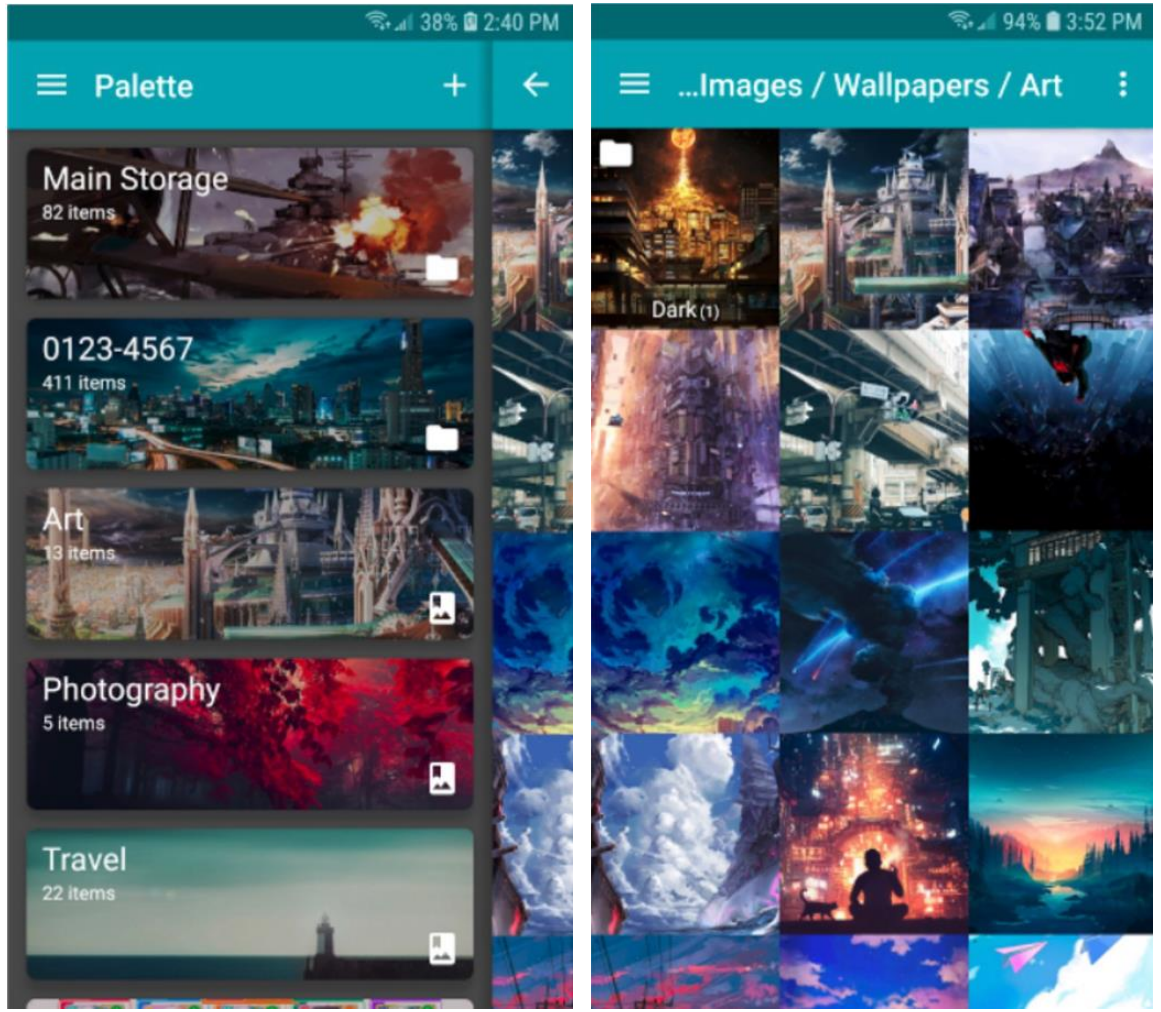
Android Basics

Jetpack Compose

University of Alberta CMPUT 301 (Fall 2026)

Activities host Compose screens

- **An Activity is an Android app entry point**
  MainActivity is usually a Kotlin class
- In a Compose app, ComponentActivity hosts the UI
- Android calls lifecycle methods as the Activity changes state



