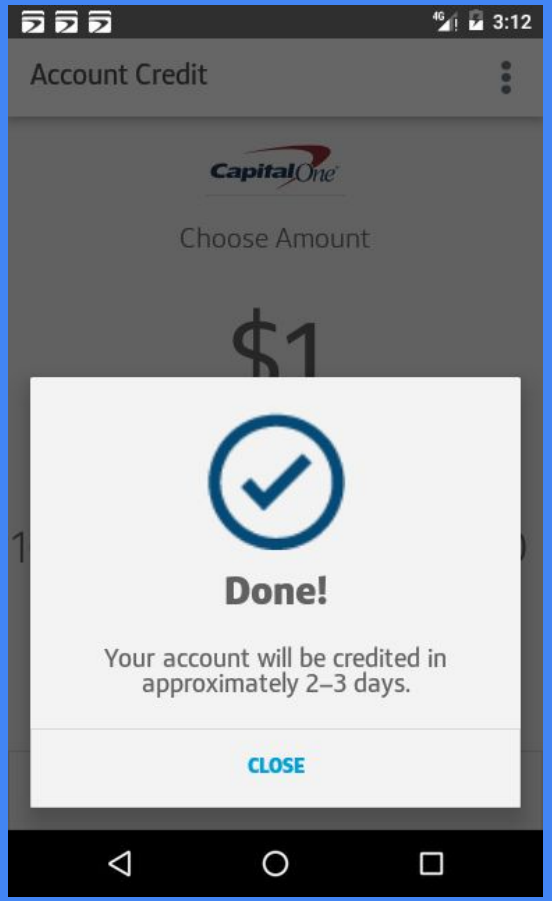
UI TESTING FOR ANDROID

espresso:

A Screenshot is Worth 1,000 Words

Sam Edwards - @HandstandSam - Capital One



Sam Edwards

@HandstandSam

http://handstandsam.com

- Android since 2011
- Architected and leading Espresso efforts for Capital One Wallet
- Opinions in this talk are my own, not Capital One

Capital One Wallet

wallet

••••••••5

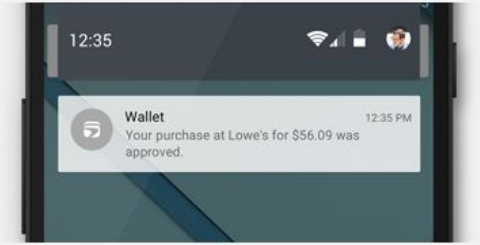
Remember me

SIGN IN

Need Help Signing In?

CAPITAL ONE 360 PIN SIGN IN

ENABLE NOTIFICATIONS?



Notifications are key to the Wallet.

We'll keep you informed of your purchases, track your rewards programs, and alert you to any potentially suspicious activity. You can adjust your notifications anytime by going to the settings menu.

NO THANKS

ENABLE

Visa Credit *3456

Card is On

My Rewards

EMBOSSED NAME

VISA...3456

Current Balance: $2,000.00

Pay My Bill

RECENT ACTIVITY

Last 7 Days

0 Credit Card Transactions

$0.00

Total Spent

TUESDAY, JULY, 7

Cards

Audience POLL:

- Testing?
- Espresso?
- Screenshots?
- Espresso with CI?

Outline

- Espresso in 2 minutes
- Screenshots - Why? How? When? The Cost?
- Maintainable Test Architecture
- Tips and Tricks

Espresso in 2 minutes

What is Espresso?

- A library provided by Google for Android UI testing
- Built on Android Test Instrumentation
- Aware of the idle state and therefore FAST
- Easy to read and write

Espresso Basics

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Actions

CLICK/PRESS
- click()
- doubleClick()
- longClick()
- pressBack()
- pressIMEActionButton()
- pressKey([int/EspressoKey])
- pressMenuKey()
- closeSoftKeyboard()
- openLink()

GESTURES
- scrollTo()
- swipeLeft()
- swipeRight()
- swipeUp()
- swipeDown()

TEXT
- clearText()
- typeText(String)
- typeTextIntoFocusedView(String)
- replaceText(String)

View Assertions

- matches(Matcher)
- doesNotExist()
- selectedDescendantsMatch(...)

LAYOUT ASSERTIONS
- noEllipseizedText(Matcher)
- noMultilineButtons()
- noOverlaps([Matcher])

POSITION ASSERTIONS
- isLeftOf(Matcher)
- isRightOf(Matcher)
- isLeftAlignedWith(Matcher)
- isRightAlignedWith(Matcher)
- isAbove(Matcher)
- isBelow(Matcher)
- isBottomAlignedWith(Matcher)
- isTopAlignedWith(Matcher)

View Matchers

USER PROPERTIES
- withId(...)
- withText(...)
- withTagKey(...)
- withTagValue(...)
- hasContentDescription(...)
- withContentDescription(...)
- withHint(...)
- withSpinnerText(...)
- hasLinks()
- hasEllipsizedText()
- hasMultilineTest()

UI PROPERTIES
- isDisplayed()
- isCompletelyDisplayed()
- isEnabled()
- hasFocus()
- isClickable()
- isChecked()
- isNotChecked()
- withEffectiveVisibility(...)
- isSelected()

OBJECT MATCHER
- allOf(Matchers)
- anyOf(Matchers)
- is(...)
- not(...)
- endsWith(String)
- startsWith(String)
- instanceOf(Class)

HIERARCHY
- withParent(Matcher)
- withChild(Matcher)
- hasDescendant(Matcher)
- isDescendantOfA(Matcher)
- hasSibling(Matcher)
- isRoot()

INPUT
- supportsInputMethods(...)
- hasIMEAction(...)

CLASS
- isAssignableFrom(...)
- withClassName(...)

ROOT MATCHERS
- isFocusable()
- isTouchable()
- isDialog()
- withDecorView()
- isPlatformPopup()

SEE ALSO
- Preference matchers
- Cursor matchers
- Layout matchers

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Espresso Matchers, Actions and Assertions

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Actions

CLICK/PRESS
- click()
- doubleClick()
- longClick()
- pressBack()
- pressIMEActionButton()
- pressKey([int/EspressoKey])
- pressMenuKey()
- closeSoftKeyboard()
- openLink()

GESTURES
- scrollTo()
- swipeLeft()
- swipeRight()
- swipeUp()
- swipeDown()

TEXT
- clearText()
- typeText(String)
- typeTextIntoFocusedView(String)
- replaceText(String)

View Assertions

- matches(Matcher)
- doesNotExist()
- selectedDescendantsMatch(...)

LAYOUT ASSERTIONS
- noEllipseizedText(Matcher)
- noMultilineButtons()
- noOverlaps([Matcher])

POSITION ASSERTIONS
- isLeftOf(Matcher)
- isRightOf(Matcher)
- isLeftAlignedWith(Matcher)
- isRightAlignedWith(Matcher)
- isAbove(Matcher)
- isBelow(Matcher)
- isBottomAlignedWith(Matcher)
- isTopAlignedWith(Matcher)

View Matchers

USER PROPERTIES
- withId(...)
- withText(...)
- withTagKey(...)
- withTagValue(...)
- hasContentDescription(...)
- withContentDescription(...)
- withHint(...)
- withSpinnerText(...)
- hasLinks()
- hasEllipsizedText()
- hasMultilineTest()

UI PROPERTIES
- isDisplayed()
- isCompletelyDisplayed()
- isEnabled()
- hasFocus()
- isClickable()
- isChecked()
- isNotChecked()
- withEffectiveVisibility(...)
- isSelected()

OBJECT MATCHER
- allOf(Matchers)
- anyOf(Matchers)
- is(...)
- not(...)
- endsWith(String)
- startsWith(String)
- instanceOf(Class)

HIERARCHY
- withParent(Matcher)
- withChild(Matcher)
- hasDescendant(Matcher)
- isDescendantOfA(Matcher)
- hasSibling(Matcher)
- isRoot()

INPUT
- supportsInputMethods(...)
- hasIMEAction(...)

CLASS
- isAssignableFrom(...)
- withClassName(...)

ROOT MATCHERS
- isFocusable()
- isTouchable()
- isDialog()
- withDecorView()
- isPlatformPopup()

SEE ALSO
- Preference matchers
- Cursor matchers
- Layout matchers

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Espresso Matchers, Actions and Assertions

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Actions

CLICK/PRESS
- click()
- doubleClick()
- longClick()
- pressBack()
- pressIMEActionButton()
- pressKey([int/EspressoKey])
- pressMenuKey()
- closeSoftKeyboard()
- openLink()

GESTURES
- scrollTo()
- swipeLeft()
- swipeRight()
- swipeUp()
- swipeDown()

TEXT
- clearText()
- typeText(String)
- typeTextIntoFocusedView(String)
- replaceText(String)

View Assertions

- matches(Matcher)
- doesNotExist()
- selectedDescendantsMatch(...)

LAYOUT ASSERTIONS
- noEllipseizedText(Matcher)
- noMultilineButtons()
- noOverlaps([Matcher])

POSITION ASSERTIONS
- isLeftOf(Matcher)
- isRightOf(Matcher)
- isLeftAlignedWith(Matcher)
- isRightAlignedWith(Matcher)
- isAbove(Matcher)
- isBelow(Matcher)
- isBottomAlignedWith(Matcher)
- isTopAlignedWith(Matcher)

View Matchers

USER PROPERTIES
- withId(...)
- withText(...)
- withTagKey(...)
- withTagValue(...)
- hasContentDescription(...)
- withContentDescription(...)
- withHint(...)
- withSpinnerText(...)
- hasLinks()
- hasEllipsizedText()
- hasMultilineTest()

UI PROPERTIES
- isDisplayed()
- isCompletelyDisplayed()
- isEnabled()
- hasFocus()
- isClickable()
- isChecked()
- isNotChecked()
- withEffectiveVisibility(...)
- isSelected()

OBJECT MATCHER
- allOf(Matchers)
- anyOf(Matchers)
- is(...)
- not(...)
- endsWith(String)
- startsWith(String)
- instanceOf(Class)

HIERARCHY
- withParent(Matcher)
- withChild(Matcher)
- hasDescendant(Matcher)
- isDescendantOfA(Matcher)
- hasSibling(Matcher)
- isRoot()

INPUT
- supportsInputMethods(...)
- hasIMEAction(...)

CLASS
- isAssignableFrom(...)
- withClassName(...)

ROOT MATCHERS
- isFocusable()
- isTouchable()
- isDialog()
- withDecorView()
- isPlatformPopup()

SEE ALSO
- Preference matchers
- Cursor matchers
- Layout matchers

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Espresso Matchers, Actions and Assertions

```
onView(ViewMatcher)
    .perform(ViewAction)
    .check(ViewAssertion);
```

View Actions

CLICK/PRESS
- click()
- doubleClick()
- longClick()
- pressBack()
- pressIMEActionButton()
- pressKey([int/EspressoKey])
- pressMenuKey()
- closeSoftKeyboard()
- openLink()

GESTURES
- scrollTo()
- swipeLeft()
- swipeRight()
- swipeUp()
- swipeDown()

TEXT
- clearText()
- typeText(String)
- typeTextIntoFocusedView(String)
- replaceText(String)

View Assertions

- matches(Matcher)
- doesNotExist()
- selectedDescendantsMatch(...)

