LE/EECS 4443: Mobile User Interfaces (LAB)

Week 6: Introduction to Android Studio Testing

S.Toyonaga[1]

[1]Lassonde School of Engineering
York University

February 16, 2024

Table of Contents

Introduction

By the end of this tutorial, you will be able to...

1. Understand the fundamentals of software testing
2. Determine when to perform software or usability testing
3. Write Espresso (Usability) tests using Android Studio
4. Stress-test your application

Why Test?

- Software testing is a way to enforce **quality assurance** and communicate a tangible agreement between clients and the developer.
- Testing outlines a a set of premises (behaviours) and antecedents (expectations).
- **Note:** While we can do our best to eliminate bugs and optimize efficiency, it is unrealistic to expect every bug to be caught at once.

Types of Testing

1. **Big Bang (Post-Hoc Analysis of Application)**
2. **Incremental (Iterative)**
   - Unit Testing (Components)
   - Integration (Groups of Components, Dependencies)
   - System ($\forall$)
3. **Black $\vee$ White Box Testing**
   - Selection depends upon the design of test-cases and transparency of the system (code, documentation, architecture, etc.,)
4. **Alpha $\vee$ Beta Testing**
   - Alpha $\rightarrow$ Very Incomplete
   - Beta $\rightarrow$ Functional Prototype

Classes of Tests (Equivalence)

- There are three types of test classes:
  1. Happy
  2. Boundary
  3. Exceptional
- A test case can cover a wide-range of happy and/or boundary classes (many-to-one)
- There can only be a one-to-one relationship between test cases that cover exceptional paths (one-to-one).

Which One Should I Choose!?

Remark

"Oak's words echoed... There's a time and place for everything, but not now."

- **Summary:** The approach that you use to test your application will always depend on the scale of your application and the process of your design.
- When choosing a testing paradigm, you must also be realistic with your cost and time budgets.

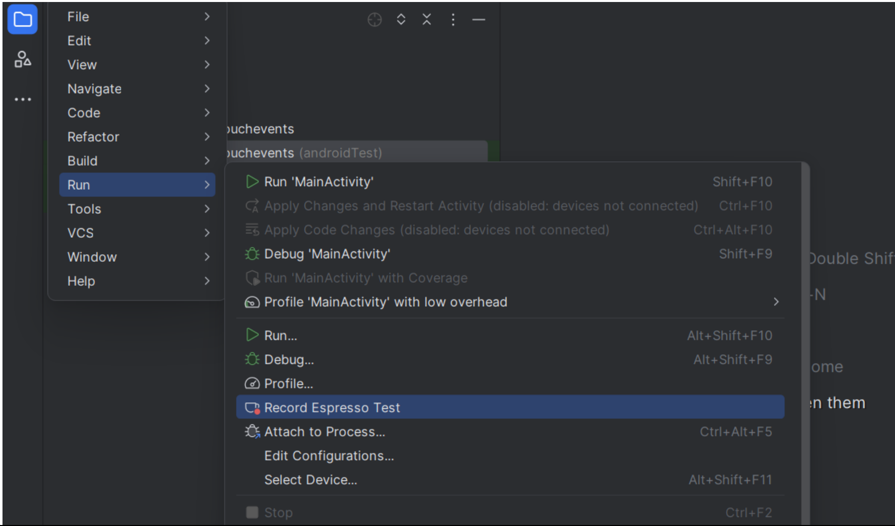
1. **Software Testing (LE/EECS 3311)**
   - Validates the behaviour of a function, given data (i.e., Model).
2. **UI Testing (LE/EECS 4443)**
   - Validate the state of the User Interface before and/or after an interaction has occurred (i.e., View-Controller)

Jetpack Frameworks

1. **Espresso Testing Framework:** "Provides APIs for writing UI tests to simulate user interactions with Views **within a single target app**."
2. **UI AUtomator:** "A UI testing framework suitable for **cross-app** functional UI testing across system and installed apps."

Source: https://developer.android.com/training/testing/instrumented-tests/ui-tests

Espresso Record & Replay


File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
Window
Help
uchevents
uchevents (androidTest)
Run 'MainActivity' Shift+F10
Apply Changes and Restart Activity (disabled: devices not connected) Ctrl+F10
Apply Code Changes (disabled: devices not connected) Ctrl+Alt+F10
Debug 'MainActivity' Shift+F9
Run 'MainActivity' with Coverage
Profile 'MainActivity' with low overhead
Run... Alt+Shift+F10
Debug... Alt+Shift+F9
Profile...
Record Espresso Test
Attach to Process... Ctrl+Alt+F5
Edit Configurations...
Select Device... Alt+Shift+F11
Stop Ctrl+F2
Double Shif
-N
ome
n them

Espresso Code

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

    @Rule
    public ActivityScenarioRule<MainActivity> mActivityScer
            new ActivityScenarioRule<>(MainActivity.class)

    @Test
    public void mainActivityTest() {
            // Your test-case goes here!
    }
}
```

UI Testing Resources

1. Espresso Cheat Sheet
2. Instrumented UI Testing Documentation
3. UI Automator
4. **Demo:** TypingTest

Note: It is highly recommended that you play around with the demo! From there, you can implement your own tests in previous labs to validate your understanding of using Espresso :)

Conclusion

Remark

Thank you for your time!
Do you have any questions? :)