
                    expanded = mExpanded,
                    onDismissRequest = { mExpanded = false },
                    modifier = Modifier
                        .width(with(LocalDensity.current) {
mTextFieldSize.width.toDp() })
```

```
                ) {
                    bloodGroups.forEach { label ->
                        DropdownMenuItem(text = { Text(text = label) },
onClick = {
                            mSelectedText = label
                            mExpanded = false
                        })
                    }
                }
            }



            Spacer(modifier = Modifier.height(40.dp))

            CustomButton(
                buttonText = "Send", onClick = {

                    navController.navigate(
                        Activities.Destination.withArgs(
                            fullName.value.text,
                            email.value.text,
                            address.value.text,
                            mobile.value.text,
                            date.value.replace('/', '-'),
                            date2.value.replace('/', '-'),
                            mSelectedText
                        )
                    )

                }, isLogo =
                true
            )

            Spacer(modifier = Modifier.height(40.dp))

        }
    }
}


@Composable
```

```
fun DonationSchedule(
    name: String?, email: String?, address: String?, mobile: String?,
    appointmentDate: String?, bloodGroup: String?
) {

    Box(contentAlignment = Alignment.Center) {
        Card(
            colors = CardDefaults.cardColors(
                containerColor = Color.White
            ),
            elevation = CardDefaults.cardElevation(
                defaultElevation = 10.dp
            ),
            shape = RoundedCornerShape(1.dp),
        ) {
            Column(Modifier.padding(20.dp)) {
                CustomButtonCheck(
                    buttonText = "Confirmed", isLogo =
                    true
                )
                Spacer(modifier = Modifier.height(15.dp))
                Divider(color = Color.LightGray, thickness = 1.dp)
                Spacer(modifier = Modifier.height(15.dp))
                Row {
                    Icon(
                        Icons.Outlined.AccountBox,
                        contentDescription = "",
                        modifier = Modifier
                            .size(30.dp)
                    )
                    Spacer(modifier = Modifier.width(20.dp))
                    Column {
                        Text(
                            text = name.toString(),
                            fontWeight = FontWeight.Bold,
                            fontSize = 24.sp
                        )
                        Text(
                            text = mobile.toString(), fontSize = 16
                                .sp, color = Color(95, 95, 95, 255)
                        )
                        Spacer(modifier = Modifier.height(15.dp))
```

```
                    }

                }
                Divider(color = Color.LightGray, thickness = 1.dp)
                Spacer(modifier = Modifier.height(15.dp))

                Row {
                    Icon(
                        Icons.Outlined.DateRange,
                        contentDescription = "",
                        modifier = Modifier
                            .size(30.dp)
                    )
                    Spacer(modifier = Modifier.width(20.dp))
                    Column {
                        Text(
                            text = "Appointment Date", fontWeight =
FontWeight.Bold,
                            fontSize = 19
                                .sp
                        )
                        Text(
                            text = appointmentDate.toString(), fontSize =
16
                                .sp, color = Color(95, 95, 95, 255)
                        )
                        Spacer(modifier = Modifier.height(15.dp))
                    }

                }
                Row {
                    Icon(
                        Icons.Outlined.Email, contentDescription = "",
modifier = Modifier
                            .size(30.dp)
                    )
                    Spacer(modifier = Modifier.width(20.dp))
                    Column {
                        Text(text = "Email", fontWeight = FontWeight.Bold,
fontSize = 19.sp)
                        Text(
                            text = email.toString(), fontSize = 16
```

```
                                .sp, color = Color(95, 95, 95, 255)
                        )
                        Spacer(modifier = Modifier.height(15.dp))
                    }

                }
                Row {
                    Icon(
                        Icons.Outlined.Place, contentDescription = "",
modifier = Modifier
                            .size(30.dp)
                    )
                    Spacer(modifier = Modifier.width(20.dp))
                    Column {
                        Text(
                            text = "Location", fontWeight =
FontWeight.Bold, fontSize = 19
                                .sp
                        )
                        Text(
                            text = address.toString(), fontSize = 16
                                .sp, color = Color(95, 95, 95, 255)
                        )
                        Spacer(modifier = Modifier.height(15.dp))
                    }