LAYOUT ASSERTIONS
- noEllipseizedText(Matcher)
- noMultilineButtons()
- noOverlaps([Matcher])

POSITION ASSERTIONS
- isLeftOf(Matcher)
- isRightOf(Matcher)
- isLeftAlignedWith(Matcher)
- isRightAlignedWith(Matcher)
- isAbove(Matcher)
- isBelow(Matcher)
- isBottomAlignedWith(Matcher)
- isTopAlignedWith(Matcher)

View Matchers

USER PROPERTIES
- withId(...)
- withText(...)
- withTagKey(...)
- withTagValue(...)
- hasContentDescription(...)
- withContentDescription(...)
- withHint(...)
- withSpinnerText(...)
- hasLinks()
- hasEllipsizedText()
- hasMultilineTest()

UI PROPERTIES
- isDisplayed()
- isCompletelyDisplayed()
- isEnabled()
- hasFocus()
- isClickable()
- isChecked()
- isNotChecked()
- withEffectiveVisibility(...)
- isSelected()

OBJECT MATCHER
- allOf(Matchers)
- anyOf(Matchers)
- is(...)
- not(...)
- endsWith(String)
- startsWith(String)
- instanceOf(Class)

HIERARCHY
- withParent(Matcher)
- withChild(Matcher)
- hasDescendant(Matcher)
- isDescendantOfA(Matcher)
- hasSibling(Matcher)
- isRoot()

INPUT
- supportsInputMethods(...)
- hasIMEAction(...)

CLASS
- isAssignableFrom(...)
- withClassName(...)

ROOT MATCHERS
- isFocusable()
- isTouchable()
- isDialog()
- withDecorView()
- isPlatformPopup()

SEE ALSO
- Preference matchers
- Cursor matchers
- Layout matchers

https://google.github.io/android-testing-support-library/docs/espresso/cheatsheet/

Espresso Example

```
@Test
public void testLogin() {
   onView(withText("LOGIN")).check(matches(not(isEnabled())));
   onView(withId(R.id.username)).perform(typeText("sam"));
   onView(withText("LOGIN")).perform(click());
}
```

Espresso Example

```
@Test
public void testLogin() {
   onView(withText("LOGIN")).check(matches(not(isEnabled())));
   onView(withId(R.id.username)).perform(typeText("sam"));
   onView(withText("LOGIN")).perform(click());
}
```

Espresso Example

```
@Test
public void testLogin() {
   onView(withText("LOGIN")).check(matches(not(isEnabled())));
   onView(withId(R.id.username)).perform(typeText("sam"));
   onView(withText("LOGIN")).perform(click());
}
```

Target Audience for Espresso

“Espresso is **targeted at developers**, who believe that automated testing is an integral part of the development lifecycle. While it can be used for black-box testing, **Espresso’s full power is unlocked by those who are familiar with the codebase under test**.”

Learn More About Espresso

Wojtek Kalicinski

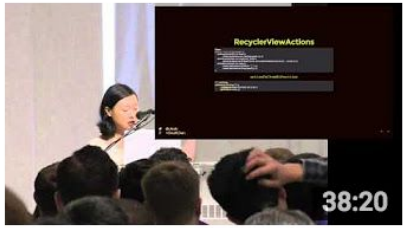
Android Testing Support - Android Testing Patterns #1
Android Developers
13,141 views

UI testing with Espresso - Android Testing Patterns #2
Android Developers
10,687 views

AdapterViews and Espresso - Android Testing Patterns #3
Android Developers
8,892 views

Chiu-Ki Chan

Advanced Android Espresso (Big Android BBQ 2015)
Android Developers
12,028 views



Droidcon NYC 2015 - Advanced Android Espresso
touchlab
2,160 views

Advanced Espresso - Google I/O 2016
Android Developers
9,152 views

Screenshots - WHY?

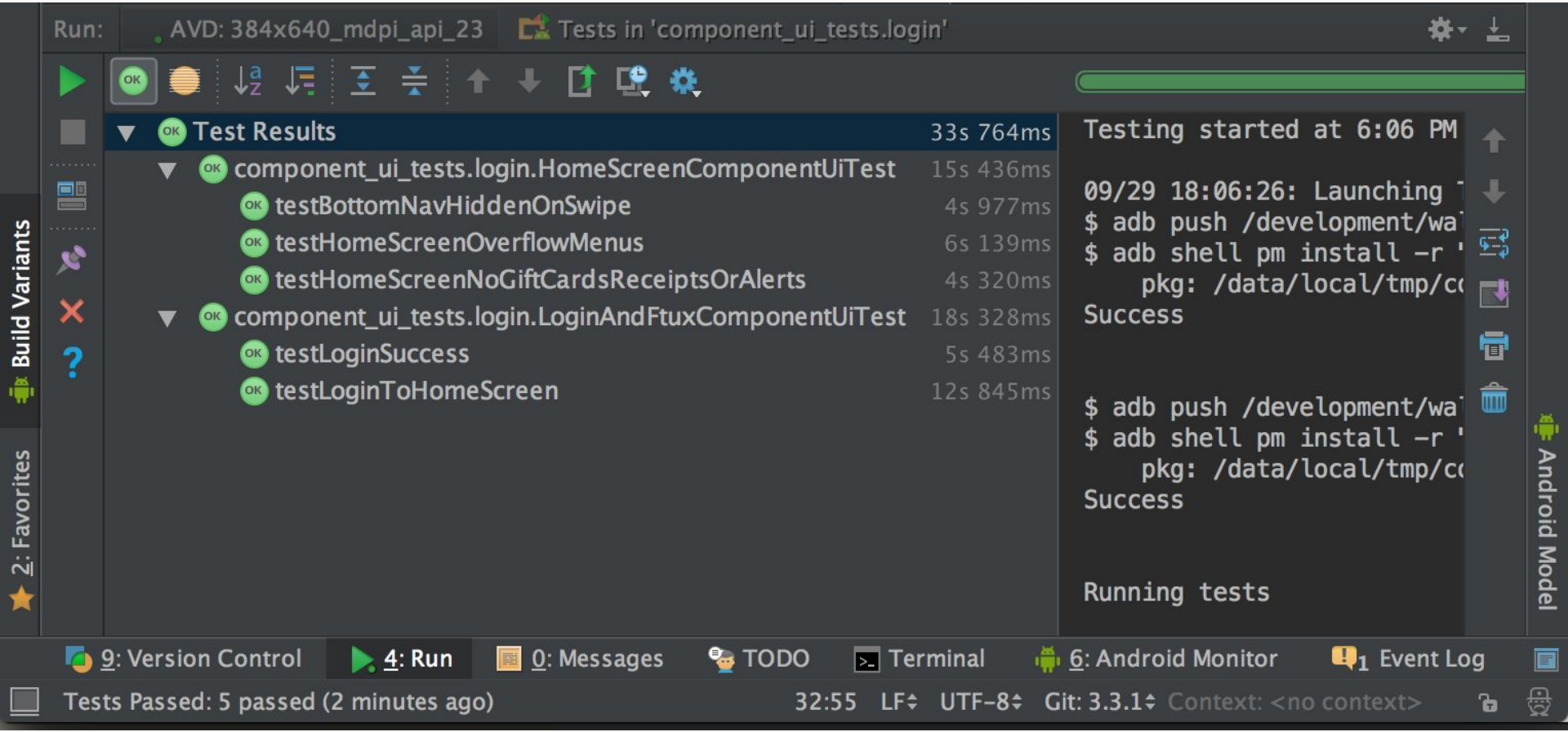
1. See What is Being Tested

Terminal Output Test Results

wallet — -bash — 151×38

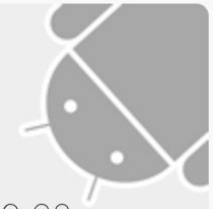
```
2016-09-29 14:07:02 [SDR.printStream] [emulator-5554] STDOUT 2016-09-29 14:07:02 [STRL.testEnded] test=component_ui_tests.rateapp.RateAppTransactionDet
ailComponentUiTest#testRateAppShownOnFirstTransactionDetailViewWithThresholdOfOne
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:24 [STRL.testRunEnded] elapsedTime=365295
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 02:07:24 I/XmlResultReporter: XML test result file generated at /development/wallet/spoon-
output/junit-reports/emulator-5556.xml. Total tests 46, passed 46,
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:24 [SDR.run] About to grab screenshots and prepare output for [emulator-5
556]
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:24 [SDR.pullDirectory] Internal path is /data/data/com.capitalone.mobile.
wallet.qa/app_spoon-screenshots
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:24 [SDR.pullDirectory] External path is /sdcard/app_spoon-screenshots
2016-09-29 14:07:24 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:24 [SDR.pullDirectory] Pulling files from external dir on [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Pulling files from internal dir on [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Done pulling app_spoon-screenshots from on [emulat
or-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Internal path is /data/data/com.capitalone.mobile.
wallet.qa/app_spoon-files
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] External path is /sdcard/app_spoon-files
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Pulling files from external dir on [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Pulling files from internal dir on [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.pullDirectory] Done pulling app_spoon-files from on [emulator-555
6]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.handleImages] Moving screenshots to the image folder on [emulator
-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.handleImages] Generating animated gifs for [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.handleFiles] Found class name dirs: []
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 2016-09-29 14:07:33 [SDR.run] Done running for [emulator-5556]
2016-09-29 14:07:33 [SDR.printStream] [emulator-5556] STDOUT 02:07:33 D/ddms: Waiting for Monitor thread
2016-09-29 14:07:34 [SDR.printStream] [emulator-5556] STDERR 2016-09-29 14:07:33 [SDR.handleImages] Unable to find test for com.capitalone.mobile.walle
t.util.espresso.ScreenshotHelper#takeScreenshot
2016-09-29 14:07:34 [SDR.printStream] [emulator-5556] STDERR 2016-09-29 14:07:33 [SDR.handleImages] Unable to find test for com.capitalone.mobile.walle
t.util.espresso.ScreenshotHelper#takeScreenshot
2016-09-29 14:07:34 [SDR.printStream] [emulator-5556] STDERR 2016-09-29 14:07:33 [SDR.handleImages] Unable to find test for com.capitalone.mobile.walle
t.util.espresso.ScreenshotHelper#takeScreenshot
2016-09-29 14:07:34 [SDR.runInNewProcess] Process.waitFor() finished for [emulator-5556] with exitCode 0
2016-09-29 14:07:35 [SR$1.run] [emulator-5556] Execution done. (1 remaining [emulator-5554])
2016-09-29 14:07:58 [SDR.printStream] [emulator-5554] STDOUT 2016-09-29 14:07:58 [STRL.testRunEnded] elapsedTime=402774
2016-09-29 14:07:58 [SDR.printStream] [emulator-5554] STDOUT 02:07:58 I/XmlResultReporter: XML test result file generated at /development/wallet/spoon-
output/junit-reports/emulator-5554.xml. Total tests 46, failure 1, passed 45,
```

Android Studio Test Results



Spoon Test Report without Screenshots

384x640_mdpi_api_23



70 tests run with 70 passing and 0 failing in 5 minutes, 45 seconds at 2016-09-20 06:27 PM

Running Android 6.0 (API 23)

Login Incorrect Username Or Password

Login Site Down Message

Login Success

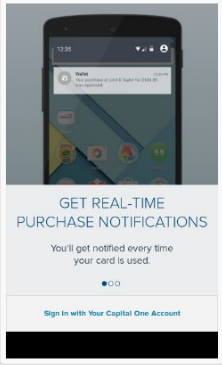
Spoon Test Report with Screenshots

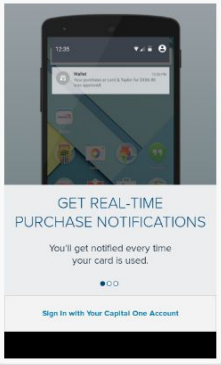
384x640_mdpi_api_23

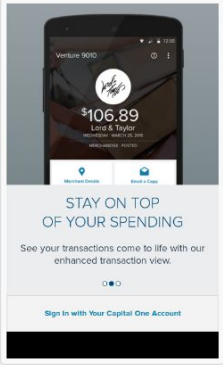
46 tests run with 46 passing and 0 failing in 7 minutes, 1 second at 2016-09-29 01:59 PM

