Behaviour Driven Development (BDD) using Cucumber for Android

Here’s a complete example of setting up **BDD with Cucumber in Kotlin** for an Android project with a **login screen**. This example will show you how to integrate **Cucumber** with Kotlin and test the UI using **Espresso**.

**1. Add Dependencies**

In your project’s build.gradle (app-level), add the necessary dependencies for Cucumber and Espresso.

gradle

Copy code

dependencies {

// Cucumber dependencies for Android

androidTestImplementation 'info.cukes:cucumber-android:1.2.5'

androidTestImplementation 'info.cukes:cucumber-picocontainer:1.2.5'

androidTestImplementation 'info.cukes:cucumber-junit:1.2.5'

// Espresso for UI testing

androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

androidTestImplementation 'androidx.test.ext:junit:1.1.3'

}

**2. Gherkin Feature File**

Create your feature file in the src/androidTest/assets/features/ directory. The feature file describes the login scenario in Gherkin syntax.

Create a file named login.feature:

gherkin

Copy code

Feature: User login

Scenario: User successfully logs in with valid credentials

Given the user is on the login screen

When the user enters valid username and password

And the user taps the login button

Then the user should see the home screen

**3. Step Definitions in Kotlin**

Now create the step definitions that link the Gherkin steps to actual Android UI interactions. These steps are written in Kotlin.

Create a file named LoginStepDefinitions.kt under src/androidTest/kotlin/your/package/name.

kotlin

Copy code

package your.package.name

import androidx.test.espresso.Espresso.onView

import androidx.test.espresso.action.ViewActions

import androidx.test.espresso.matcher.ViewMatchers.withId

import androidx.test.espresso.matcher.ViewMatchers.withText

import cucumber.api.java.en.Given

import cucumber.api.java.en.When

import cucumber.api.java.en.Then

import androidx.test.espresso.assertion.ViewAssertions.matches

import androidx.test.espresso.matcher.ViewMatchers.isDisplayed

import your.package.name.R

class LoginStepDefinitions {

@Given("^the user is on the login screen$")

fun userIsOnLoginScreen() {

// In real scenarios, you may launch the login activity if needed

// Here, Espresso is used to check if login screen is visible

onView(withId(R.id.login\_button)).check(matches(isDisplayed()))

}

@When("^the user enters valid username and password$")

fun userEntersValidCredentials() {

// Using Espresso to input the username and password

onView(withId(R.id.username)).perform(ViewActions.typeText("testUser"))

onView(withId(R.id.password)).perform(ViewActions.typeText("password123"))

}

@When("^the user taps the login button$")

fun userTapsLoginButton() {

// Using Espresso to click the login button

onView(withId(R.id.login\_button)).perform(ViewActions.click())

}

@Then("^the user should see the home screen$")

fun userSeesHomeScreen() {

// Check if the home screen is displayed after successful login

onView(withId(R.id.home\_screen)).check(matches(isDisplayed()))

}

}

**4. Login Activity**

You will need a simple Login screen layout and Activity in your Android app. Here’s an example of a **LoginActivity** and its layout.

**LoginActivity.kt**

kotlin

Copy code

package your.package.name

import android.content.Intent

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

import android.widget.Button

import android.widget.EditText

class LoginActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_login)

val loginButton: Button = findViewById(R.id.login\_button)

val usernameField: EditText = findViewById(R.id.username)

val passwordField: EditText = findViewById(R.id.password)

loginButton.setOnClickListener {

val username = usernameField.text.toString()

val password = passwordField.text.toString()

if (username == "testUser" && password == "password123") {

// Navigate to home screen if login is successful

val intent = Intent(this, HomeActivity::class.java)

startActivity(intent)

}

}

}

}

**activity\_login.xml**

Here’s a simple layout for the Login screen with EditText for username and password and a Button for login.

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".LoginActivity">

<EditText

android:id="@+id/username"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:hint="Username"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintWidth\_percent="0.8"

android:layout\_marginTop="150dp"/>

<EditText

android:id="@+id/password"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:hint="Password"

android:inputType="textPassword"

app:layout\_constraintTop\_toBottomOf="@id/username"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintWidth\_percent="0.8"

android:layout\_marginTop="20dp"/>

<Button

android:id="@+id/login\_button"

android:layout\_width="0dp"

android:layout\_height="wrap\_content"

android:text="Login"

app:layout\_constraintTop\_toBottomOf="@id/password"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintWidth\_percent="0.8"

android:layout\_marginTop="20dp"/>

</androidx.constraintlayout.widget.ConstraintLayout>

**5. Home Activity**

Here’s a simple HomeActivity that will be shown after a successful login.

**HomeActivity.kt**

kotlin

Copy code

package your.package.name

import android.os.Bundle

import androidx.appcompat.app.AppCompatActivity

class HomeActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_home)

}

}

**activity\_home.xml**

xml

Copy code

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:id="@+id/home\_screen">

<TextView

android:id="@+id/home\_text"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Welcome to the Home Screen"

android:textSize="18sp"

android:layout\_gravity="center"

android:layout\_marginTop="200dp"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toEndOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**6. Cucumber Test Runner**

Create a CucumberTestRunner to link Cucumber with Android's JUnit.

Create a file named CucumberTestRunner.kt under src/androidTest/kotlin/your/package/name:

kotlin

Copy code

package your.package.name

import cucumber.api.CucumberOptions

import cucumber.api.junit.Cucumber

import org.junit.runner.RunWith

@RunWith(Cucumber::class)

@CucumberOptions(

features = "features", // Path to the feature file

glue = ["your.package.name"] // Package containing the step definitions

)

class CucumberTestRunner

**7. Running the Tests**

To run the tests, either:

* Right-click on the CucumberTestRunner class and select **Run 'CucumberTestRunner'**.
* Use the command-line:

bash

Copy code

./gradlew connectedAndroidTest

**Summary**

* **Gherkin feature file** describes the login scenario.
* **Step definitions** are written in Kotlin, where Espresso is used to interact with UI components.
* **LoginActivity** and **HomeActivity** handle the actual UI flow of the login screen.
* **CucumberTestRunner** runs the BDD tests.

This example provides a full BDD flow for an Android app using Kotlin, Cucumber, and Espresso.