**Lab Exercise 10- Snackbar in Jetpack Compose**

**Objective:**

Learn how to display a **Snackbar** in **Jetpack Compose** when interacting with UI components like buttons. The exercise will also cover how to manage the lifecycle of the Snackbar and handle user actions like dismissing it.

**Prerequisites:**

* Basic understanding of **Kotlin** and **Jetpack Compose**.
* Familiarity with Compose components such as Button and Scaffold.

**Steps:**

**Step 1: Set up a new Jetpack Compose project**

1. Open **Android Studio**.
2. Create a new project by selecting **New Project** -> **Empty Compose Activity**.
3. Set the project name and finish the setup.
4. Make sure you have the correct dependencies for Jetpack Compose in your build.gradle:

dependencies {

implementation "androidx.activity:activity-compose:1.7.0"

implementation "androidx.compose.material:material:1.4.0"

implementation "androidx.compose.material3:material3:1.1.0"

}

**Step 2: Basic Snackbar Implementation**

In Jetpack Compose, **Snackbar** is usually displayed using a Scaffold, which provides a structure for UI components like Snackbar.

1. Open the MainActivity.kt file.
2. Modify your code to show a Snackbar when a button is clicked.

Here's an example:

package com.example.demosnackbar2  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.activity.enableEdgeToEdge  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.foundation.layout.padding  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.\*  
import androidx.compose.runtime.Composable  
import androidx.compose.runtime.remember  
import androidx.compose.runtime.rememberCoroutineScope  
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.demosnackbar2.ui.theme.DemoSnackBar2Theme  
import kotlinx.coroutines.launch

class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 *setContent* **{** SnackbarExample()  
 **}** }  
}  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun SnackbarExample() {  
 *// State to control Snackbar visibility* val snackbarHostState = remember **{** SnackbarHostState() **}** val scope = rememberCoroutineScope()  
  
 *// Scaffold provides the basic layout with top bar, bottom bar, and Snackbar* Scaffold(  
 snackbarHost = **{** SnackbarHost(hostState = snackbarHostState)  
 **}**,  
 content = **{** padding **->** Box(  
 contentAlignment = Alignment.Center,  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(padding)  
 ) **{** *// Button to trigger Snackbar* Button(  
 onClick = **{** *// Show Snackbar using Coroutine to display asynchronously* scope.*launch* **{** snackbarHostState.showSnackbar(  
 message = "This is a Material 3 Snackbar!",  
 actionLabel = "Dismiss"  
 )  
 **}  
 }**,  
 modifier = Modifier.*padding*(16.*dp*)  
 ) **{** Text(text = "Show Snackbar")  
 **}  
 }  
 }** )  
}  
  
@Preview(showBackground = true)  
@Composable  
fun SnackbarExamplePreview()  
{  
 SnackbarExample()  
}

**Step 3: Run the Application**

1. Run the app on an emulator or a physical device.
2. Click the **"Show Snackbar"** button.
3. A Snackbar message will appear at the bottom of the screen with the text **"Hello from Snackbar"** and an **action** button labeled **"Dismiss"**.

**Explanation:**

1. **Scaffold**: This composable provides a structure to place UI components like top bars, floating action buttons, and snackbars.
2. **ScaffoldState and SnackbarHostState**: The ScaffoldState holds the state for displaying a Snackbar using SnackbarHostState. This allows the Snackbar to be controlled programmatically.
3. **CoroutineScope**: A Snackbar has to be launched using a coroutine because it suspends the execution until the Snackbar is dismissed.
4. **showSnackbar()**: This function displays the Snackbar with a message and an optional action (like a dismiss button).

**Lab Exercise Tasks:**

**Task 1: Customize Snackbar Behavior**

Modify the existing code to include a custom duration for the Snackbar.

* Use SnackbarDuration.Short, SnackbarDuration.Long, or SnackbarDuration.Indefinite to control how long the Snackbar is visible.

**Example:**

scaffoldState.snackbarHostState.showSnackbar(

message = "Snackbar with custom duration",

actionLabel = "Dismiss",

duration = SnackbarDuration.Long

)

**Task 2: Add Snackbar with Action**

Modify the code to perform an action when the user clicks the **"Dismiss"** button in the Snackbar.

* Use the return value of showSnackbar() to detect if the action was clicked.

**Example:**

val result = scaffoldState.snackbarHostState.showSnackbar(

message = "Undo the action?",

actionLabel = "Undo"

)

if (result == SnackbarResult.ActionPerformed) {

// Perform undo operation

}