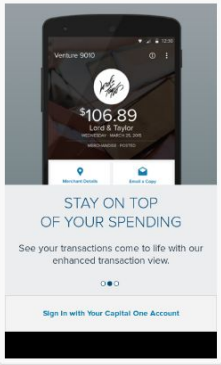
Running Android 6.0 (API 23)

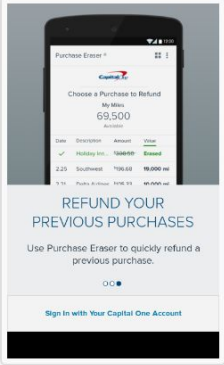
Welcome Screen Pages

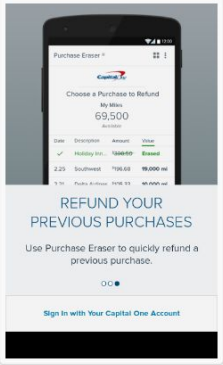


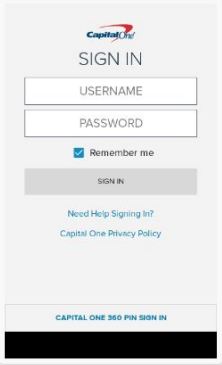












1. See What is Being Tested
2. Diagnose Failures

Oops, Your Test Failed

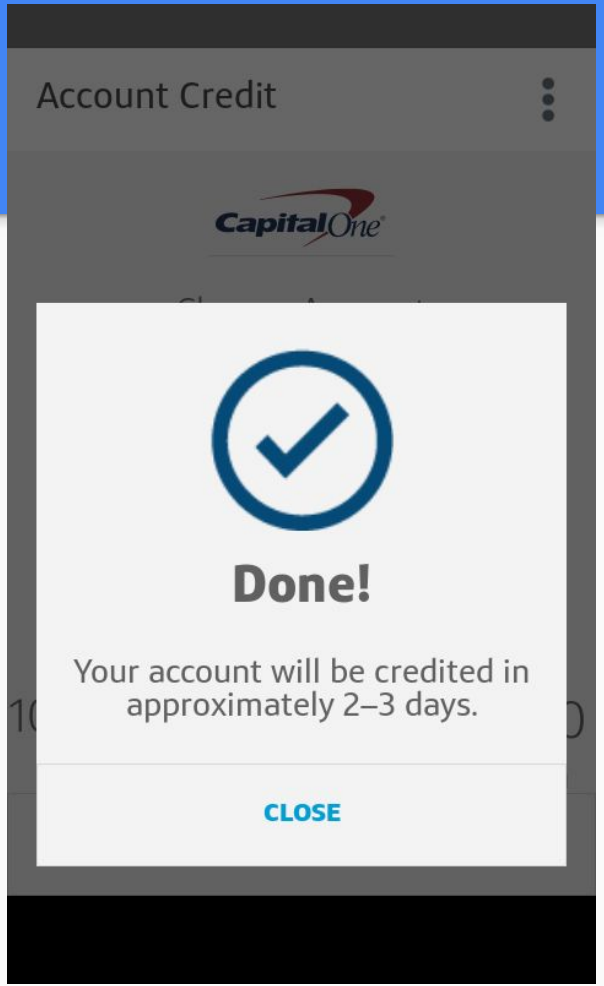
android.support.test.espresso.NoMatchingViewException: No views in hierarchy found matching: with text: “DONE!”

View Hierarchy:
+>DecorView{id=-1, visibility=VISIBLE, width=720, height=1280, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=0.0, child-count=3}
|
+->ActionBarOverlayLayout{id=16909230, res-name=decor_content_parent, visibility=VISIBLE, width=720, height=1184, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=0.0, child-count=2}
|
+-->FrameLayout{id=16908290, res-name=content, visibility=VISIBLE, width=720, height=1024, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=160.0, child-count=1}
|
+--->TextView{id=2131230720, res-name=greeting, visibility=VISIBLE, width=656, height=960, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=32.0, y=32.0, text=Hello world!, input-type=0, ime-target=false, has-links=false}
|
+-->ActionBarContainer{id=16909231, res-name=action_bar_container, visibility=VISIBLE, width=720, height=112, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=48.0, child-count=2}
|

Ahh, I See Why

android.support.test.espresso.NoMatchingViewException: No views in hierarchy found matching: with text: “DONE!”

View Hierarchy:
+>DecorView{id=-1, visibility=VISIBLE, width=720, height=1280, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=0.0, child-count=3}
|
+->ActionBarOverlayLayout{id=16909230, res-name=decor_content_parent, visibility=VISIBLE, width=720, height=1184, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false, has-input-connection=false, x=0.0, y=0.0, child-count=2}
|
+-->FrameLayout{id=16908290, res-name=content, visibility=VISIBLE, width=720, height=1024, has-focus=false, has-focusable=false, has-window-focus=true, is-clickable=false, is-enabled=true, is-focused=false, is-focusable=false, is-layout-requested=false, is-selected=false, root-is-layout-requested=false,



Screenshots on FAILURES

```
Espresso.setFailureHandler(new FailureHandler() {
    @Override
    public void handle(Throwable throwable, Matcher<View> matcher) {
        ScreenshotHelper.takeScreenshotForcefully("test_failed");
        try {
            new DefaultFailureHandler(applicationContext).handle(throwable, matcher);
        } catch (Exception e) {
            logger.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
});
```

Screenshots on FAILURES

```
Espresso.setFailureHandler(new FailureHandler() {
   @Override
   public void handle(Throwable throwable, Matcher<View> matcher) {
      ScreenshotHelper.takeScreenshotForcefully("test_failed");
      try {
         new DefaultFailureHandler(applicationContext).handle(throwable, matcher);
      } catch (Exception e) {
         logger.error(e.getMessage(), e);
         throw new RuntimeException(e);
      }
   }
});
```

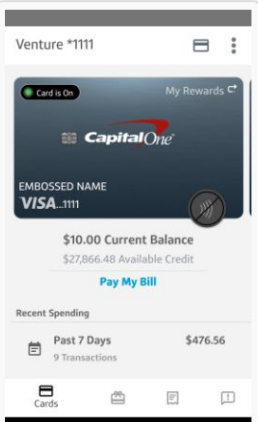
Screenshots on FAILURES

```
Espresso.setFailureHandler(new FailureHandler() {
    @Override
    public void handle(Throwable throwable, Matcher<View> matcher) {
        ScreenshotHelper.takeScreenshotForcefully("test_failed");
        try {
            new DefaultFailureHandler(applicationContext).handle(throwable, matcher);
        } catch (Exception e) {
            logger.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
});
```

Screenshot on FAILURE Example Report

Credit Card Smoke Test

java.lang.RuntimeException: android.support.test.espresso.NoMatchingViewException: No views in hierarchy found matching: with id: com.capitalone.mobile.wallet.qa:id/negativeButton
If the target view is not part of the view hierarchy, you may need to use Espresso.onData to load it from one of the following AdapterViews:com.tonicartos.widget.stickygridheaders.StickyGridHeadersGridView{8e1ab82 G.ED.VC.. .....ID 0,0-1080,1608 #7f0f0165 app:id/grid}



1. See What is Being Tested
2. Diagnose Failures
3. Share Your Tests

NO ONE Sees The Value of Your Tests… YET

- Your Boss
- Product Owners
- Designers
- Manual Testers

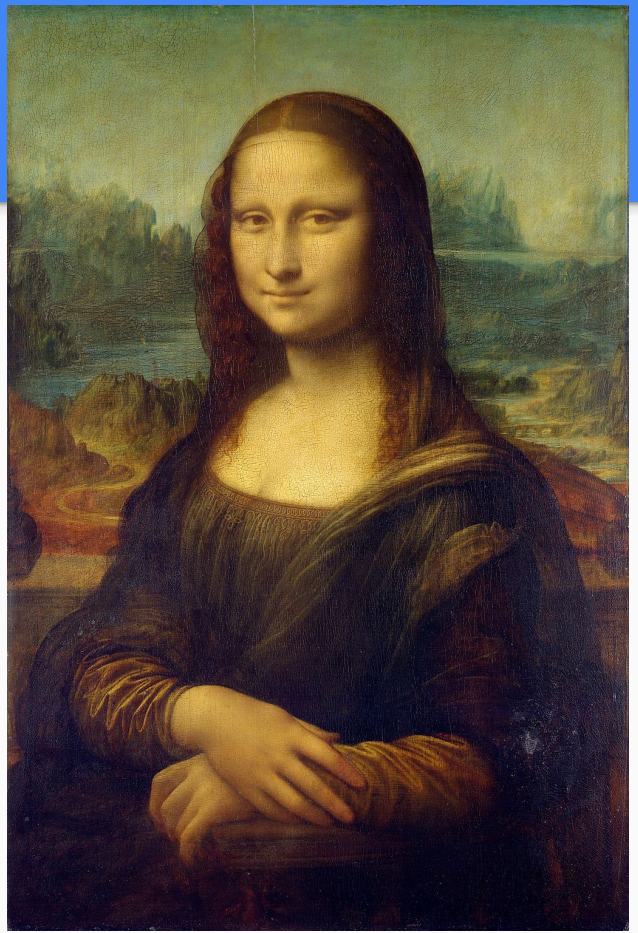
That’s because they can’t tell what your tests are doing and don’t find any use for them.

To them it’s just console output.

Let EVERYONE see what’s being tested.

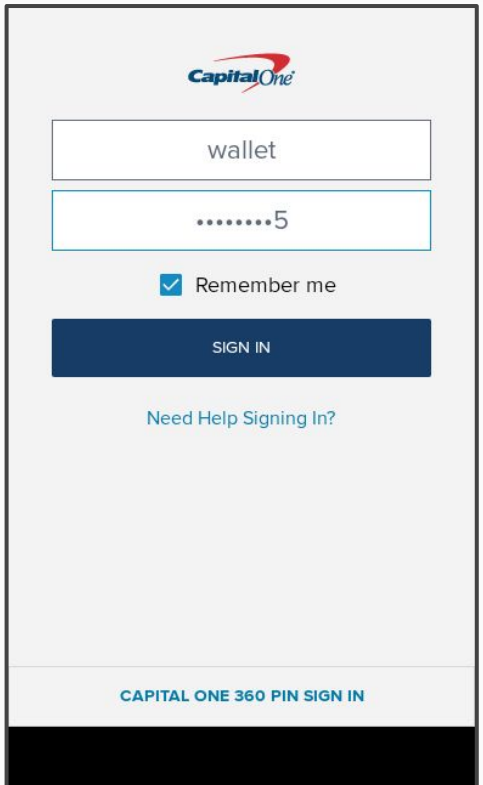
User Interfaces are VISUAL

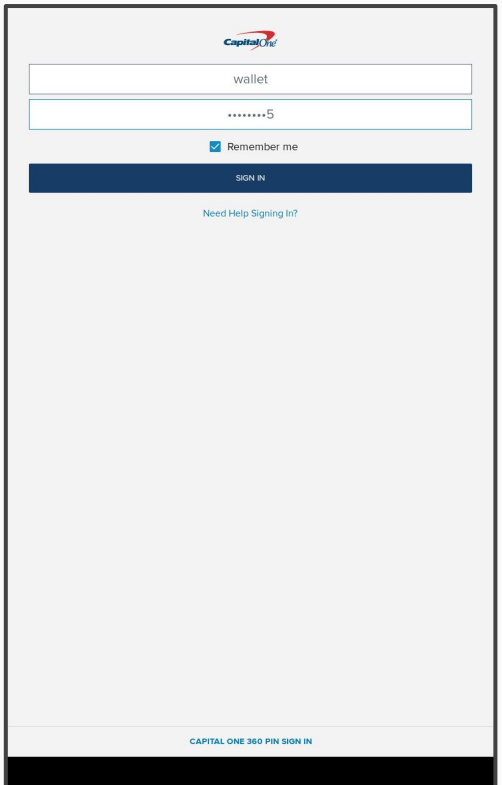
- A User Interface is something very hard to represent in a non-visual way.
- Exercise:
  - Imagine you need to tell someone who is blind what the Mona Lisa painting looks like. What would you say?



