Assignment – 1

Name: MD SHABREZ

Register Number: 20BCE1690

Google Signup Page UI Design

```
package com.zach.vit_20bce1690_googlesignuppageuidesign

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
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.Row
import androidx.compose.foundation.layout.Spacer
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.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Settings
import androidx.compose.material3.AlertDialog
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
```

```
import androidx.compose.material3.Checkbox
import androidx.compose.material3.CheckboxDefaults
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.MenuDefaults
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.MutableState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
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.unit.dp
import androidx.compose.ui.unit.sp
import
com.zach.vit_20bce1690_googlesignuppageuidesign.ui.theme.VIT_20BCE1690_Goo
gleSignupPageuiDesignTheme

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            VIT_20BCE1690_GoogleSignupPageuiDesignTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
```

```
                ) {
                    App()
                }
            }
        }
    }
}

@Composable
fun App() {
    val showDialog = remember { mutableStateOf(false) }

    val dialogText = remember { mutableStateOf("") }

    val primaryTextColor = remember {
        mutableStateOf(Color(32, 33, 36))
    }

    val secondaryTextColor = remember {
        mutableStateOf(Color(26, 115, 232))
    }

    val tertiaryTextColor = remember {
        mutableStateOf(Color(95, 99, 104))
    }

    val primaryBackground = remember {
        mutableStateOf(Color.White)
    }

    val secondaryBackground = remember {
        mutableStateOf(Color.White)
    }

    val showPass = remember {
        mutableStateOf(false)
    }

    val firstName = remember {
        mutableStateOf(TextFieldValue())
    }
```

```
    val lastName = remember {
        mutableStateOf(TextFieldValue())
    }

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

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

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

    fun validateTextFields() {
        val firstNameText = firstName.value.text
        val lastNameText = lastName.value.text
        val userNameText = userName.value.text
        val passwordText = password.value.text
        val confirmText = confirm.value.text

        if (firstNameText.isBlank()) {
            showDialog.value = true
            dialogText.value = "Please enter your first name."
            return
        }

        if (lastNameText.isBlank()) {
            showDialog.value = true
            dialogText.value = "Please enter your last name."
            return
        }

        if (userNameText.isBlank()) {
            showDialog.value = true
            dialogText.value = "Please enter a username."
            return
        }

        if (!userNameText.matches("[a-zA-Z0-9.]+".toRegex())) {
```

```
            showDialog.value = true
            dialogText.value =
                "Username should only contain letters, numbers, and full
stops."
            return
        }

        if (passwordText.isBlank()) {
            showDialog.value = true
            dialogText.value = "Please enter a password."
            return
        }

        if (passwordText.length < 8 || !passwordText.matches("^(?=.*[a-zA-
Z])(?=.*\\d)(?=.*[!@#\$%^&*()_+\\-
=\\[\\]{};':\"\\\\|,.<>/?]).+\$".toRegex())) {
            showDialog.value = true
            dialogText.value =
                "Password should be at least 8 characters long and include
a mix of letters, numbers, and symbols."
            return
        }

        if (confirmText.isBlank()) {
            showDialog.value = true
            dialogText.value = "Please confirm your password."
            return
        }

        if (passwordText != confirmText) {
            showDialog.value = true
            dialogText.value = "Password and confirm password do not
match."
            return
        }

        showDialog.value = true
        dialogText.value = "All fields are valid."
