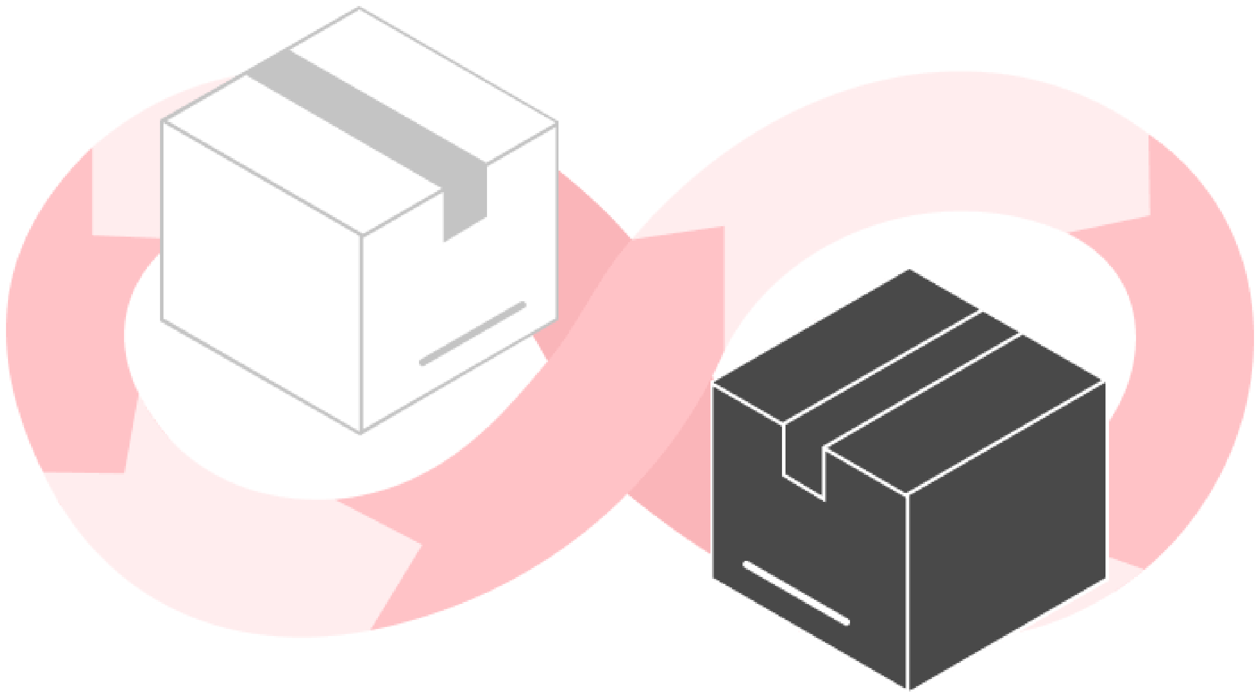
Lab 7 CMPUT 301

Android UI Testing

Espresso

Different approach to Software Testing



Black Box Testing	White Box Testing
Internal structure/implementation of the item being tested is not known to the tester. Only the external design and structure are tested.	Internal structure/implementation of the item being tested is known to the tester. Implementation and impact of the code are tested.