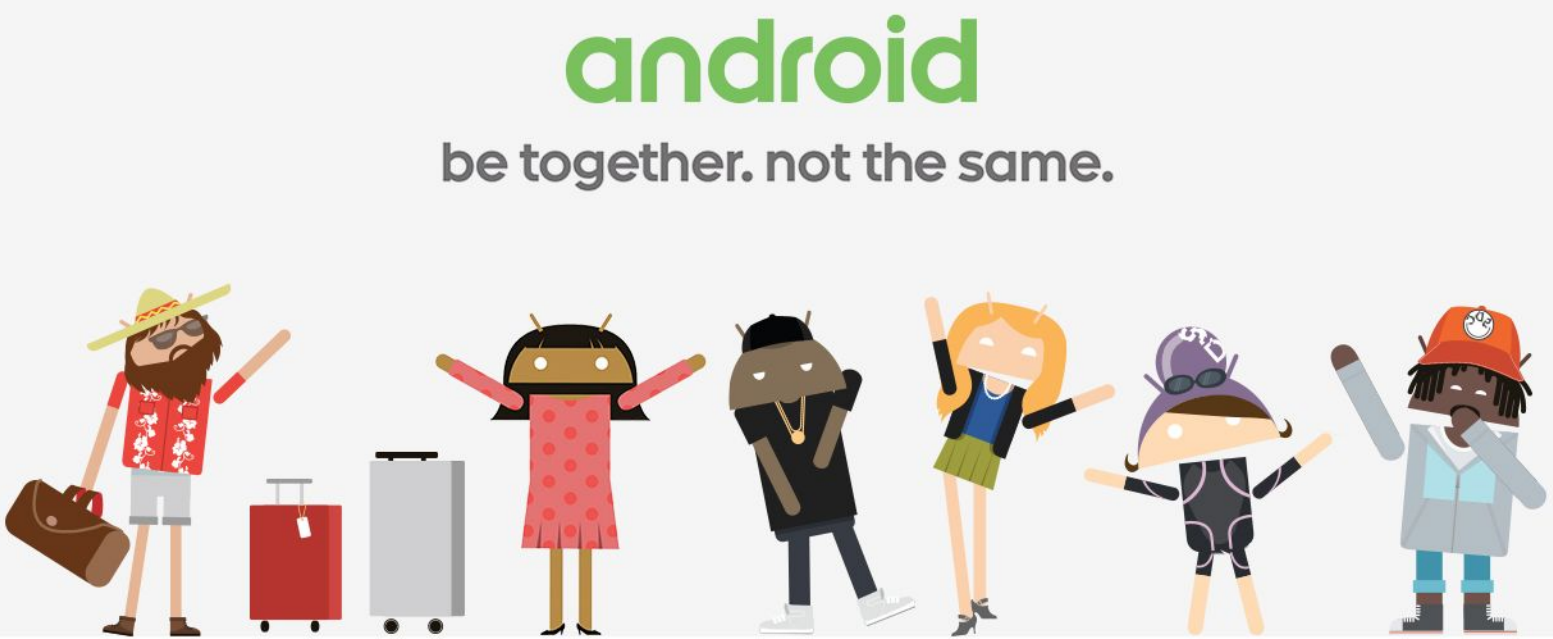
https://en.wikipedia.org/wiki/Mona_Lisa

Same Test,
But Looks Different on Different Devices





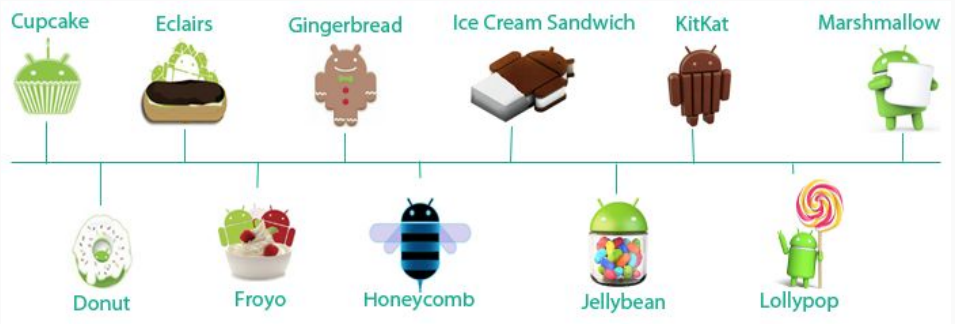
Android Embraces Differences



Fragmentation and Permutations

DEVICE SIZE

ANDROID OS VERSION



DEVICE TYPE

LANGUAGE

Spoon - http://square.github.io/spoon/

Spoon

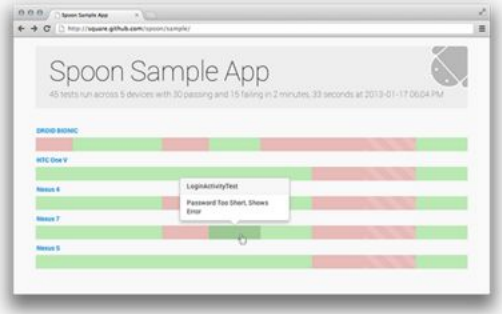
Download v1.7.0

Distributing **instrumentation tests** to all your Androids

Introduction

Android's ever-expanding ecosystem of devices creates a unique challenge to testing applications. Spoon aims to simplify this task by distributing instrumentation test execution and displaying the results in a meaningful way.

Instead of attempting to be a new form of testing, Spoon makes existing instrumentation tests more useful. Using the application APK and instrumentation APK, Spoon runs the tests on multiple devices simultaneously. Once all tests have completed, a static HTML summary is generated with detailed information about each device and test.



With the high-level output you can immediately see whether or not a test failure is specific to a single device or all devices. This view is designed to be displayed on a dedicated monitor or TV.

Spoon will run on all targets which are visible to `adb devices`. Plug in multiple different phones and tablets, start different configurations of emulators, or use some combination of both!

The greater diversity of the targets in use, the more useful the output will be in visualizing your applications.

DEVICE VIEW

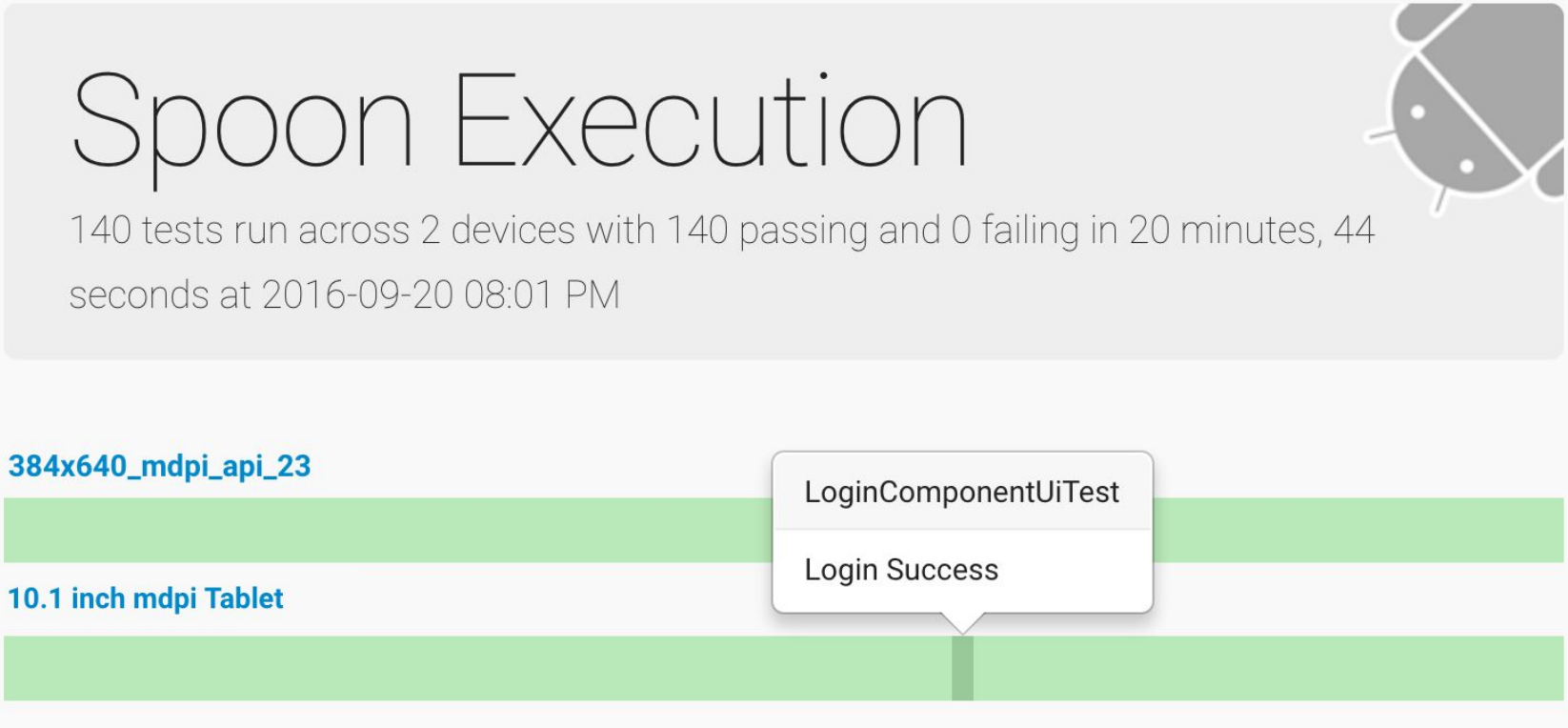
The device view outlines the results of each test one a single device. This is useful for tracking down device-specific failures of individual tests.

