Session Report

1. Introduction

This session focuses on controlling the User Interface (UI) using structured state in modern Android development with Jetpack Compose. Instead of manually updating UI elements, Compose automatically updates the screen when the state changes. This approach makes applications more dynamic, responsive, and easier to maintain.

2. Objectives of the Session

- Understand state management in Compose
- Learn how UI reacts to user interaction
- Handle navigation between screens
- Manage configuration changes (like screen rotation)
- Build dynamic multi-element interfaces

3. Tools and Technologies Used

- Android Studio
- Kotlin

4. Understanding Structured State in Compose

In Compose, UI is built using Composable functions. When state changes, the UI automatically recomposes (updates).

Important state concepts:

remember

mutableStateOf

rememberSaveable

Recomposition

5. Example : Simple Counter App (State Control)

This example demonstrates how UI updates when a button is clicked.

```
import androidx.compose.runtime.*
import androidx.compose.material3.*
import androidx.compose.foundation.layout.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp

@Composable
fun CounterScreen() {

    var count by remember { mutableStateOf(0) }

    Column(
        modifier = Modifier.fillMaxSize(),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text = "Count: $count")

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

        Button(onClick = { count++ }) {
            Text("Increase")
        }
    }
}
```

Explanation:

- remember { mutableStateOf(0) } stores state.
- When count changes, UI automatically updates.
- No manual refresh is required. 8. Key Concepts Learned
- State-driven UI updates
- Automatic recomposition
- Managing multiple UI elements
- Handling user interaction
- Preserving state during configuration changes

6. Challenges Faced

- Understanding recomposition
- Managing multiple state variables
- Preventing unnecessary recompositions
- Debugging UI state issues

7. Outcomes of the Session

- After completing this session, students are able to:
- Build dynamic and interactive UI screens
- Control UI behavior using structured state
- Handle configuration changes effectively
- Implement basic navigation logic
- Develop confidence in modern Android UI development

Conclusion

This session provided hands-on experience in building responsive and dynamic applications using structured state management in Compose. By understanding how state controls UI behavior, students gained practical knowledge of modern Android development practices. This foundation prepares them to build complex, multi-screen applications efficiently.