Topic for discussion:

UI Test in Jetpack Compose

Presented by **Pankaj Dholakiya**
(Android Engineer)

What is UI Test?

- **UI test is the term used for any test that verifies the correct behavior of a UI.**
- **Testing the individual components that make up your Android application, it is also important that you test the behavior of your application's user interface (UI) when it is running on a device.**

Benefits of UI Testing

- **Tests the user interface from the users perspective.**
- **Offers developers and testers ease of use and learning.**
- **Helps validate the compliance of various icons and elements with their design specifications.**
- **Increases reliability and improves the quality of the product.**

Jetpack frameworks

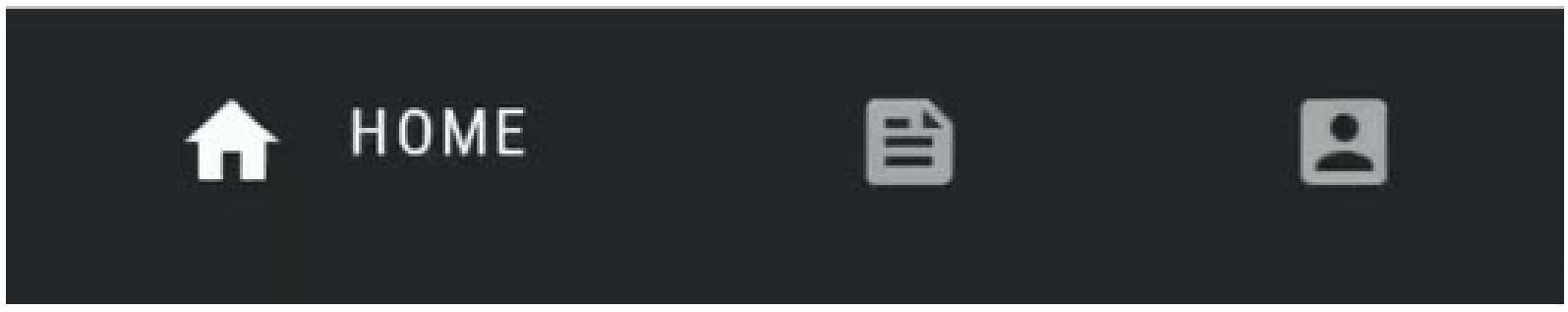
- **The Espresso testing framework (Android 4.0.1, API level 14 or higher)**
- **Jetpack Compose (Android 5.0, API level 21 or higher)**
- **UI Automator (Android 4.3, API level 18 or higher)**
- **Robolectric (Android 4.1, API level 16 or higher)**

Setup and Implementation

- **Add below dependency into build.gradle file (module level gradle file)**

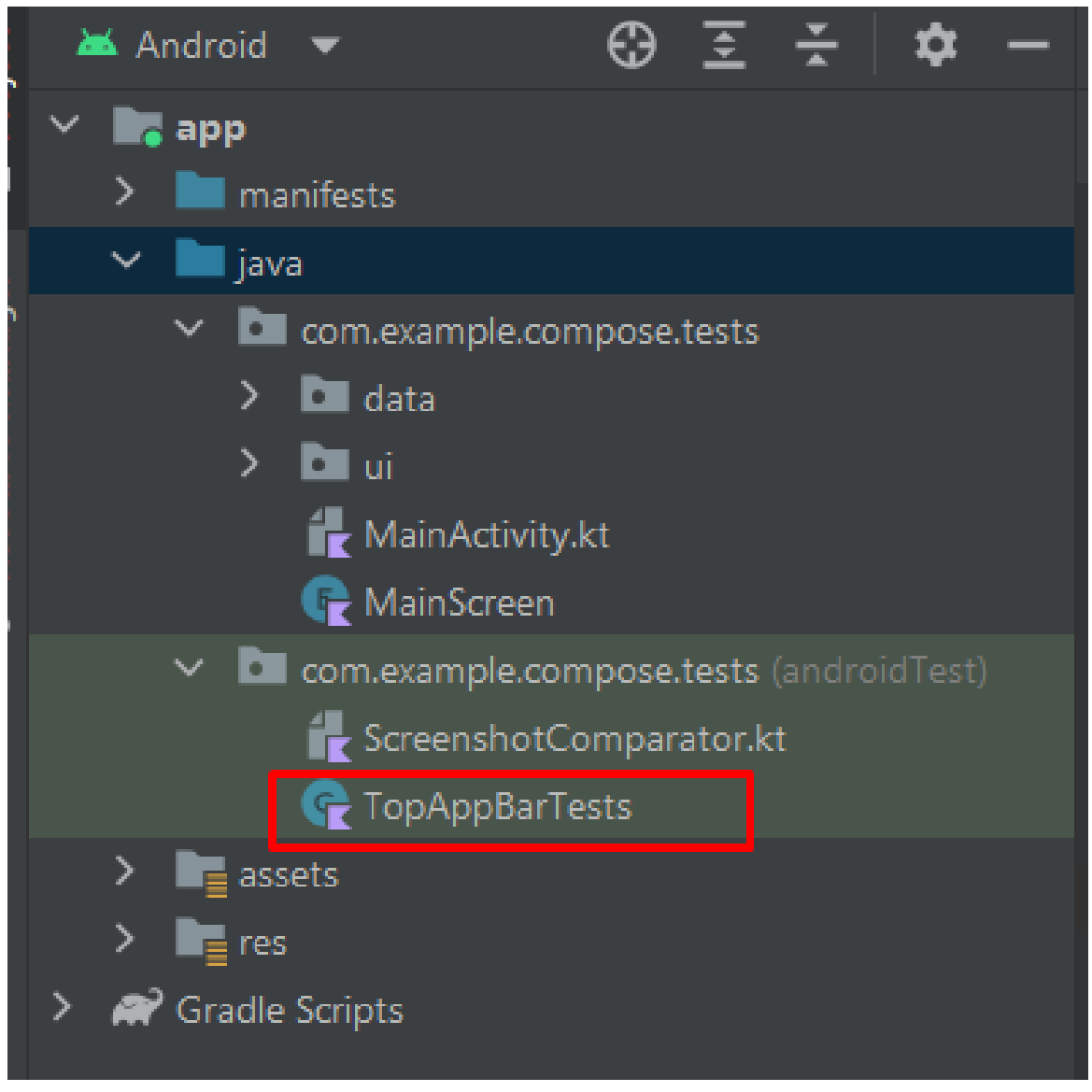
```
// Compose testing dependencies
androidTestImplementation "androidx.compose.ui:ui-test:1.2.0"
androidTestImplementation "androidx.compose.ui:ui-test-junit4:1.2.0"
debugImplementation "androidx.compose.ui:ui-test-manifest:1.2.0"
```