Spoon Test Runner

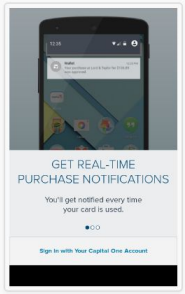
- Run instrumentation tests in parallel
- Shard tests across multiple devices
- Beautiful HTML reports

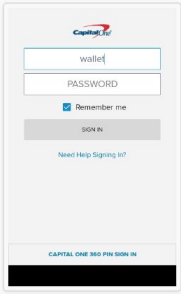
```
java -jar spoon-runner-1.7.0-jar-with-dependencies.jar \
    --apk example-app.apk \
    --test-apk example-tests.apk
```

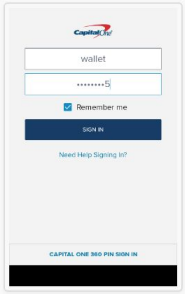
Spoon HTML Reports

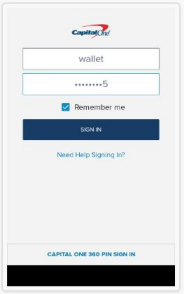


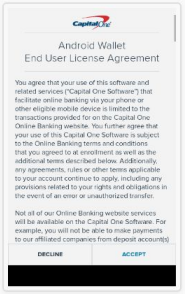
Visualize What Your Tests are Doing

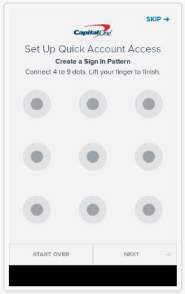


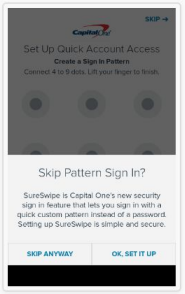


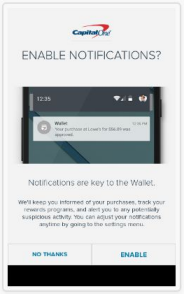


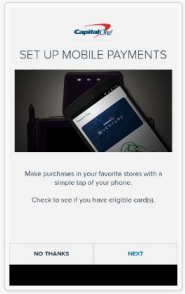


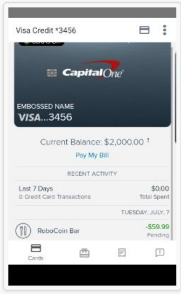


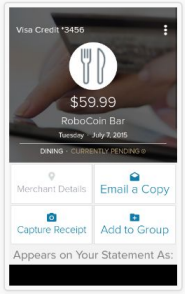


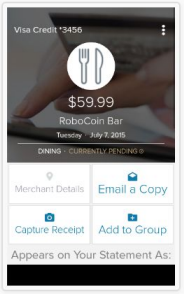


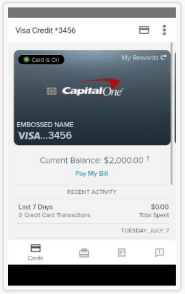


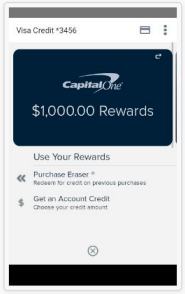


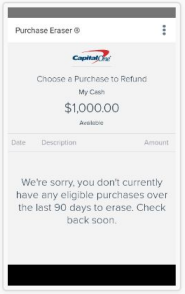


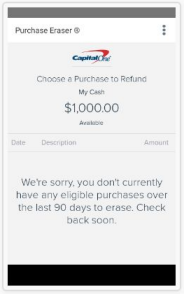


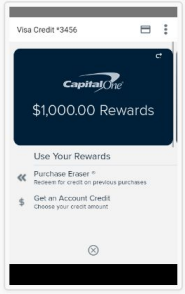


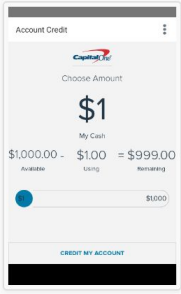


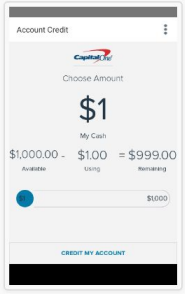


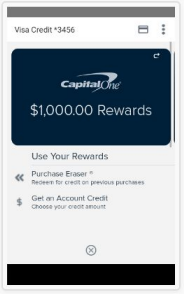


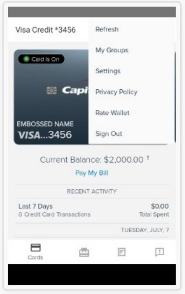




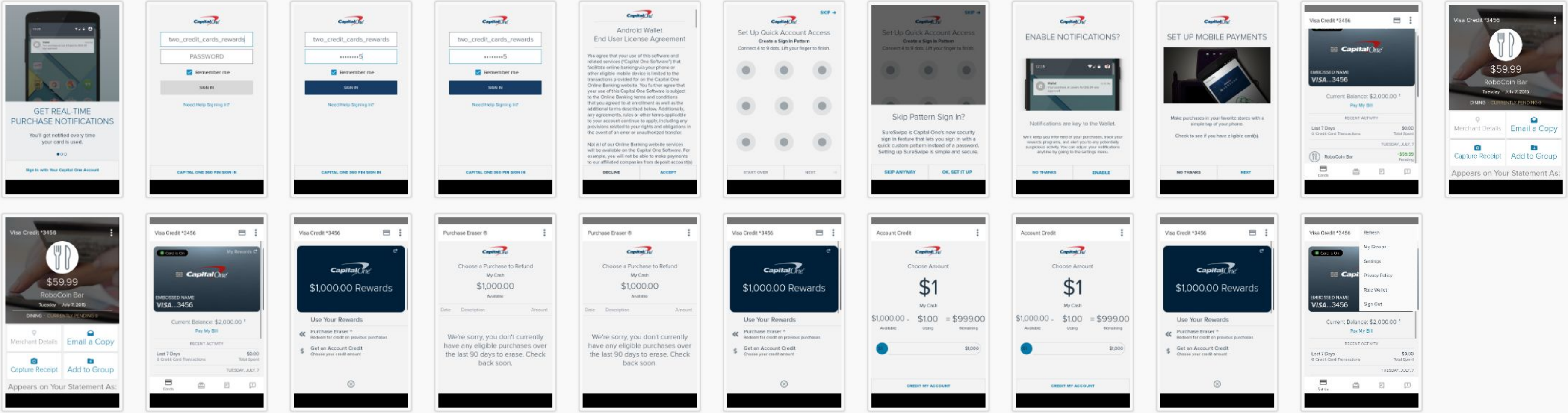




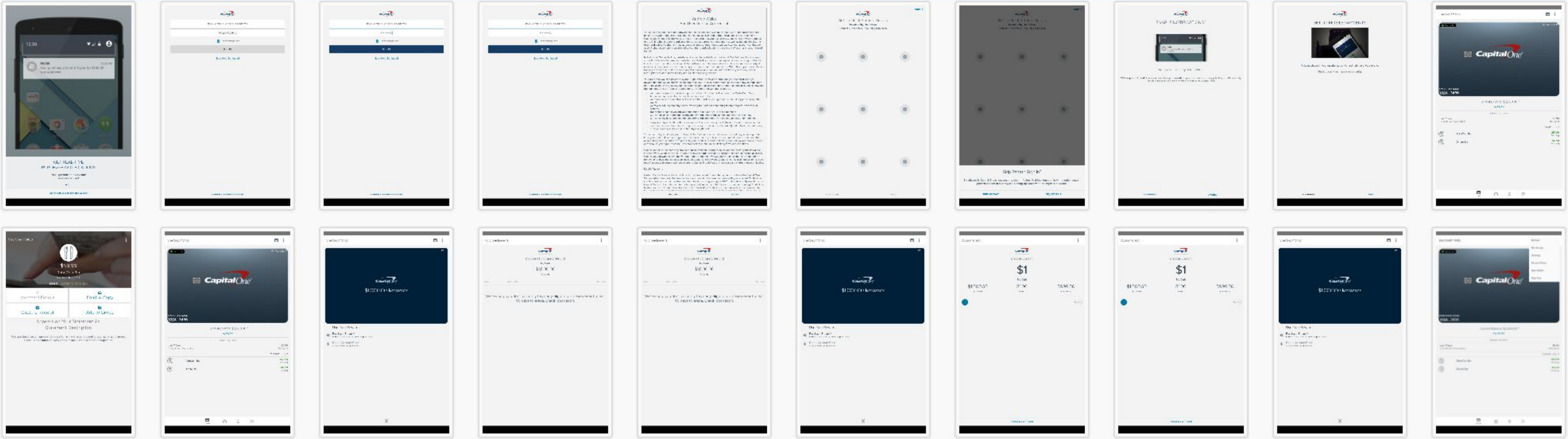


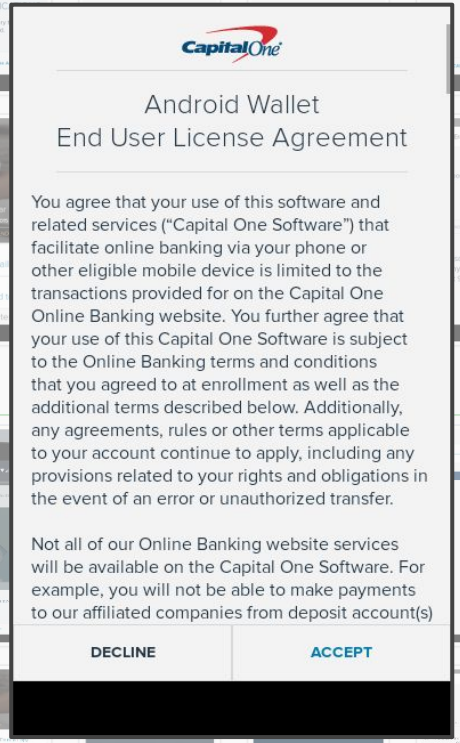


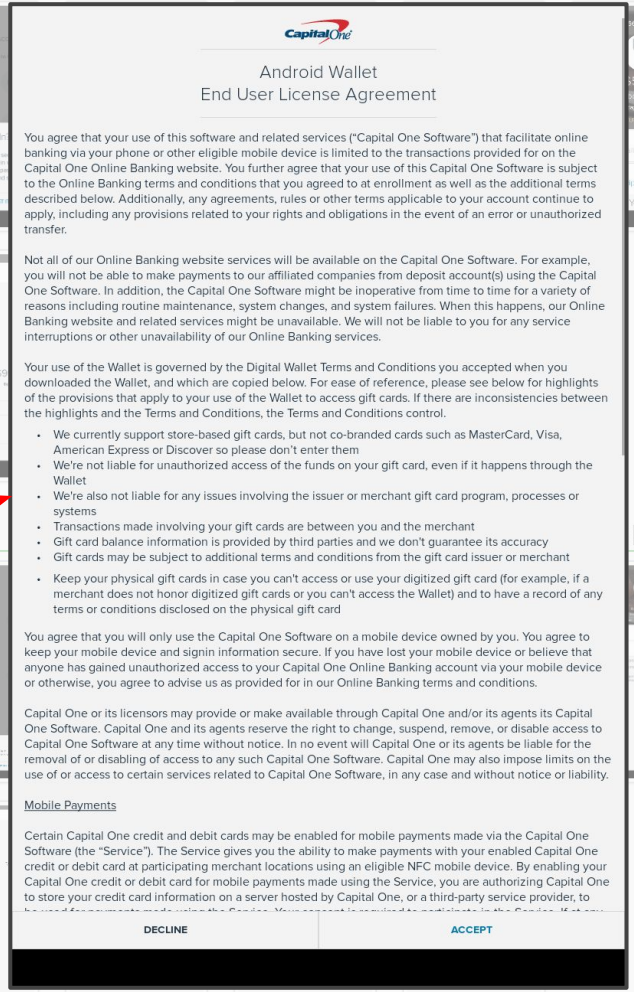
384x640_mdpi_api_23



10.1



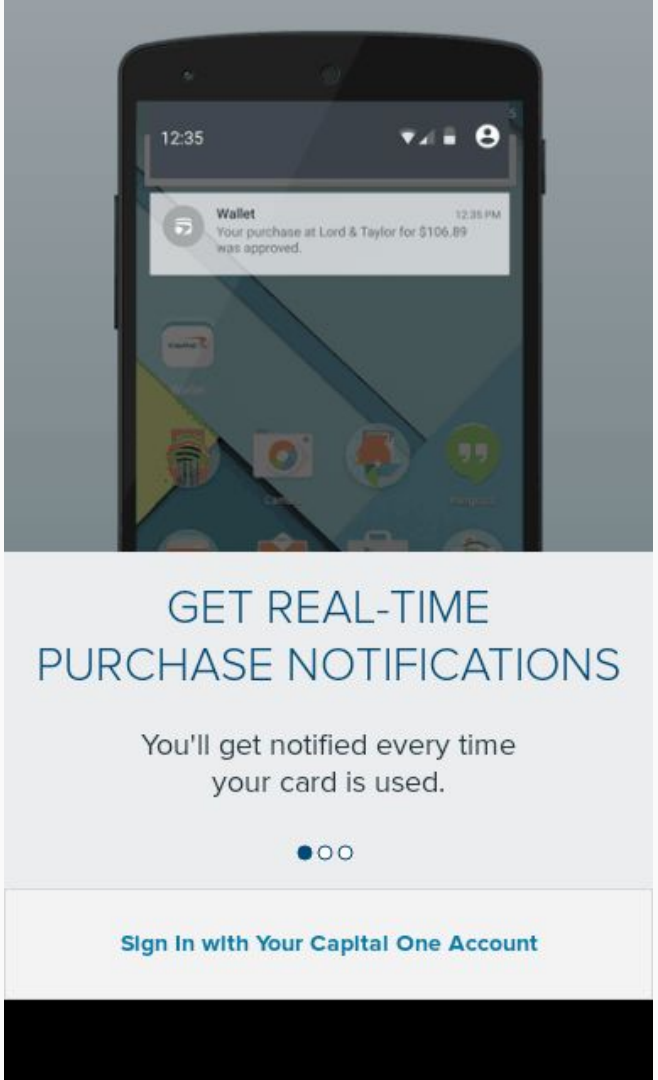

Android Wallet
End User License Agreement
You agree that your use of this software and related services ("Capital One Software") that facilitate online banking via your phone or other eligible mobile device is limited to the transactions provided for on the Capital One Online Banking website. You further agree that your use of this Capital One Software is subject to the Online Banking terms and conditions that you agreed to at enrollment as well as the additional terms described below. Additionally, any agreements, rules or other terms applicable to your account continue to apply, including any provisions related to your rights and obligations in the event of an error or unauthorized transfer.
Not all of our Online Banking website services will be available on the Capital One Software. For example, you will not be able to make payments to our affiliated companies from deposit account(s)
DECLINE
ACCEPT


Android Wallet
End User License Agreement
You agree that your use of this software and related services ("Capital One Software") that facilitate online banking via your phone or other eligible mobile device is limited to the transactions provided for on the Capital One Online Banking website. You further agree that your use of this Capital One Software is subject to the Online Banking terms and conditions that you agreed to at enrollment as well as the additional terms described below. Additionally, any agreements, rules or other terms applicable to your account continue to apply, including any provisions related to your rights and obligations in the event of an error or unauthorized transfer.
Not all of our Online Banking website services will be available on the Capital One Software. For example, you will not be able to make payments to our affiliated companies from deposit account(s) using the Capital One Software. In addition, the Capital One Software might be inoperative from time to time for a variety of reasons including routine maintenance, system changes, and system failures. When this happens, our Online Banking website and related services might be unavailable. We will not be liable to you for any service interruptions or other unavailability of our Online Banking services.
