**Lab Exercise 13- State and MutableState in Jetpack Compose**

**Objective:**

In this lab, you will learn how to manage state in Jetpack Compose using State and MutableState. You will build a simple counter app that updates the UI when the state changes.

**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 StateDemoCompose and finish the setup.
3. Make sure your project includes the necessary Jetpack Compose dependencies in build.gradle:

**Step 2: Create a Simple Counter App Using MutableState**

1. Open the MainActivity.kt file.
2. Define a composable function that manages the state of the counter using MutableState.

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.compose.foundation.layout.\*

import androidx.compose.material3.Button

import androidx.compose.material3.Text

import androidx.compose.runtime.\*

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.statedemocompose.ui.theme.StateDemoComposeTheme

class MainActivity : ComponentActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

StateDemoComposeTheme {

CounterApp() // Call the CounterApp composable

}

}

}

}

@Composable

fun CounterApp() {

// Define a state variable to hold the counter value

var count by remember { mutableStateOf(0) }

// Layout for the UI

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

// Display the current count

Text(text = "Counter: $count", style = MaterialTheme.typography.headlineLarge)

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

// Button to increment the counter

Button(onClick = { count++ }) {

Text(text = "Increment")

}

}

}

@Preview(showBackground = true)

@Composable

fun CounterAppPreview() {

StateDemoComposeTheme {

CounterApp() // Preview the CounterApp composable

}

}

**Explanation:**

1. **State Management with MutableState**:
   * You use mutableStateOf(0) to create a state variable count that holds the current value of the counter. The by keyword allows the value to be updated directly.
   * The remember function is used to ensure that the state is retained across recompositions.
2. **Updating the State**:
   * Each time the **Increment** button is clicked, the counter is incremented by 1 (count++).
   * When the state changes, the UI is automatically recomposed to reflect the new value.

**Step 3: Run the Application**

1. Select an emulator or a physical device.
2. Run the project.
3. You should see a screen with:
   * The current count displayed as Counter: 0.
   * A button labeled **"Increment"**.
4. Press the button, and the counter should increment each time you press it, with the UI updating in real-time.

**Step 4: Understanding State in Jetpack Compose**

* **State in Compose**: In Jetpack Compose, UI is driven by state. Any changes to the state automatically trigger recomposition, updating the UI with the latest data.
* **MutableState and remember**:
  + mutableStateOf: Creates a mutable state holder that can be read and updated.
  + remember: Ensures that the state survives across recompositions, so the counter value doesn't reset each time the UI is redrawn.

**Step 5: Advanced Task - Add a Decrement Button**

* **Task 1**: Add a button to decrement the counter. Update the UI so the user can both increment and decrement the counter value.

@Composable

fun CounterApp() {

var count by remember { mutableStateOf(0) }

Column(

modifier = Modifier

.fillMaxSize()

.padding(16.dp),

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally

) {

Text(text = "Counter: $count", style = MaterialTheme.typography.headlineLarge)

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

// Increment Button

Row {

Button(onClick = { count++ }, modifier = Modifier.padding(end = 8.dp)) {

Text(text = "Increment")

}

// Decrement Button

Button(onClick = { count-- }) {

Text(text = "Decrement")

}

}

}

}

* **Task 2**: Prevent the counter from going below zero by adding a condition to the decrement logic.

Button(onClick = { if (count > 0) count-- }) {

Text(text = "Decrement")

}