Instrumented Tests

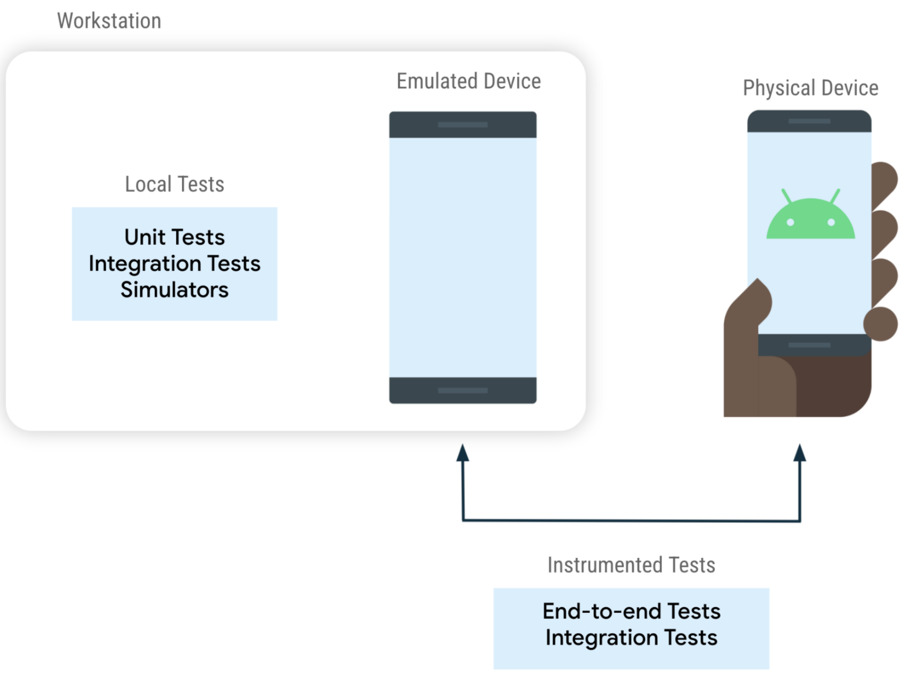
Instrumented tests run on an Android device, either **physical** or **emulated**. The app is built and installed alongside a test app that **injects commands** and **reads the state**. Instrumented tests are usually **UI tests**, launching an app and then interacting with it.

Some libraries that can be used for Intent testing

1. *Espresso*
   https://developer.android.com/training/testing/espresso

2. *Robotium*
   https://github.com/RobotiumTech/robotium

3. *UI Automator*
   https://developer.android.com/training/testing/other-components/ui-automator



Lab 7 Instructions
Espresso

---

1. Clone the repo https://github.com/iAniket23/Android-UI-testing.git

2. Make sure these **dependencies** are present in the **app Gradle (build.gradle(Module:app)) file**. If it is not present in the build.gradle file then add these dependencies and **sync** the build.gradle file..

```java
testImplementation("junit:junit:4.13.2")
androidTestImplementation("androidx.test.ext:junit:1.1.5")
androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
```
`

3. We want to create UI tests for our **MainActivity** class as it is our UI class

4. Under **com.example.androiduitesting(androidTest)** folder, create a new class '*MainActivityTest*' to test the UI functionalities.

5. Let's add some **decorators**
   The AndroidJUnitRunner class is a JUnit test runner that lets you run instrumented JUnit 4 tests on Android devices, including those using the Espresso, UI Automator, and Compose testing frameworks.
   The test runner handles loading your test package and the app under test to a device, running your tests, and reporting test results.

```java
@RunWith(AndroidJUnit4.class)
@LargeTest
public class MainActivityTest {

}
```

6. Now in order to test MainActivity we need to make a ‘**Rule**’
   This rule provides functional testing of a single activity. The rule launches the chosen activity before each test annotated with @Test.
   ActivityScenario provides APIs to start and drive an Activity's lifecycle state for testing.

```java
@Rule
public ActivityScenarioRule<MainActivity> scenario = new
ActivityScenarioRule<MainActivity>(MainActivity.class);
```

7. Now we are set to **write our first Test**
   We will click on the Add City Button and enter the name of the city and then we will check if that city is displayed. If the city is displayed then the test should be pass else it should fail

```java
 @Test
  public void testAddCity(){
    // Click on Add City button
    onView(withId(R.id.button_add)).perform(click());

    // Type "Edmonton" in the editText

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));

    // Click on Confirm
    onView(withId(R.id.button_confirm)).perform(click());

    // Check if text "Edmonton" is matched with any of the text
displayed on the screen
    onView(withText("Edmonton")).check(matches(isDisplayed()));
  }
```

8. Now we will **write our second test** -
   We will add cities and then click on Clear All button, if the cities displayed are removed then test should pass else it should fail

```java
@Test
  public void testClearCity(){
    // Add first city to the list
    onView(withId(R.id.button_add)).perform(click());

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));
    onView(withId(R.id.button_confirm)).perform(click());

    //Add another city to the list
    onView(withId(R.id.button_add)).perform(click());

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Vancouv
er"));
    onView(withId(R.id.button_confirm)).perform(click());

    //Clear the list
    onView(withId(R.id.button_clear)).perform(click());
    onView(withText("Edmonton")).check(doesNotExist());
    onView(withText("Vancouver")).check(doesNotExist());
  }
```

