**Lab Exercise 8- Typography in Jetpack Compose**

**Objective:**

In this lab, you will learn how to define and apply custom typography styles in a Jetpack Compose application. You will customize text styles such as headlines, body text, and buttons, and apply them to UI components.

**Steps:**

**Step 1: Set up a New Jetpack Compose Project**

1. Open **Android Studio**.
2. Create a new project:
   * Select **New Project** -> **Empty Compose Activity**.
   * Set the project name to TypographyDemoCompose and finish the setup.
3. Ensure that your project includes the necessary Jetpack Compose dependencies in build.gradle:

**Step 2: Define Custom Typography in Jetpack Compose**

1. Navigate to the ui/theme folder and open the **Theme.kt** file.
2. Define custom typography styles by creating a Typography object.

import androidx.compose.material3.Typography

import androidx.compose.ui.text.TextStyle

import androidx.compose.ui.text.font.FontFamily

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.unit.sp

// Define custom typography styles

val CustomTypography = Typography(

headlineLarge = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.Bold,

fontSize = 30.sp

),

bodyLarge = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.Normal,

fontSize = 18.sp

),

labelLarge = TextStyle(

fontFamily = FontFamily.Serif,

fontWeight = FontWeight.Medium,

fontSize = 14.sp

)

// You can add more custom text styles here

)

@Composable

fun MyAppTheme(content: @Composable () -> Unit) {

MaterialTheme(

typography = CustomTypography, // Apply the custom typography

colorScheme = lightColorScheme(),

content = content

)

}

**Step 3: Use Custom Typography in Your UI**

1. Open the MainActivity.kt file and modify the content to use the custom typography:

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.\*

import androidx.compose.material3.\*

import androidx.compose.runtime.Composable

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

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

import androidx.compose.ui.unit.dp

import com.example.typographydemo.ui.theme.MyAppTheme

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

MyAppTheme {

TypographyDemoScreen()

}

}

}

}

@Composable

fun TypographyDemoScreen() {

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.spacedBy(16.dp),

horizontalAlignment = Alignment.CenterHorizontally

) {

// Headline text with custom typography

Text(

text = "Headline Text",

style = MaterialTheme.typography.headlineLarge

)

// Body text with custom typography

Text(

text = "This is body text with custom font size and weight.",

style = MaterialTheme.typography.bodyLarge

)

// Label text with custom typography

Text(

text = "Label Text",

style = MaterialTheme.typography.labelLarge

)

// Button with custom typography

Button(onClick = {}) {

Text(text = "Custom Button Text", style = MaterialTheme.typography.labelLarge)

}

}

}

@Preview(showBackground = true)

@Composable

fun TypographyDemoPreview() {

MyAppTheme {

TypographyDemoScreen()

}

}

**Explanation:**

1. **Custom Typography**: You defined custom text styles using Typography in Theme.kt. These include headlineLarge, bodyLarge, and labelLarge with different font sizes, weights, and font families.
2. **Using Custom Styles**: You applied the custom typography styles to different UI elements:
   * **Text Composables**: Text composables display the headlineLarge, bodyLarge, and labelLarge styles.
   * **Button**: The button text also uses a custom typography style (labelLarge).
3. **MaterialTheme.typography**: You access your custom typography by using MaterialTheme.typography and specifying the style you want to apply.

**Step 4: Run the Application**

1. Select an emulator or a physical device.
2. Run the project.
3. The app should display:
   * A headline text with large, bold font.
   * A body text with a normal weight and larger font size.
   * A label text with medium weight and a smaller font size.
   * A button using the custom typography for the label.