Your use of the Wallet is governed by the Digital Wallet Terms and Conditions you accepted when you downloaded the Wallet, and which are copied below. For ease of reference, please see below for highlights of the provisions that apply to your use of the Wallet to access gift cards. If there are inconsistencies between the highlights and the Terms and Conditions, the Terms and Conditions control.
• We currently support store-based gift cards, but not co-branded cards such as MasterCard, Visa, American Express or Discover so please don't enter them
• We're not liable for unauthorized access of the funds on your gift card, even if it happens through the Wallet
• We're also not liable for any issues involving the issuer or merchant gift card program, processes or systems
• Transactions made involving your gift cards are between you and the merchant
• Gift card balance information is provided by third parties and we don't guarantee its accuracy
• Gift cards may be subject to additional terms and conditions from the gift card issuer or merchant
• Keep your physical gift cards in case you can't access or use your digitized gift card (for example, if a merchant does not honor digitized gift cards or you can't access the Wallet) and to have a record of any terms or conditions disclosed on the physical gift card
You agree that you will only use the Capital One Software on a mobile device owned by you. You agree to keep your mobile device and signin information secure. If you have lost your mobile device or believe that anyone has gained unauthorized access to your Capital One Online Banking account via your mobile device or otherwise, you agree to advise us as provided for in our Online Banking terms and conditions.
Capital One or its licensors may provide or make available through Capital One and/or its agents its Capital One Software. Capital One and its agents reserve the right to change, suspend, remove, or disable access to Capital One Software at any time without notice. In no event will Capital One or its agents be liable for the removal of or disabling of access to any such Capital One Software. Capital One may also impose limits on the use of or access to certain services related to Capital One Software, in any case and without notice or liability.
Mobile Payments
Certain Capital One credit and debit cards may be enabled for mobile payments made via the Capital One Software (the "Service"). The Service gives you the ability to make payments with your enabled Capital One credit or debit card at participating merchant locations using an eligible NFC mobile device. By enabling your Capital One credit or debit card for mobile payments made using the Service, you are authorizing Capital One to store your credit card information on a server hosted by Capital One, or a third-party service provider, to
DECLINE
ACCEPT

10.1

GIFs

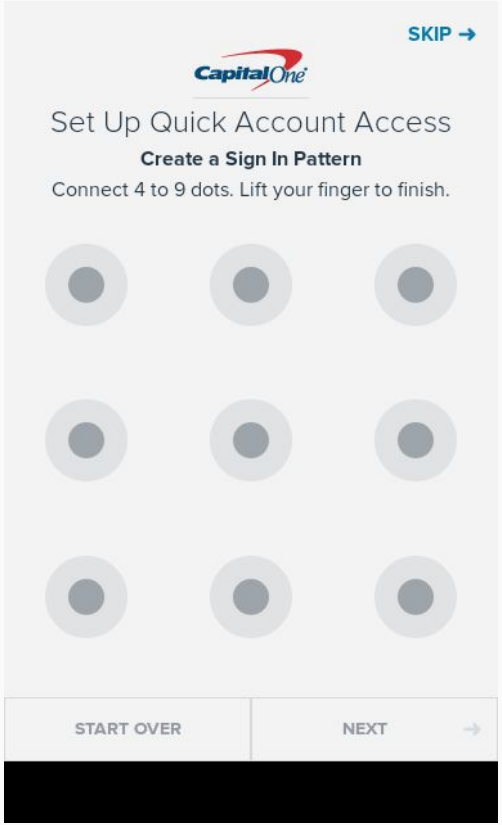
- Generated by default
- Processing overhead
  - 11 Seconds for 21 Screenshots ->
- Disable using “--no-animations”



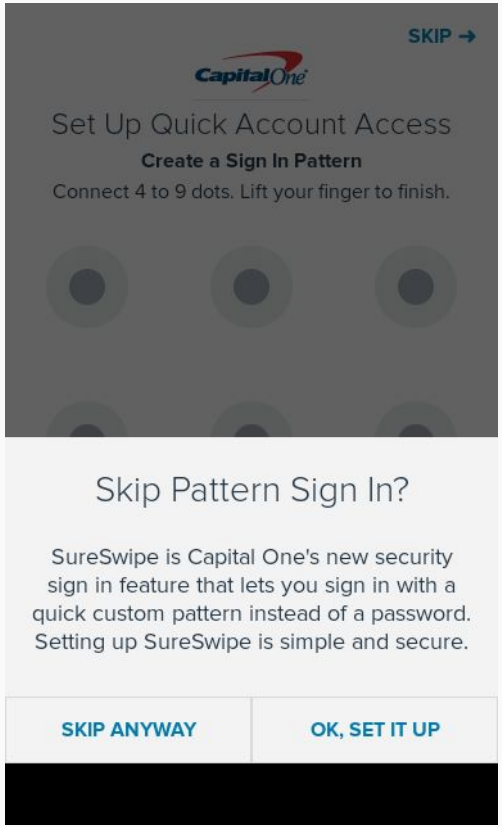
Screenshots - HOW?