9. **Test number three** -
   We see if the city “Edmonton” that we add is located at position 0 of the list

```java
@Test
  public void testListView(){
    // Add a city
    onView(withId(R.id.button_add)).perform(click());

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));
    onView(withId(R.id.button_confirm)).perform(click());

    // Check if in the Adapter view (given id of that adapter view),
there is a data
    // (which is an instance of String) located at position zero.
    // If this data matches the text we provided then Voila! Our test
should pass
    // You can also use anything() in place of
is(instanceOf(String.class))

onData(is(instanceOf(String.class))).inAdapterView(withId(R.id.city_list
)).atPosition(0).check(matches((withText("Edmonton"))));

  }
```

Important Notes
Make sure you click (alt + enter) when there is a red line error under any unrecognized term in order to import it.
These were the **imports** for this Lab demo

```java
package com.example.androiduitesting;

import static androidx.test.espresso.Espresso.onData;
import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.click;
import static
androidx.test.espresso.assertion.ViewAssertions.doesNotExist;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.isDisplayed;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;

import static org.hamcrest.CoreMatchers.anything;
import static org.hamcrest.CoreMatchers.instanceOf;
import static org.hamcrest.CoreMatchers.is;

import androidx.test.core.app.ActivityScenario;
import androidx.test.espresso.action.ViewActions;
import androidx.test.ext.junit.rules.ActivityScenarioRule;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.filters.LargeTest;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
```

Complete code for MainActivityTest file

```java
package com.example.androiduitesting;

import static androidx.test.espresso.Espresso.onData;
import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.click;
import static
androidx.test.espresso.assertion.ViewAssertions.doesNotExist;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.isDisplayed;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;

import static org.hamcrest.CoreMatchers.anything;
import static org.hamcrest.CoreMatchers.instanceOf;
import static org.hamcrest.CoreMatchers.is;

import androidx.test.core.app.ActivityScenario;
import androidx.test.espresso.action.ViewActions;
import androidx.test.ext.junit.rules.ActivityScenarioRule;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.filters.LargeTest;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

@RunWith(AndroidJUnit4.class)
@LargeTest
public class MainActivityTest {
  @Rule
  public ActivityScenarioRule<MainActivity> scenario = new
ActivityScenarioRule<MainActivity>(MainActivity.class);

  @Test
  public void testAddCity(){
    // Click on Add City button
    onView(withId(R.id.button_add)).perform(click());
    // Type "Edmonton" in the editText

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));

    // Click on Confirm
    onView(withId(R.id.button_confirm)).perform(click());

    // Check if text "Edmonton" is matched with any of the text
displayed on the screen
    onView(withText("Edmonton")).check(matches(isDisplayed()));
  }

  @Test
  public void testClearCity(){
    // Add first city to the list
    onView(withId(R.id.button_add)).perform(click());

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));
    onView(withId(R.id.button_confirm)).perform(click());

    //Add another city to the list
    onView(withId(R.id.button_add)).perform(click());

onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Vancouv
er"));
    onView(withId(R.id.button_confirm)).perform(click());

    //Clear the list
    onView(withId(R.id.button_clear)).perform(click());
    onView(withText("Edmonton")).check(doesNotExist());
    onView(withText("Vancouver")).check(doesNotExist());
  }

  @Test
  public void testListView(){
    // Add a city
    onView(withId(R.id.button_add)).perform(click());
onView(withId(R.id.editText_name)).perform(ViewActions.typeText("Edmonto
n"));
    onView(withId(R.id.button_confirm)).perform(click());

    // Check if in the Adapter view (given id of that adapter view),
there is a data
    // (which is an instance of String) located at position zero.
    // If this data matches the text we provided then Voila! Our test
should pass
    // You can also use anything() in place of
is(instanceOf(String.class))

onData(is(instanceOf(String.class))).inAdapterView(withId(R.id.city_list
)).atPosition(0).check(matches((withText("Edmonton"))));

  }

}
```