- **We're going to focus on tab bar, which contains a row of tabs (Home, feed, and Account).**



Create Simple UI Test

- **Create a new file in the androidTest package and named it as TopAppBarTests**



```
class TopAppBarTests {

    @get:Rule
    val composeTestRule = createComposeRule()

    // TODO: Add Tests

}
```

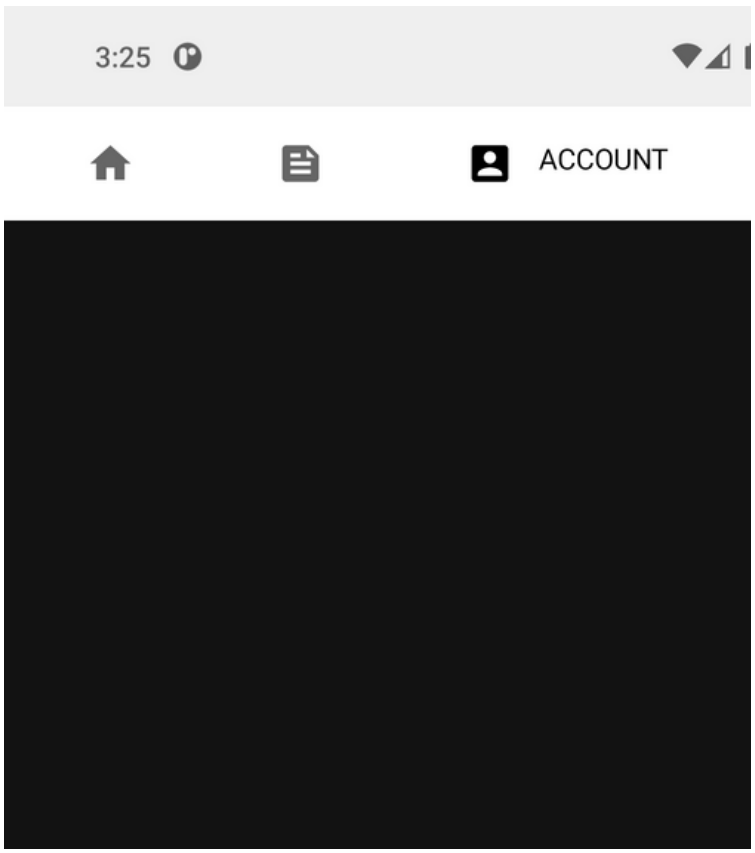
```
class TopAppBarTests {

    @get:Rule
    val composeTestRule = createComposeRule()

    @Test
    fun myTest() {
        composeTestRule.setContent {
            Text(text = "You can set any Compose content!")
        }
    }
```

- **MainScreenTopAppBar takes three parameters that are easy to provide so we can pass fake data that we control. For example:**

```
@Test
fun mainScreenTopAppBarTest(){
    val allScreens = MainScreen.values().toList()
    composeTestRule.setContent {
        MainScreenTopAppBar(
            allScreens = allScreens,
            onTabSelected = { MainScreen.Account },
            currentScreen = MainScreen.Account
        )
    }
    Thread.sleep( millis: 5000)
}
```