    }

    val showDropdown = remember { mutableStateOf(false) }
```

```
    Column(
        Modifier
            .fillMaxSize()
            .background(primaryBackground.value), verticalArrangement =
        Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Box(
            modifier = Modifier
                .background(secondaryBackground.value)
                .fillMaxWidth()
                .wrapContentSize(Alignment.TopEnd)
        ) {
            IconButton(onClick = { showDropdown.value =
!showDropdown.value }) {
                Icon(
                    imageVector = Icons.Default.Settings,
contentDescription = "More",
                    tint = primaryTextColor.value
                )
            }

            DropdownMenu(expanded = showDropdown.value, onDismissRequest =
{
                showDropdown
                    .value = false
            }, modifier = Modifier.background(primaryBackground.value)) {

                DropdownMenuItem(
                    text = {
                        Text(
                            "Light Mode",
                            color = primaryTextColor.value
                        )
                    },
                    onClick = {
                        primaryTextColor.value = Color(32, 33, 36)

                        secondaryTextColor.value = Color(26, 115, 232)

                        tertiaryTextColor.value = Color(95, 99, 104)
```

```
                        primaryBackground.value = (Color.White)

                        secondaryBackground.value = Color.White

                    },
                    colors = MenuDefaults.itemColors(Color.Black)
                )

                DropdownMenuItem(
                    text = { Text("Dark Mode", color =
primaryTextColor.value) },
                    onClick = {
                        primaryTextColor.value = Color(211, 207, 201)

                        secondaryTextColor.value = Color(48, 146, 234)

                        tertiaryTextColor.value = Color(169, 162, 151)

                        primaryBackground.value = Color(19, 21, 22)

                        secondaryBackground.value = Color(16, 18, 19, 255)
                    },
                    colors = MenuDefaults.itemColors(Color.Black)
                )
            }
        }

        Card(
            shape = RoundedCornerShape(5.dp), elevation =
CardDefaults.cardElevation(
                defaultElevation = 10.dp

            ),
            colors = CardDefaults.cardColors(primaryBackground.value),
            modifier = Modifier
                .verticalScroll(rememberScrollState())
                .padding(vertical = 10.dp)
        ) {
            Column(
                Modifier
                    .padding(horizontal = 25.dp, vertical = 10.dp)
                    .background(color = primaryBackground.value)
```

```
                    .fillMaxWidth(0.98f)
            ) {
                Image(
                    painterResource(id = R.drawable.google),
contentDescription = "google " +
                            "logo", modifier = Modifier.size(100.dp)
                )
                Text(
                    "Create your Google Account",
                    color = primaryTextColor.value,
                    fontFamily =
                    FontFamily.SansSerif,
                    fontSize = 24.sp
                )

                Spacer(modifier = Modifier.height(40.dp))
                CustomTextField(
                    modifier = Modifier.fillMaxWidth(),
                    mutableValue =
                    firstName,
                    label = "First Name",
                    placeholder =
                    "First Name",
                    secondaryTextColor.value,
                    textColor = tertiaryTextColor.value
                )
                Spacer(modifier = Modifier.height(20.dp))
                CustomTextField(
                    modifier = Modifier.fillMaxWidth(),
                    mutableValue =
                    lastName,
                    label = "Last Name",
                    focusedColor = secondaryTextColor.value,
                    textColor = tertiaryTextColor.value
                )
                Spacer(modifier = Modifier.height(20.dp))
                CustomTextField(
                    modifier = Modifier.fillMaxWidth(), mutableValue =
                    userName, label = "Username", placeholder =
                    "Username", secondaryTextColor.value,
                    isTrail = true, textColor = tertiaryTextColor.value
                )
```

```
                Row(modifier = Modifier.padding(horizontal = 15.dp)) {
                    Text(
                        "You can use letters, numbers and full stops",
                        color = tertiaryTextColor.value, fontFamily =
                        FontFamily.SansSerif, fontSize = 14.sp
                    )
                }

                Spacer(modifier = Modifier.height(20.dp))
                Text(
                    "Use my current email address instead",
                    color = secondaryTextColor.value,
                    fontFamily =
                    FontFamily.SansSerif,
                    fontSize = 16.sp,
                    fontWeight = FontWeight.Bold,
                    modifier = Modifier
