20BCE1458

AYUSH KAPRI

Blood Bank Application

```
package com.example.vit_20bce1458

import android.app.DatePickerDialog
import android.os.Build
import android.os.Bundle
import android.widget.Toast
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.annotation.RequiresApi
import androidx.compose.foundation.Image
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.DateRange
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.input.KeyboardType
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.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.window.Dialog
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import androidx.navigation.compose.rememberNavController
import java.text.SimpleDateFormat
import java.time.LocalDate
import java.time.ZoneId
import java.time.format.DateTimeFormatter
import java.util.*
import androidx.compose.material.icons.filled.Visibility
import androidx.compose.material.icons.filled.VisibilityOff
import com.example.vit_20bce1458.ui.theme.Vit_20BCE1458Theme

class MainActivity : ComponentActivity() {
    @RequiresApi(Build.VERSION_CODES.O)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Vit_20BCE1458Theme {
                // A surface container using the 'background' color from
the theme
```

```
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = Color(201, 85, 85, 255)
                ) {
                    BloodBankApp()
                }
            }
        }
    }
}

@RequiresApi(Build.VERSION_CODES.O)
@Composable
fun BloodBankApp() {
    val navController = rememberNavController()

    NavHost(navController, startDestination = "login") {
        composable("login") { Bank(navController) }
        composable("signIn") { SignIn(navController) }
        //composable("signUp") { SignUp(navController)}
    }
}



@Composable
fun Bank(navController: NavHostController) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf(TextFieldValue()) }
    val context = LocalContext.current

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(horizontal = 16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    )
    {
        Image(
            painter = painterResource(id = R.drawable.logo),
            contentDescription = null,
            Modifier.height(height = 118.dp)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Welcome Donar", fontSize = 34.sp)
        Spacer(modifier = Modifier.height(464.dp))
    }
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(horizontal = 24.dp),
        horizontalAlignment = Alignment.Start,
        verticalArrangement = Arrangement.Center
    )
    {
        Spacer(modifier = Modifier.height(123.dp))
        Text(text = "Username", fontSize = 24.sp)
        OutlinedTextField(
            value = username,
            onValueChange = { username = it },
```

```
            modifier = Modifier.fillMaxWidth(),

            )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Password", fontSize = 24.sp)
        var passwordVisibility by remember { mutableStateOf(false) }

        OutlinedTextField(
            value = password,
            onValueChange = { password = it },
            modifier = Modifier.fillMaxWidth(),
            label = { Text("Password") },
            visualTransformation = if (passwordVisibility)
VisualTransformation.None else PasswordVisualTransformation(),
            keyboardOptions = KeyboardOptions(keyboardType =
KeyboardType.Password),
            trailingIcon = {
                IconButton(onClick = { passwordVisibility =
!passwordVisibility }) {
                    Icon(
                        imageVector = if (passwordVisibility)
Icons.Filled.Visibility else Icons.Filled.VisibilityOff,
                        contentDescription = if (passwordVisibility) "Hide
Password" else "Show Password"
                    )
                }
            }
        )
        Spacer(modifier = Modifier.height(16.dp))
        Row(
            modifier = Modifier.fillMaxWidth(),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center
        ) {
            Text(text = "Donate Blood", fontSize = 18.sp)
            Text(text = "Save Life", fontSize = 18.sp, modifier =
Modifier.padding(start = 4.dp))
        }
        Spacer(modifier = Modifier.height(24.dp))
        Button(
            onClick = {
                Toast.makeText(
                    context,
                    "${username}\n${password}",
                    Toast.LENGTH_LONG
                ).show()
            },
            modifier = Modifier.fillMaxWidth(),
            colors = ButtonDefaults.buttonColors(Color.DarkGray)
        ) {
            Text(
                text = "Sign In",
                fontSize = 24.sp,
                color = Color.White,
                modifier = Modifier
                    .padding(start = 4.dp)