Compose UI uses Kotlin

- Compose does not require an XML layout file
- @Composable functions describe the UI in Kotlin

Parameters control what appears on screen

When state changes, Compose updates the affected UI

```
@Composable
fun JetpackCompose() {
    Card {
        var expanded by remember { mutableStateOf(false) }
        Column(Modifier.clickable { expanded = !expanded }) {
            Image(painterResource(R.drawable.jetpack_compose))
            AnimatedVisibility(expanded) {
                Text(
                    text = "Jetpack Compose",
                    style = MaterialTheme.typography.bodyLarge,
                )
            }
        }
    }
}
```

MainActivity starts the Compose UI

- MainActivity extends ComponentActivity
  - Android calls onCreate() when the Activity begins
    - setContent { … } starts the Compose UI
- Composable functions define screen behaviour and appearance

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            Text("Hello world!")
        }
    }
}
```

setContent connects the Activity to its Compose UI

Data flows through functions

- Composable functions receive data through parameters
  - User actions are reported through callback functions
  - State flows down; events flow up
  - Example: CityListScreen(cities)

```
CityListScreen(
    cities = cityRepository.cities,
    modifier = Modifier.padding( paddingValues = innerPadding)
```

Composable UI elements

- A composable is a reusable UI function
- Text displays text

  OutlinedTextField collects input
- Button responds through onClick
- Modifier controls size, padding and placement

```
@Composable
fun ExampleForm(
    name: String,
    onNameChange: (String) -> Unit,
    onSubmit: () -> Unit
) {
    Column(modifier = Modifier.padding( all = 16.dp)) {
        Text("Enter your name")

        OutlinedTextField(
            value = name,
            onValueChange = onNameChange,
            label = { Text("Name") }
        )

        Button(onClick = onSubmit) {
            Text("Submit")
        }
    }
}
```

Composable code

Compose layouts

- Layout composables arrange child composables
- Column places children vertically
  - Row places children horizontally
    - Box layers children
    - Scaffold provides a basic screen structure
    - Modifier controls each composable’s layout

State Changes the UI

```
@Composable
fun ShowMessageExample() {
    var showMessage by remember {
        mutableStateOf( value = false)
    }

    Column {
        Button(
            onClick = {
                showMessage = !showMessage
            }
        ) {
            Text("Toggle message")
        }

        if (showMessage) {
            Text("Hello!")
        }
    }
}
```

State: remember { mutableStateOf(...) }