                        .clickable(enabled = true, onClick = {
                            userName.value =
TextFieldValue("md.shabrez2020")
                        })
                )
                Spacer(modifier = Modifier.height(40.dp))
                CustomTextField(
                    mutableValue = password,
                    label = "Password",
                    focusedColor = secondaryTextColor.value,
                    modifier = Modifier.fillMaxWidth(),
                    isHideVal = !showPass.value,
                    textColor = tertiaryTextColor.value
                )
                Spacer(modifier = Modifier.height(20.dp))
                CustomTextField(
                    mutableValue = confirm,
                    label = "Confirm",
                    focusedColor = secondaryTextColor.value,
                    modifier = Modifier.fillMaxWidth(),
                    isHideVal = !showPass.value,
                    textColor = tertiaryTextColor.value
                )
                Row(modifier = Modifier.padding(horizontal = 15.dp)) {
                    Text(
```

```
                        "Use 8 or more characters with a mix of letters,
numbers & symbols",
                        color = tertiaryTextColor.value, fontFamily =
                        FontFamily.SansSerif, fontSize = 14.sp
                    )
                }
                Row(verticalAlignment = Alignment.CenterVertically) {
                    Checkbox(
                        checked = showPass.value, onCheckedChange = {
                            showPass.value =
                                !showPass
                                    .value
                        },
                        colors = CheckboxDefaults.colors(
                            checkedColor = Color
                                (
                                26, 115,
                                232
                            )
                        )
                    )

                    Text(
                        text = "Show password", color =
primaryTextColor.value, fontFamily =
                        FontFamily.SansSerif, fontSize = 16.sp
                    )
                }
                Spacer(modifier = Modifier.height(40.dp))
                Row(
                    horizontalArrangement = Arrangement.SpaceBetween,
                    modifier = Modifier.fillMaxWidth()
                ) {
                    CustomButton(
                        buttonText = "Sign in instead", textColor =
secondaryTextColor
                            .value, backgroundColor =
primaryBackground.value
                    )
                    CustomButton(buttonText = "Next", onClick = {
validateTextFields() })
                }
```

```
            }

            if (showDialog.value) {
                AlertDialog(
                    onDismissRequest = { showDialog.value = false },
                    title = { Text("Alert", color =
primaryTextColor.value) },
                    text = { Text(dialogText.value, color =
tertiaryTextColor.value) },
                    confirmButton = {
                        CustomButton(
                            buttonText = "Ok",
                            onClick = { showDialog.value = false })
                    },
                    shape = RoundedCornerShape(5.dp),
                    containerColor = primaryBackground.value
                )
            }
        }
    }
}

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CustomTextField(
    modifier: Modifier = Modifier,
    mutableValue: MutableState<TextFieldValue>, label: String,
    placeholder: String
    = label,
    focusedColor: Color, isTrail: Boolean = false,
    isHideVal: Boolean = false, textColor: Color
) {
    OutlinedTextField(
        modifier = modifier,
        value = mutableValue.value,
        onValueChange = { mutableValue.value = it },
        label = { Text(text = label) },
        placeholder = { Text(text = placeholder) },
        colors = TextFieldDefaults.outlinedTextFieldColors(
            focusedBorderColor = focusedColor,
            focusedLabelColor = focusedColor,
            placeholderColor = Color.Transparent,
```

```
            textColor = textColor,
            unfocusedBorderColor = textColor,
            unfocusedLabelColor = textColor,
            unfocusedLeadingIconColor = textColor,
            focusedLeadingIconColor = textColor

        ),
        trailingIcon = {
            if (isTrail) {
                Text(
                    text = "@gmail.com", color = textColor, fontFamily =
                    FontFamily.SansSerif, fontSize = 17.sp, modifier =
Modifier.padding
                        (horizontal = 15.dp)
                )
            }

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


    )
}

@Composable
fun CustomButton(
    buttonText: String,
    textColor: Color = Color.White,
    backgroundColor: Color = Color(26, 115, 232),
    onClick: () -> Unit = {}
) {
    Button(
        onClick = onClick,
        shape = RoundedCornerShape(5.dp),
        colors = ButtonDefaults.buttonColors(backgroundColor)
    ) {
        Text(
            buttonText,
```

```
            color = textColor,
            fontSize = 16.sp,
            fontWeight = FontWeight.Bold
        )
    }
}
```