Screenshot Libraries - Dialog Comparison

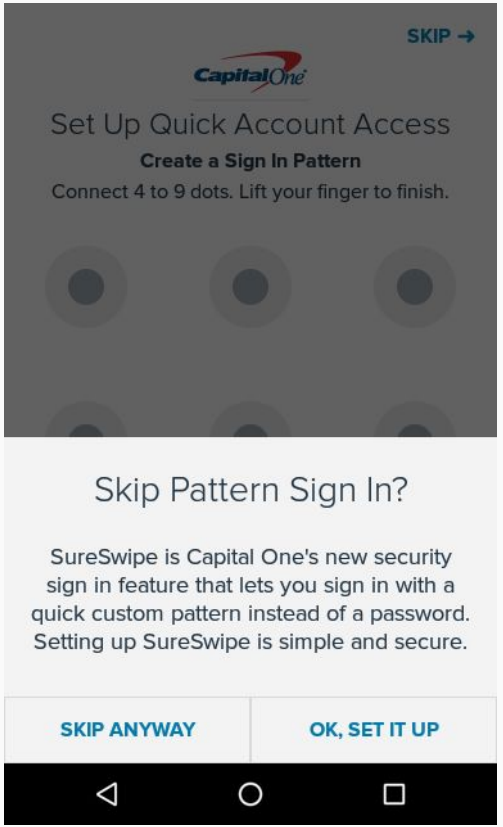
Spoon



Falcon

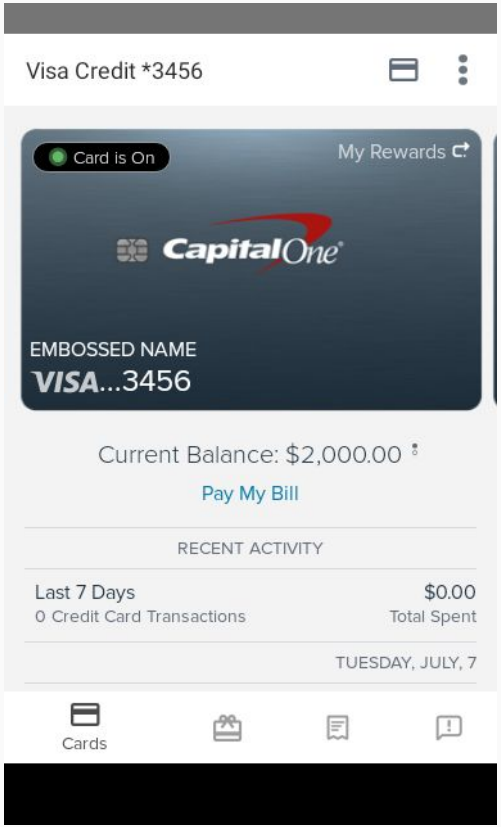


UiAutomator

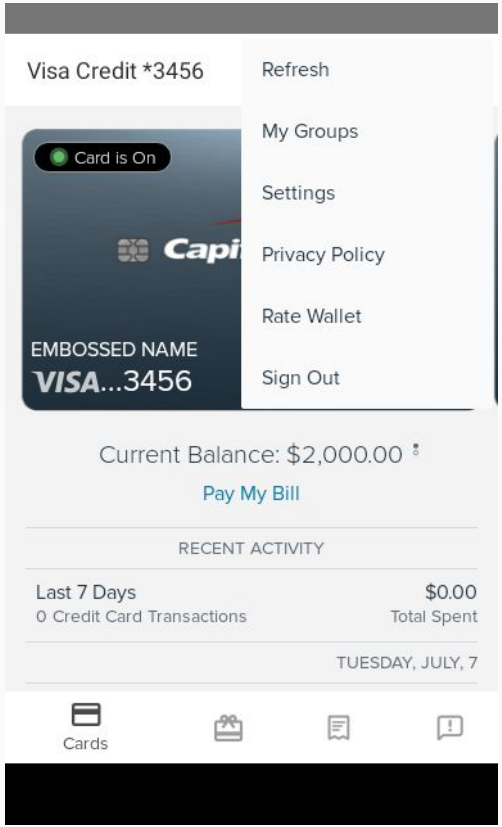


Screenshot Libraries - Overflow Menus Comparison

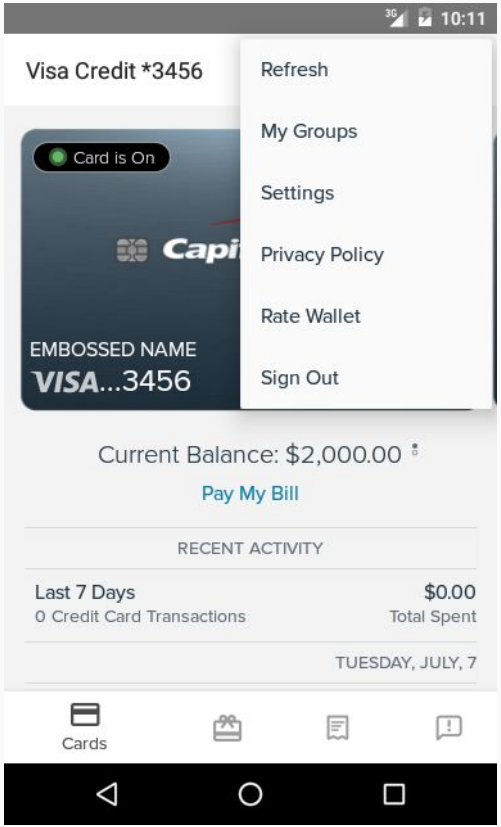
Spoon



Falcon



UiAutomator



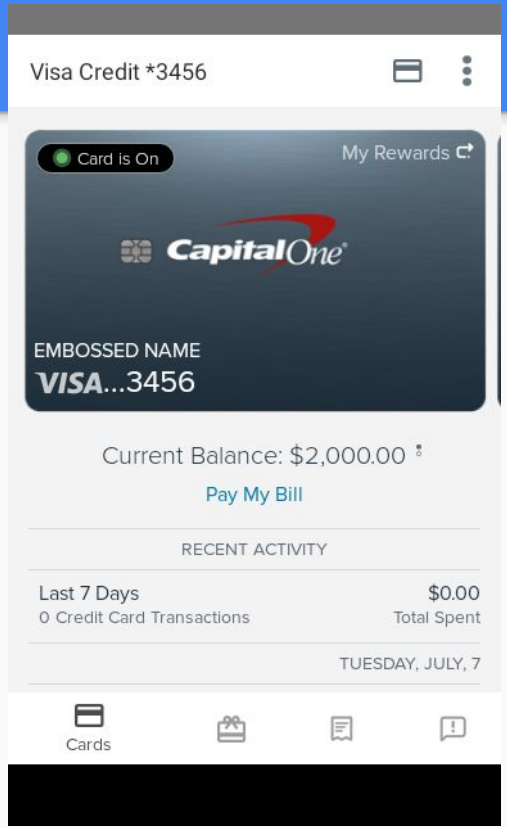
Spoon

Spoon.screenshot(activity, tag);

Spoon.screenshot(activity, tag, testClassName, testMethodName);

https://github.com/square/spoon

https://github.com/square/spoon/issues/4

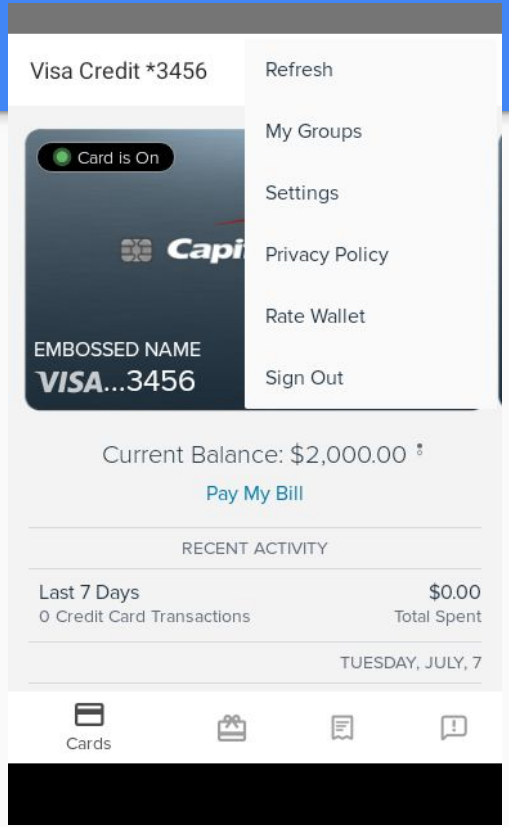


Falcon & (Falcon Spoon Compat)

FalconSpoon.screenshot(activity, tag);

FalconSpoon.screenshot(activity, tag, testClassName, testMethodName);

https://github.com/jraska/Falcon/

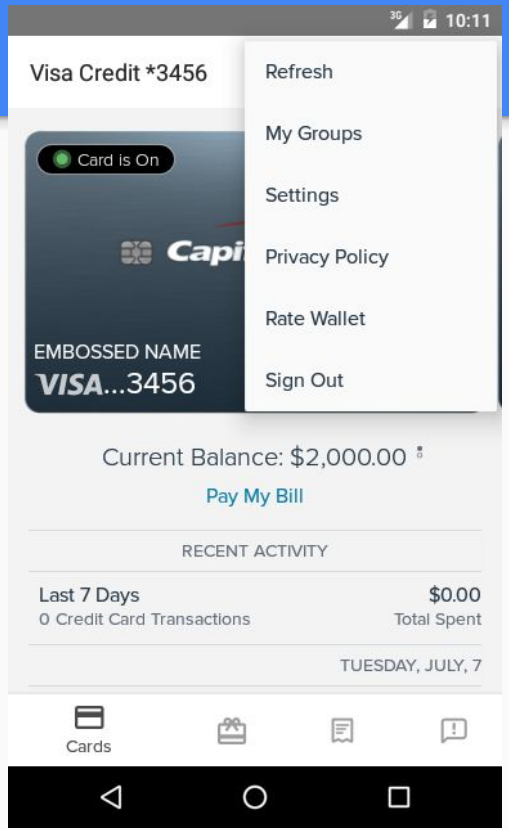


UiAutomator

File outputFile = Spoon.screenshot(activity, tag, testClass, testMethod);
uiDevice.takeScreenshot(outputFile);

- **Works inside and outside of your app**
- **API 18+ (Jelly Bean MR2)**

https://developer.android.com/topic/libraries/testing-support-library/index.html#UIAutomator



Other Alternatives

- Firebase Test Lab's "Screenshotter"
  - https://firebase.google.com/docs/test-lab/test-screenshots
- Fastlane's "Screengrab"
  - https://github.com/fastlane/fastlane/tree/master/screengrab

Get Currently RESUMED Activity

```
private static Activity getCurrentActivityInstance() {
   final Activity[] activity = new Activity[1];
   getInstrumentation().runOnMainSync(new Runnable() {
      public void run() {
         Collection resumedActivities = ActivityLifecycleMonitorRegistry.getInstance()
         .getActivitiesInStage(RESUMED);
         if (resumedActivities.iterator().hasNext()) {
            Activity currentActivity = (Activity) resumedActivities.iterator().next();
            activity[0] = currentActivity;
            logger.trace("Activity obtained of type: " +
currentActivity.getClass().getName());
         }
      }
   });
   return activity[0];
}
```

Get Test Method & Class from Current Stack Trace

```
public static StackTraceElement getTestMethodStackTraceElement() {
  StackTraceElement[] stackTrace = Thread.currentThread().getStackTrace();
  for (int i = 4; i < stackTrace.length; i++) {
    StackTraceElement stackTraceElement = stackTrace[i];
      try {
        Method method = Class.forName(stackTraceElement.getClassName()).getMethod(
            stackTraceElement.getMethodName());
        Annotation annotation = method.getAnnotation(Test.class);
        if (annotation != null) {
          return stackTraceElement;
        }
      } catch (Exception e) {
        // That's fine, will look at the next stack trace element
      }
  }
  return stackTrace[4];
}
```

Get Test Method & Class from Current Stack Trace

```
public static StackTraceElement getTestMethodStackTraceElement() {
  StackTraceElement[] stackTrace = Thread.currentThread().getStackTrace();
  for (int i = 4; i < stackTrace.length; i++) {
    StackTraceElement stackTraceElement = stackTrace[i];
      try {
        Method method = Class.forName(stackTraceElement.getClassName()).getMethod(
            stackTraceElement.getMethodName());
        Annotation annotation = method.getAnnotation(Test.class);
        if (annotation != null) {
          return stackTraceElement;
        }
      } catch (Exception e) {
        // That's fine, will look at the next stack trace element
      }
  }
  return stackTrace[4];
}
```

Resulting Screenshot Code

screenshot(activity, “username_entered”, testClass, testMethod);

takeScreenshot(“username_entered”);

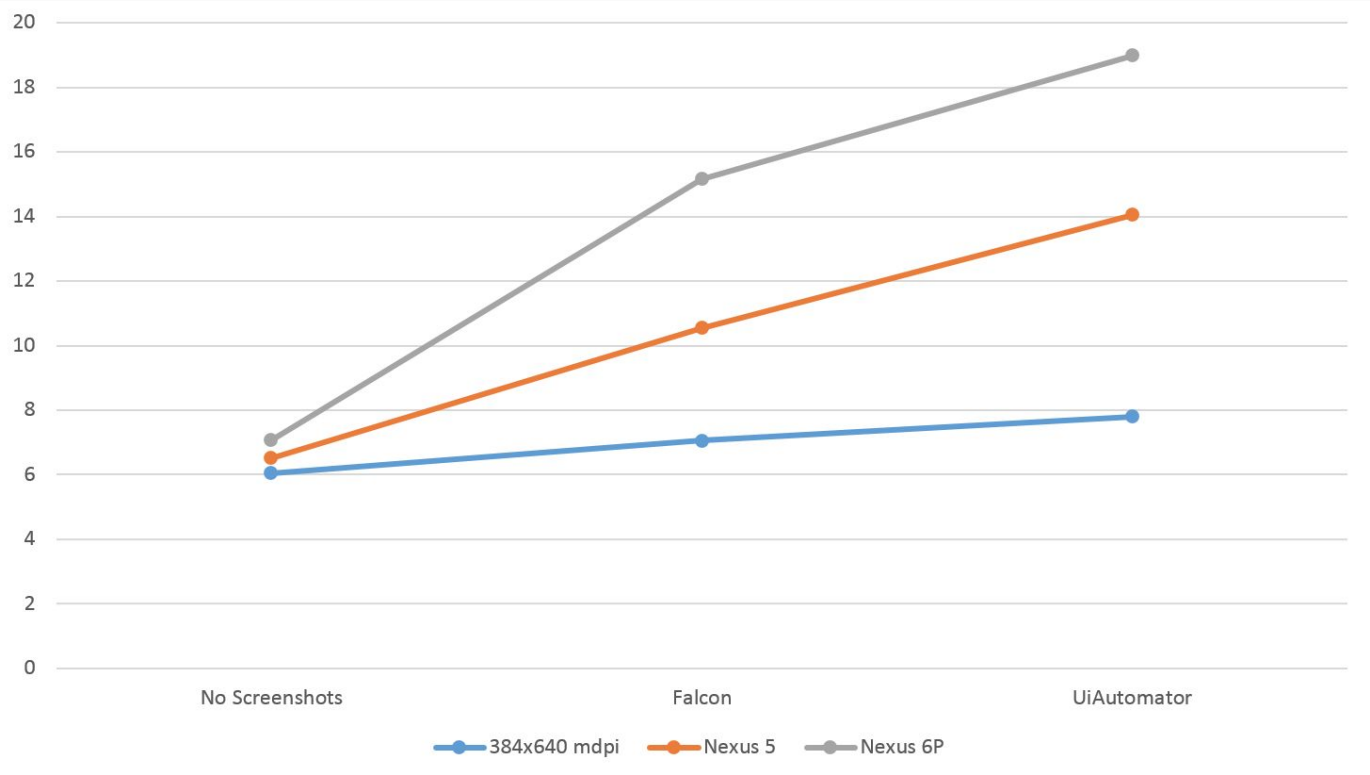
Screenshots - WHEN?