```
        if (showMessage) {
            Text("Hello!")
        }
    }
```

Compose shows (or hides) the input fields

onCreate starts the Compose UI

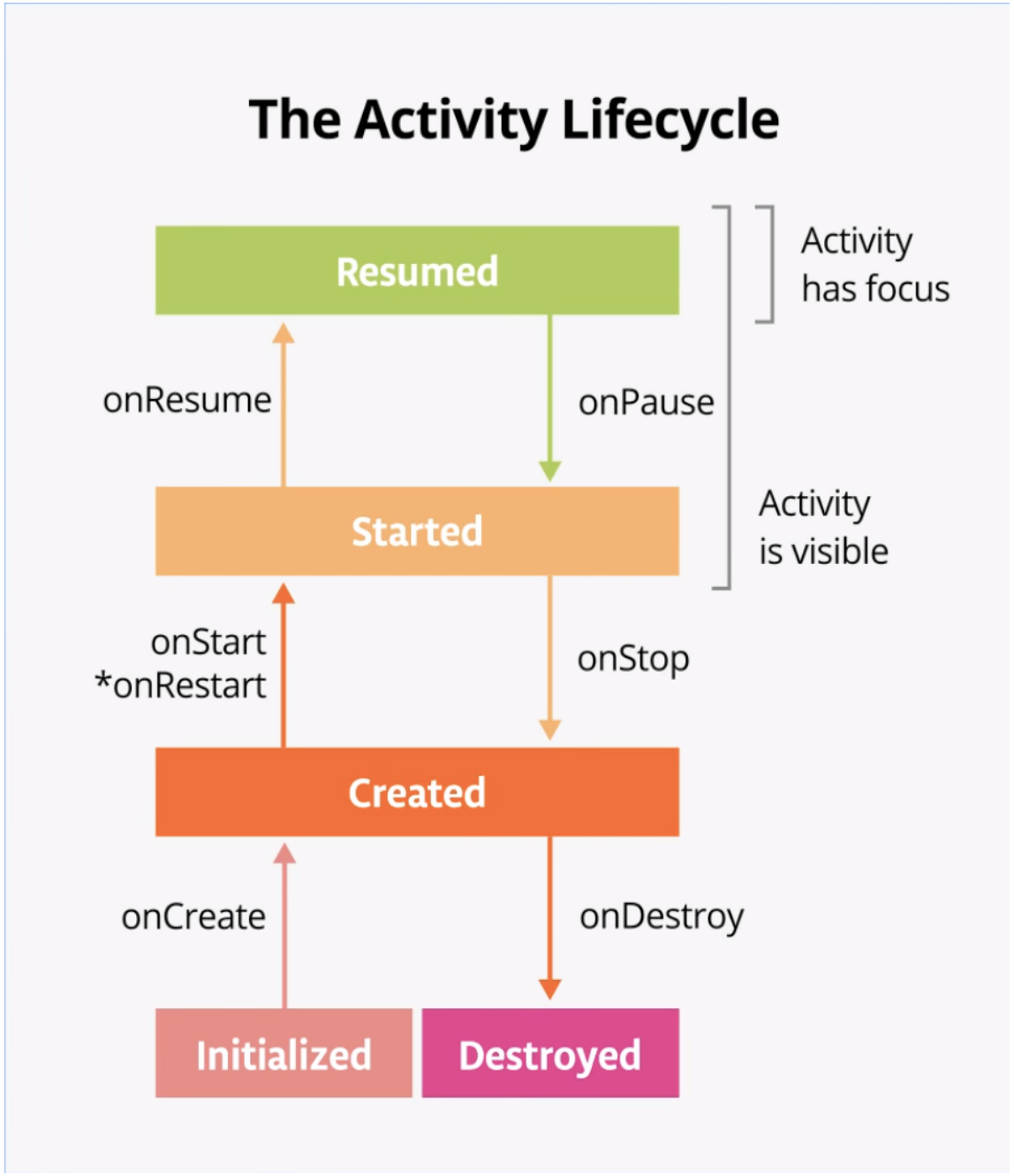
- **onCreate() is called when Android creates the Activity**
- In a Compose app, use it to:
  - call setContent { … }
  - apply the app theme
  - provide data to the first screen
- Think of onCreate() as the Activity's setup method

```
class MainActivity : ComponentActivity() {  1 Usage
    override fun onCreate(
        savedInstanceState: Bundle?
    ) {
        super.onCreate(savedInstanceState)

        setContent {
            MaterialTheme {
                Text("Hello, Android!")
            }
        }
    }
}
```

Activity lifecycle

- Android calls lifecycle methods as an Activity changes state
- onCreate() — set up the Activity and Compose content
  - onStart() — Activity becomes visible
  - onResume() — Activity becomes interactive
  - onPause()/onStop() — Activity leaves the foreground
  - onDestroy() — Activity is being destroyed
  - Compose does not replace the Activity lifecycle



Lists use LazyColumn

- **LazyColumn creates a vertically scrolling list**
  - items(names) creates one row for each city
- Each row is its own reusable composable
- Lazy lists compose visible items as needed
- This replaces ListView/ListAdapter in this lab

```
@Composable
fun NameList(names: List<String>) {
    LazyColumn {
        items(names) { name ->
            Text(name)
        }
    }
}
```

Good to Know for Assignment 1

- How to do something when a view is clicked
  - (Buttons, onClick, modifiers)
- Use of Intents to navigate to another activity
- How to send data between activities (send it through Intent)
- Android Manifest
- How to display a scrolling list of items (LazyColumn)

- These concepts will be covered in the lab
- **Intents and the Android manifest remain important Android concepts for Assignment 1**

References

Android Developers - https://developer.android.com/develop/ui/compose
- Jetpack Compose overview
- State and Jetpack Compose
- Lazy lists and lazy grids

The Activity lifecycle (https://developer.android.com/guide/components/activities/activity-lifecycle)

Assets
- Android icon - Freepik, flaticon.com
- Activity lifecycle diagram - Android Developers