**Lab Exercise 6- Material Theming in Jetpack Compose**

**Objective:**

* Learn how to apply Material theming to your Jetpack Compose project.
* Customize themes with colors, typography, and shapes.
* Use Material components like Button, Card, and Text with your custom theme.

**Step 1: Set Up a New Project**

1. **Open Android Studio**.
2. **Create a new project** with **Empty Compose Activity**.
3. Set the **Minimum SDK** to 21 or higher and **Click Finish**.

**Step 2: Understanding the Default Theme**

When you create a new Jetpack Compose project, it comes with a default Material theme. Let’s explore this:

1. Open MainActivity.kt and locate the setContent block, which looks like this by default:

setContent {

MyApplicationTheme {

// A surface container using the 'background' color from the theme

Surface(modifier = Modifier.fillMaxSize(), color = MaterialTheme.colorScheme.background) {

Greeting("Android")

}

}

}

* MyApplicationTheme: This is the default theme provided by the template.
* MaterialTheme.colorScheme.background: Uses the background color from the theme.

**Step 3: Explore the Theme Files**

1. Open the **Theme.kt** file inside the ui/theme package. This file contains the Material theming setup.

You will see these key elements:

* **Colors**: LightColors and DarkColors define the color schemes for light and dark themes.
* **Typography**: Defines the text styles used throughout the app.
* **Shapes**: Defines the shape styles for UI elements like buttons, cards, etc.

**Step 4: Apply and Modify Colors in the Theme**

Let's modify the color scheme of the app.

1. Open Color.kt in the ui/theme package and modify the colors like this:

val Purple80 = Color(0xFFBB86FC)

val PurpleGrey80 = Color(0xFF6200EE) // Modify to a new shade of purple

val Pink80 = Color(0xFF03DAC5)

val Purple40 = Color(0xFF6200EE)

val PurpleGrey40 = Color(0xFF3700B3) // Modify to a new shade of grey

val Pink40 = Color(0xFF018786)

1. Now apply these colors in the theme inside Theme.kt:

private val LightColorScheme = lightColorScheme(

primary = Purple80,

secondary = PurpleGrey80,

background = Color.White,

surface = Color.White,

onPrimary = Color.White,

onSecondary = Color.Black,

)

private val DarkColorScheme = darkColorScheme(

primary = Purple40,

secondary = PurpleGrey40,

background = Color.Black,

surface = Color.Black,

onPrimary = Color.White,

onSecondary = Color.White,

)

**Step 5: Using Material Components with the Custom Theme**

1. In MainActivity.kt, replace the existing UI with a **Button** and **Card**:

@Composable

fun ThemedComponents() {

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Button(onClick = { /\* TODO \*/ }, colors = ButtonDefaults.buttonColors(containerColor = MaterialTheme.colorScheme.primary)) {

Text(text = "Click Me", color = MaterialTheme.colorScheme.onPrimary)

}

Spacer(modifier = Modifier.height(16.dp))

Card(

modifier = Modifier.fillMaxWidth(),

colors = CardDefaults.cardColors(containerColor = MaterialTheme.colorScheme.secondary)

) {

Text(

text = "This is a Card",

modifier = Modifier.padding(16.dp),

color = MaterialTheme.colorScheme.onSecondary

)

}

}

}

@Preview(showBackground = true)

@Composable

fun ThemedComponentsPreview() {

MyApplicationTheme {

ThemedComponents()

}

}

1. **Explanation**:
   * ButtonDefaults.buttonColors: Customizes the button color to use the primary color from the theme.
   * CardDefaults.cardColors: Customizes the card color to use the secondary color from the theme.
   * MaterialTheme.colorScheme.primary and MaterialTheme.colorScheme.secondary: Apply theme colors.
2. **Run the app** to see your themed button and card.

**Step 6: Customizing Typography**

1. Open Typography.kt and modify the typography:

val Typography = Typography(

bodyLarge = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.Normal,

fontSize = 16.sp

),

titleLarge = TextStyle(

fontFamily = FontFamily.Serif, // Change to Serif font

fontWeight = FontWeight.Bold,

fontSize = 20.sp,

letterSpacing = 0.15.sp

)

)

1. In MainActivity.kt, modify the Text composable to use this new typography:

Text(

text = "This is a Card",

style = MaterialTheme.typography.titleLarge, // Apply new titleLarge style

modifier = Modifier.padding(16.dp),

color = MaterialTheme.colorScheme.onSecondary

)

1. **Run the app** again to see the updated typography.

**Step 7: Customizing Shapes**

1. Open Shape.kt and modify the shape definitions:

val Shapes = Shapes(

small = RoundedCornerShape(8.dp), // Set small shapes to 8dp rounded corners

medium = RoundedCornerShape(16.dp), // Set medium shapes to 16dp rounded corners

large = RoundedCornerShape(24.dp) // Set large shapes to 24dp rounded corners

)

1. Apply this shape to the Button in MainActivity.kt:

Button(

onClick = { /\* TODO \*/ },

colors = ButtonDefaults.buttonColors(containerColor = MaterialTheme.colorScheme.primary),

shape = MaterialTheme.shapes.medium // Apply medium shape

) {

Text(text = "Click Me", color = MaterialTheme.colorScheme.onPrimary)

}

1. **Run the app** to see the button with the custom rounded corners.

**Step 8: Dark Mode Support**

1. Modify the theme to support dark mode by adjusting the isSystemInDarkTheme() parameter:

@Composable

fun MyApplicationTheme(

darkTheme: Boolean = isSystemInDarkTheme(), // Automatically switch between light and dark themes

content: @Composable () -> Unit

) {

val colorScheme = if (darkTheme) {

DarkColorScheme

} else {

LightColorScheme

}

MaterialTheme(

colorScheme = colorScheme,

typography = Typography,

shapes = Shapes,

content = content

)

}

1. **Test dark mode** by enabling dark mode on your device or emulator.