                }
                Row {
                    Icon(
                        Icons.Outlined.Add, contentDescription = "",
modifier = Modifier
                            .size(30.dp)
                    )
                    Spacer(modifier = Modifier.width(20.dp))
                    Column {
                        Text(
                            text = "Blood Group", fontWeight =
FontWeight.Bold, fontSize
                            = 19.sp
                        )
                        Text(
                            text = bloodGroup.toString(), fontSize = 16
```

```
                                .sp, color = Color(95, 95, 95, 255)
                        )
                        Spacer(modifier = Modifier.height(15.dp))
                    }

                }
            }
        }
    }
}


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CustomTextField(
    modifier: Modifier = Modifier,
    mutableValue: MutableState<TextFieldValue>, label: String,
    placeholder: String
    = label,
    focusedColor: Color, conColor: Color = Color(250, 250, 250),
    isHideVal: Boolean = false, textColor: Color
) {
    TextField(
        modifier = modifier.border(
            BorderStroke(0.2.dp, focusedColor), RoundedCornerShape
                (4.dp)
        ),
        value = mutableValue.value,
        onValueChange = { mutableValue.value = it },
        label = { Text(text = label) },
        placeholder = { Text(text = placeholder) },
        colors = TextFieldDefaults.outlinedTextFieldColors(
            focusedBorderColor = Color.Transparent,
            focusedLabelColor = focusedColor,
            placeholderColor = focusedColor,
            textColor = textColor,
            unfocusedBorderColor = Color.Transparent,
            unfocusedLabelColor = focusedColor,
            unfocusedLeadingIconColor = focusedColor,
            focusedLeadingIconColor = focusedColor,
            containerColor = conColor,
```

```
            ),

        visualTransformation = if (isHideVal)
PasswordVisualTransformation(
            mask =
            '\u2022'
        ) else VisualTransformation.None,

        )
}

@Composable
fun CustomButton(
    buttonText: String,
    textColor: Color = Color.White,
    backgroundColor: Color = Color(237, 38, 71),
    onClick: () -> Unit = {},
    isLogo: Boolean = false
) {
    Button(
        onClick = onClick,
        shape = RoundedCornerShape(10.dp),
        colors = ButtonDefaults.buttonColors(backgroundColor),
        modifier = Modifier.fillMaxWidth()
    ) {
        if (isLogo) {
            Icon(
                Icons.Filled.Send, contentDescription =
                "facebook logo", modifier = Modifier.size(25.dp)
            )
        }
        Spacer(modifier = Modifier.width(10.dp))
        Text(
            buttonText,
            color = textColor,
            fontSize = 16.sp,
            fontWeight = FontWeight.Bold
        )
    }
}

@Composable
```

```
fun CustomButtonCheck(
    buttonText: String,
    textColor: Color = Color.White,
    backgroundColor: Color = Color(73, 211, 162, 255),
    onClick: () -> Unit = {},
    isLogo: Boolean = false
) {
    Button(
        onClick = onClick,
        shape = RoundedCornerShape(10.dp),
        colors = ButtonDefaults.buttonColors(backgroundColor),
        modifier = Modifier
    ) {
        if (isLogo) {
            Icon(
                Icons.Filled.Check, contentDescription =
                "facebook logo", modifier = Modifier.size(25.dp)
            )
        }
        Spacer(modifier = Modifier.width(10.dp))
        Text(
            buttonText,
            color = textColor,
            fontSize = 16.sp,
            fontWeight = FontWeight.Bold
        )
    }
}
```

Activities.kt

```
package com.zach.vit_20bce1690_bloodbankapp

sealed class Activities(val route: String){
    object Home : Activities("home")
    object Destination : Activities("destination")

    fun withArgs(vararg args: String): String{
        return buildString {
            append(route)
```

```
            args.forEach { arg ->
                append("/$arg")
            }
        }
    }
}
```