                    .clickable { navController.navigate("signIn") }
            )
        }
        Spacer(modifier = Modifier.height(34.dp))
        Row(
```

```
            modifier = Modifier.fillMaxWidth(),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center
        ) {
            Text(text = "New Member?", fontSize = 18.sp)
            Text(
                text = "Sign Up",
                fontSize = 18.sp,
                color = Color.Blue,
                modifier = Modifier
                    .padding(start = 14.dp)
                    .clickable { navController.navigate("signUp") }
            )
        }
    }
}
@RequiresApi(Build.VERSION_CODES.O)
@Composable
fun SignIn(navController: NavHostController) {
    var name by remember { mutableStateOf("") }
    var age by remember { mutableStateOf("") }
    var group by remember { mutableStateOf("") }
    var lastDonationDate by remember { mutableStateOf<LocalDate?>(null) }
    val context = LocalContext.current

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(horizontal = 24.dp),
        horizontalAlignment = Alignment.Start,
        verticalArrangement = Arrangement.Center
    ) {
        Text(text="Scheduling Blood Donation", fontSize = 24.sp)
        Spacer(modifier = Modifier.height(64.dp))
        Text(text = "Name", fontSize = 24.sp)
        OutlinedTextField(
            value = name,
            onValueChange = { name = it },
            modifier = Modifier.fillMaxWidth(),

            )
        Spacer(modifier = Modifier.height(16.dp))
        Text(text = "Age", fontSize = 24.sp)
        OutlinedTextField(
            value = age,
            onValueChange = { age = it },
            modifier = Modifier.fillMaxWidth(),
        )
        Text(text = "Blood Group", fontSize = 24.sp)
        OutlinedTextField(
            value = group,
            onValueChange = { group = it },
            modifier = Modifier.fillMaxWidth(),

            )
        Spacer(modifier = Modifier.height(16.dp))
        val selectedDate = remember { mutableStateOf<LocalDate?>(null) }
        var isDatePickerVisible by remember { mutableStateOf(false) }

        Text(text = "Date of Donation", fontSize = 24.sp)
        Box(modifier = Modifier.clickable { isDatePickerVisible = true }) {
```

```
            Text(
                text =
selectedDate.value?.format(DateTimeFormatter.ofPattern("dd-MM-yyyy")) ?:
"Select Last Donation Date",
                modifier = Modifier.padding(top = 16.dp),
                color = if (selectedDate.value != null) Color.Black else
Color.LightGray
            )
        }


        if (isDatePickerVisible) {
            Dialog(
                onDismissRequest = { isDatePickerVisible = false }
            ) {
                DatePicker(
                    selectedDate = selectedDate,
                    onDateSelected = { date ->
                        selectedDate.value = date
                        isDatePickerVisible = false
                    }
                )
            }
            Spacer(modifier = Modifier.height(26.dp))
        }
        Spacer(modifier = Modifier.height(24.dp))
        Button(
            onClick = {
                Toast.makeText(
                    context,
                    "Scheduled",
                    Toast.LENGTH_LONG
                ).show()
                name = ""
                age = ""
                group = ""
                selectedDate.value = null
            }
        )
        {
            Text(text = "Submit")
        }
    }
}
@RequiresApi(Build.VERSION_CODES.O)
@Composable
fun DatePicker(selectedDate: MutableState<LocalDate?>, onDateSelected:
(LocalDate) -> Unit) {
    val context = LocalContext.current
    val dateFormatter = remember { SimpleDateFormat("dd-MM-yyyy",
Locale.getDefault()) }
    val calendar = Calendar.getInstance()

    val onClick: () -> Unit = {
        val datePickerDialog = DatePickerDialog(
            context,
            { _, year, month, day ->
                calendar.set(year, month, day)
                val date = calendar.time
                selectedDate.value =
date.toInstant().atZone(ZoneId.systemDefault()).toLocalDate()
```

```
            },
            calendar.get(Calendar.YEAR),
            calendar.get(Calendar.MONTH),
            calendar.get(Calendar.DAY_OF_MONTH)
        )
        datePickerDialog.show()
    }

    TextField(
        value = selectedDate.value?.let {
dateFormatter.format(Date.from(it.atStartOfDay(ZoneId.systemDefault()).toIn
stant())) } ?: "",
        onValueChange = { },
        label = { Text(text = "Select Date") },
        readOnly = true,
        trailingIcon = {
            IconButton(onClick = onClick) {
                Icon(imageVector = Icons.Default.DateRange,
contentDescription = "Select Date")
            }
        }
    )
}
```