**Lab Exercise 11- BottomSheet in Jetpack Compose**

**Objective:**

Learn how to create and manage a **Modal BottomSheet** using Jetpack Compose. You'll build a simple UI that displays a BottomSheet when a button is clicked, allowing the user to interact with it.

**Prerequisites:**

* Basic understanding of **Kotlin** and **Jetpack Compose**.
* Familiarity with composables such as Button and state management in Compose.

**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. Ensure your build.gradle file contains the necessary Jetpack Compose dependencies:

dependencies {

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

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

}

**Step 2: Create a BottomSheet Layout**

In Jetpack Compose, we use the ModalBottomSheetLayout to create a bottom sheet. The sheet’s state is managed by ModalBottomSheetState.

Here’s how you can display a **Modal BottomSheet** when a button is clicked:

1. Open your MainActivity.kt file.
2. Modify your code to include a button that opens the **BottomSheet** when clicked.

import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.activity.enableEdgeToEdge  
import androidx.compose.foundation.layout.Arrangement  
import androidx.compose.foundation.layout.Box  
import androidx.compose.foundation.layout.Column  
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.material3.Button  
import androidx.compose.material3.ExperimentalMaterial3Api  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.ModalBottomSheet  
import androidx.compose.material3.Scaffold  
import androidx.compose.material3.SheetValue  
import androidx.compose.material3.Text  
import androidx.compose.material3.rememberModalBottomSheetState  
import androidx.compose.runtime.Composable  
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.demobottomsheetnew.ui.theme.DemoBottomSheetNewTheme  
import kotlinx.coroutines.launch  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 *setContent* **{** BottomSheetExample()  
 **}** }  
}  
  
  
@OptIn(ExperimentalMaterial3Api::class)  
@Composable  
fun BottomSheetExample() {  
 *// State to control the visibility of the BottomSheet* val bottomSheetState = rememberModalBottomSheetState(  
 skipPartiallyExpanded = true *// Controls whether to skip the partially expanded state* )  
 val scope = rememberCoroutineScope()  
  
 *// Scaffold to hold the screen content and BottomSheet* Scaffold(  
 content = **{** padding **->** Box(  
 contentAlignment = Alignment.Center,  
 modifier = Modifier  
 .*fillMaxSize*()  
 .*padding*(padding)  
 ) **{** *// Button to trigger the BottomSheet* Button(onClick = **{** scope.*launch* **{** bottomSheetState.show()  
 **}  
 }**) **{** Text("Show Bottom Sheet")  
 **}  
 }  
 }** )  
  
 *// Modal BottomSheet displaying some content* if (bottomSheetState.isVisible) {  
 ModalBottomSheet(  
 onDismissRequest = **{** scope.*launch* **{** bottomSheetState.hide() **}  
 }**,  
 sheetState = bottomSheetState  
 ) **{** *// Content inside the BottomSheet* Column(  
 modifier = Modifier  
 .*fillMaxWidth*()  
 .*padding*(16.*dp*),  
 verticalArrangement = Arrangement.Center,  
 horizontalAlignment = Alignment.CenterHorizontally  
 ) **{** Text(text = "This is a Bottom Sheet", style = MaterialTheme.typography.headlineSmall)  
 Spacer(modifier = Modifier.*height*(16.*dp*))  
 Button(onClick = **{** scope.*launch* **{** bottomSheetState.hide() **}  
 }**) **{** Text(text = "Close Bottom Sheet")  
 **}  
 }  
 }** }  
}  
  
@Preview(showBackground = true)  
@Composable  
fun BottomSheetExamplePreview()  
{  
 BottomSheetExample()  
}

**Step 3: Run the Application**

1. Run the app on an emulator or a physical device.
2. Click the **"Show Bottom Sheet"** button.
3. A BottomSheet will slide up from the bottom of the screen with the text **"This is a Bottom Sheet"** and a button labeled **"Close Sheet"**.
4. Clicking the **"Close Sheet"** button will hide the BottomSheet.

**Explanation:**

1. **ModalBottomSheetLayout**: This is a layout composable that defines the bottom sheet's content. It takes in the sheetState to control the visibility of the BottomSheet.
2. **ModalBottomSheetState**: This holds the state of the bottom sheet, such as whether it is hidden or expanded. You can programmatically show or hide the sheet using coroutines.
3. **CoroutineScope**: The bottomSheetState.show() and bottomSheetState.hide() methods are suspending functions, so you need to use a coroutine to execute them.
4. **Sheet Content**: The content inside the ModalBottomSheetLayout is defined in the sheetContent block. Here, you can add various UI elements like text, buttons, and more.

**Lab Exercise Tasks:**

**Task 1: Customize the BottomSheet Content**

* Modify the BottomSheet to display more content such as an image, multiple buttons, or text fields.

**Example:**

sheetContent = {

Column(

modifier = Modifier

.fillMaxWidth()

.padding(16.dp)

) {

Text(text = "More content in Bottom Sheet", style = MaterialTheme.typography.h6)

Spacer(modifier = Modifier.height(8.dp))

TextField(value = "Enter something", onValueChange = {}, label = { Text("Input Field") })

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

Button(onClick = { /\* Action \*/ }) {

Text("Action Button")

}

Spacer(modifier = Modifier.height(8.dp))

Button(onClick = {

coroutineScope.launch {

bottomSheetState.hide()

}

}) {

Text("Close Sheet")

}

}

}

**Task 2: Handle BottomSheet Lifecycle Events**

* Use LaunchedEffect to detect when the BottomSheet is opened or closed and display a **Snackbar** notifying the user.

**Example:**

LaunchedEffect(bottomSheetState.isVisible) {

if (bottomSheetState.isVisible) {

scaffoldState.snackbarHostState.showSnackbar("Bottom Sheet Opened")

} else {

scaffoldState.snackbarHostState.showSnackbar("Bottom Sheet Closed")

}

}

**Task 3: Add an Expandable BottomSheet**

* Modify the BottomSheet so that it starts partially expanded and can be swiped to fully expand.

**Additional Considerations:**

* **Animations**: You can customize how the BottomSheet animates when it opens or closes by modifying the state and handling gestures.
* **Handling Input**: If your BottomSheet contains input fields, be sure to manage focus appropriately when the sheet opens or closes.

By completing this lab exercise, you'll gain a solid understanding of how to create and interact with BottomSheets in Jetpack Compose. You'll also learn how to manage the sheet’s state and customize its content and behavior.