Unit 2 — Building App UI

Mobile Application Development — Lecture 2

Masoud Hamad

State University of Zanzibar (SUZA)

Semester II, 2025/2026

Outline

Jetpack Compose

- Android's **modern declarative UI toolkit**
- Replaces XML layouts with Kotlin code
- **UI = function of state** — when state changes, Compose re-renders ("recomposes")
- Fully interoperable with legacy View system
- Stable since 2021

Core Idea

You describe *what* the UI should look like for a given state — not *how* to mutate widgets.

Composable Functions

A `@Composable` function is the building block of Compose UI.

```
@Composable
fun Greeting(name: String) {
    Text(text = "Hello, $name!")
}
```

Rules

- Annotated with `@Composable`
- Can only be called from another `@Composable`
- Uses **PascalCase** (like a type)
- Returns `Unit` — describes UI, does not build widgets

Previewing Composables

```
@Preview(showBackground = true)
@Composable
fun GreetingPreview() {
    Greeting("Android")
}
```

Android Studio renders previews live in the IDE without running the app.

Text, Image, Button

```
Text(
    text = "Welcome",
    fontSize = 24.sp,
    fontWeight = FontWeight.Bold,
    color = Color.Blue
)

Image(
    painter = painterResource(R.drawable.logo),
    contentDescription = "SUZA logo"
)

Button(onClick = { /* action */ }) {
    Text("Click Me")
}
```

Modifiers

Modifiers decorate/configure a Composable — **order matters**.

```
Text(
    text = "Hello",
    modifier = Modifier
        .padding(16.dp)
        .background(Color.Yellow)
        .fillMaxWidth()
        .clickable { /* handle */ }
)
```

Common: `padding`, `background`, `size`, `fillMaxWidth`, `fillMaxHeight`, `fillMaxSize`, `clickable`, `border`, `clip`.

Column, Row, Box

```
Column(
    verticalArrangement = Arrangement.Center,
    horizontalAlignment = Alignment.CenterHorizontally
) {
    Text("Top"); Text("Middle"); Text("Bottom")
}

Row(horizontalArrangement = Arrangement.SpaceBetween) {
    Text("Left"); Text("Right")
}

Box(contentAlignment = Alignment.Center) {
    Image(...)
    Text("Overlay")
}
```

Spacer & Arrangement

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

Arrangement (main axis): `Start`, `Center`, `End`, `SpaceBetween`, `SpaceAround`, `SpaceEvenly`.

Alignment (cross axis):

- Row: `Top`, `CenterVertically`, `Bottom`
- Column: `Start`, `CenterHorizontally`, `End`

Resources

- **Strings:** `res/values/strings.xml` → `stringResource(R.string.greeting)`
- **Colors:** `res/values/colors.xml` or Material theme
- **Images:** `res/drawable/` → `painterResource(R.drawable.image)`
- **Dimensions:** use `dp` for sizes, `sp` for text

Material Design 3

```
MaterialTheme {
    Surface(color = MaterialTheme.colorScheme.background) {
        // your UI
    }
}
```

Ready-made components: `Card`, `Scaffold`, `TopAppBar`, `FloatingActionButton`, `Icon`, `Snackbar`.

State in Compose

State = any value that can change over time (counter, text input, checkbox).

UI = f(state)

When state changes, Compose automatically **recomposes** the affected UI.

```
@Composable
fun Counter() {
    var count by remember { mutableStateOf(0) }
    Column {
        Text("Count: $count")
        Button(onClick = { count++ }) { Text("Increment") }
    }
}
```

remember & rememberSaveable

```
var count by remember { mutableStateOf(0) }
// survives recomposition, lost on rotation

var name by rememberSaveable { mutableStateOf("") }
// survives recomposition AND config changes
```

TextField & State Hoisting

```
// GOOD -- state hoisted to parent
@Composable
fun EditableText(text: String, onTextChange: (String) -> Unit) {
    TextField(value = text, onValueChange = onTextChange,
              label = { Text("Enter name") })
}

@Composable
fun Parent() {
    var name by remember { mutableStateOf("") }
    EditableText(text = name, onTextChange = { name = it })
}
```

Pattern: **value** down, **events** up.

Business Card Composable

```
@Composable
fun BusinessCard() {
    Column(
        modifier = Modifier.fillMaxSize().padding(16.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Image(
            painter = painterResource(R.drawable.profile),
            contentDescription = null,
            modifier = Modifier.size(120.dp).clip(CircleShape)
        )
        Spacer(Modifier.height(16.dp))
        Text("Masoud Hamad", fontSize = 22.sp, fontWeight = FontWeight.Bold)
        Text("Lecturer, SUZA", color = Color.Gray)
    }
}
```

Summary

- Composables are Kotlin functions annotated `@Composable`
- Layouts built with `Column`, `Row`, `Box`
- Modifiers customize size, padding, background, click behaviour
- `remember { mutableStateOf }` makes composables reactive
- Hoist state to the lowest common ancestor (value down, events up)

Next: Lists & Material Design (LazyColumn, Card, Scaffold).

Questions?