- **We also add a sleep() so you can see what's going on. Right-click on the mainScreenTopAppBarTest and click on "Run mainScreenTopAppBarTest ()".**

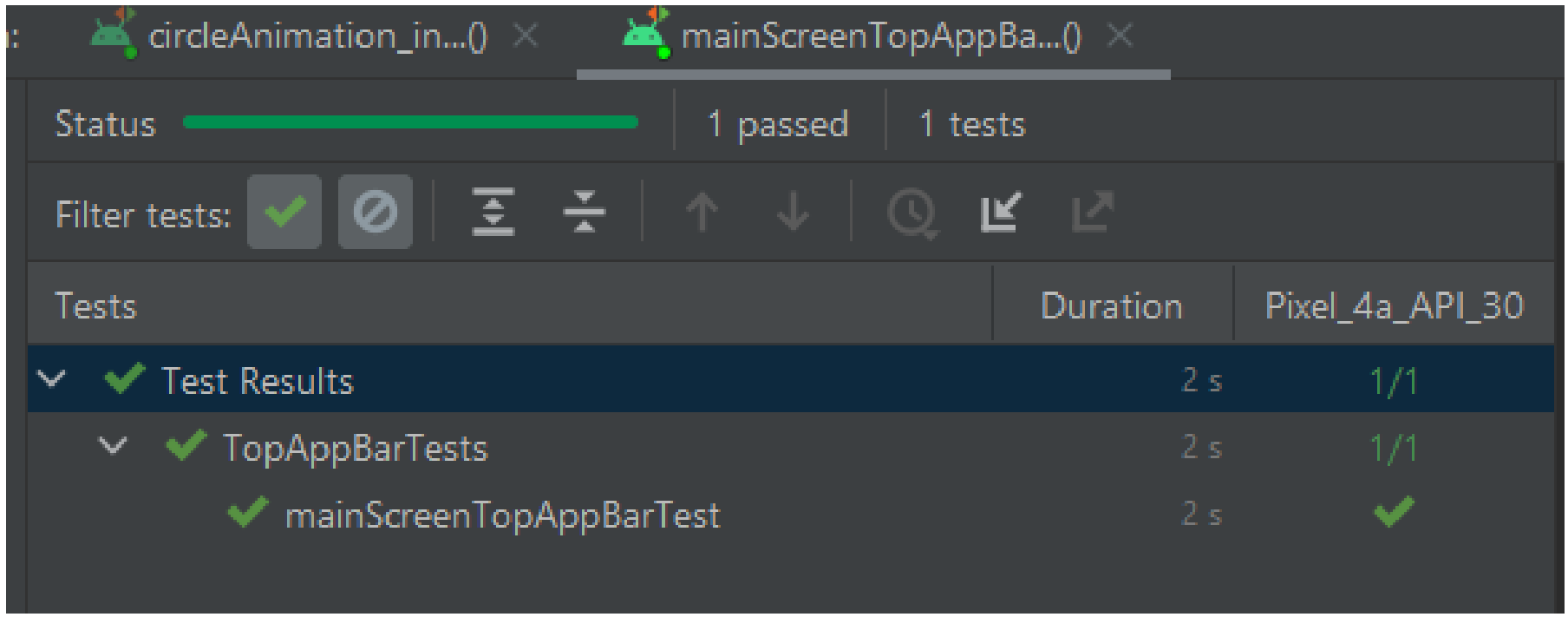
- **The test shows the top app bar for 5 seconds, but it doesn't look as we expected: it has a light theme!**




```
@Test
fun mainScreenTopAppBarTest(){
    val allScreens = MainScreen.values().toList()
    composeTestRule.setContent {
        MainTheme {
            MainScreenTopAppBar(
                allScreens = allScreens,
                onTabSelected = { MainScreen.Account },
                currentScreen = MainScreen.Account
            )
        }
    }
    Thread.sleep( millis: 5000)
}
```

- **In our example, each tab has a content description.**
  - **Home**
  - **Feed**
  - **Account**
- **So, we will replace Thread.sleep(5000) with a statement that looks for a content description and asserts that it exists:**

```
@Test
fun mainScreenTopAppBarTest(){
    val allScreens = MainScreen.values().toList()
    composeTestRule.setContent {
        MainTheme {
            MainScreenTopAppBar(
                allScreens = allScreens,
                onTabSelected = { MainScreen.Account },
                currentScreen = MainScreen.Account
            )
        }
    }

    composeTestRule.onNodeWithContentDescription(
        MainScreen.Account.name
    ).assertIsSelected()
}
```


circleAnimation_in...()
mainScreenTopAppBa...()
Status
1 passed
1 tests
Filter tests:
Tests
Duration
Pixel_4a_API_30
Test Results
2 s
1/1
TopAppBarTests
2 s
1/1
mainScreenTopAppBarTest
2 s

it Agenturen