Well… It Depends

- ALWAYS on FAILURE
- To generate reports for product and design teams
- On small/low-res emulators during test development/debugging

Screenshots - The Cost?

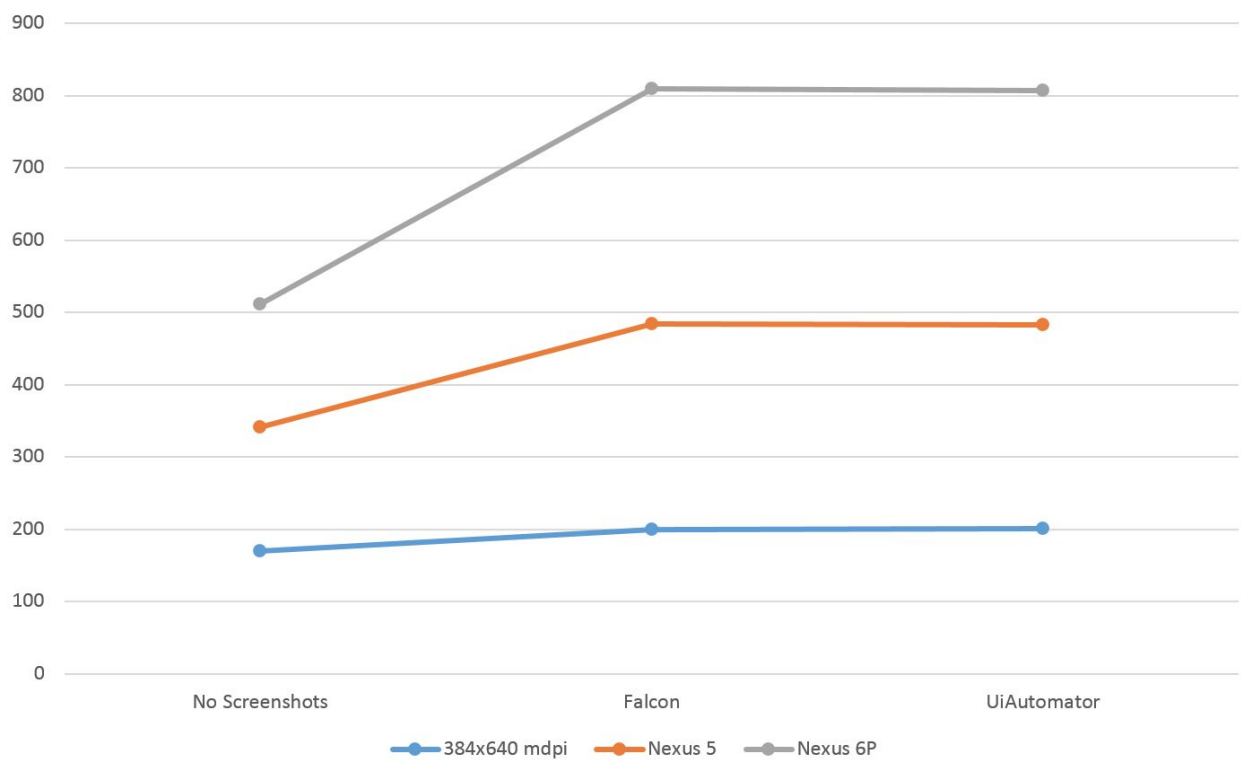
Execution Time in Minutes



Execution Time Overhead in Minutes

	384x640 mdpi	**Nexus 5**	**Nexus 6P**
Falcon	+16.53%	+62.31%	+114.39%
UiAutomator	+28.93%	+116.15%	+168.87%

Report Size in MB



Report Size Overhead in MB

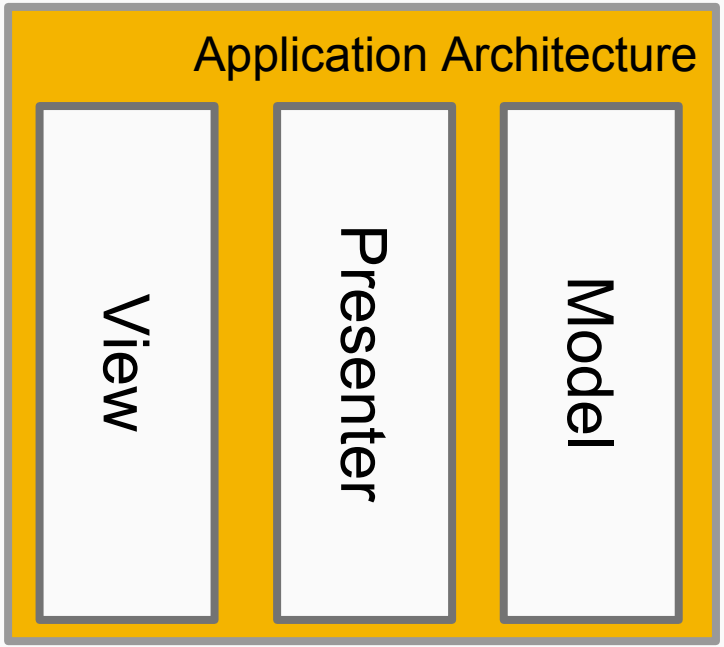
	384x640 mdpi	**Nexus 5**	**Nexus 6P**
Falcon	+17.35%	+66.41%	+90.13%
UiAutomator	+18.41%	+64.77%	+89.19%

Maintainable Test Architecture (Screenshots for Free)

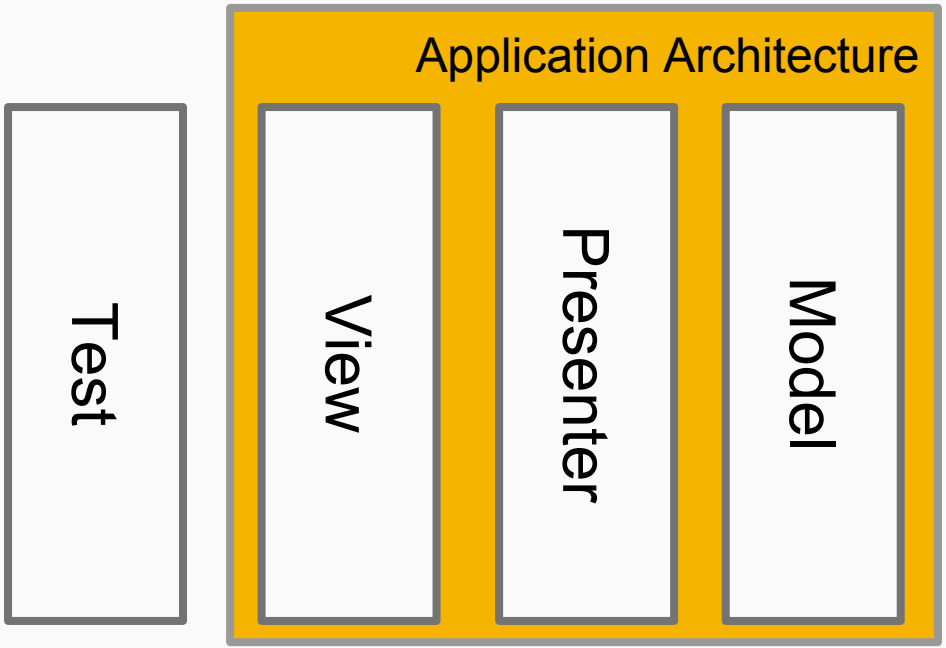
Robot Testing Pattern

- A Test Architecture Pattern for Maintainability
- Learned about it from Jake Wharton's talk:
  - VIDEO: https://realm.io/news/kau-jake-wharton-testing-robots/
  - SLIDES: https://speakerdeck.com/jakewharton/testing-robots-kotlin-night-may-2016

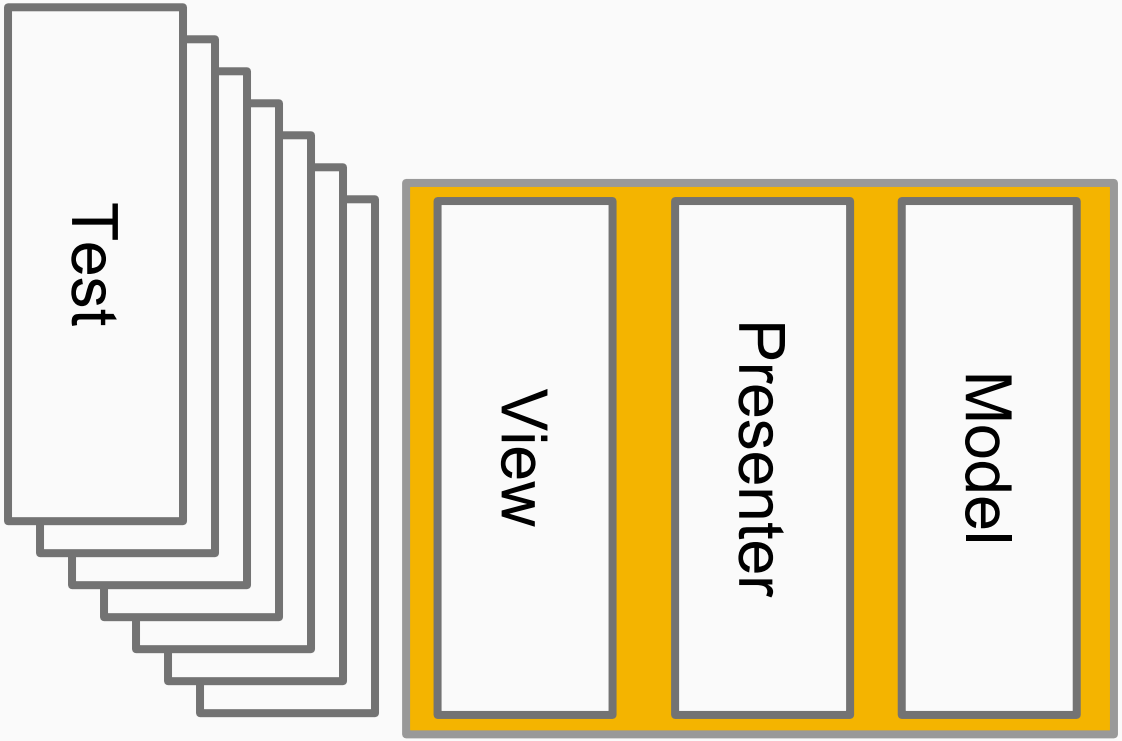
Robot Testing Pattern



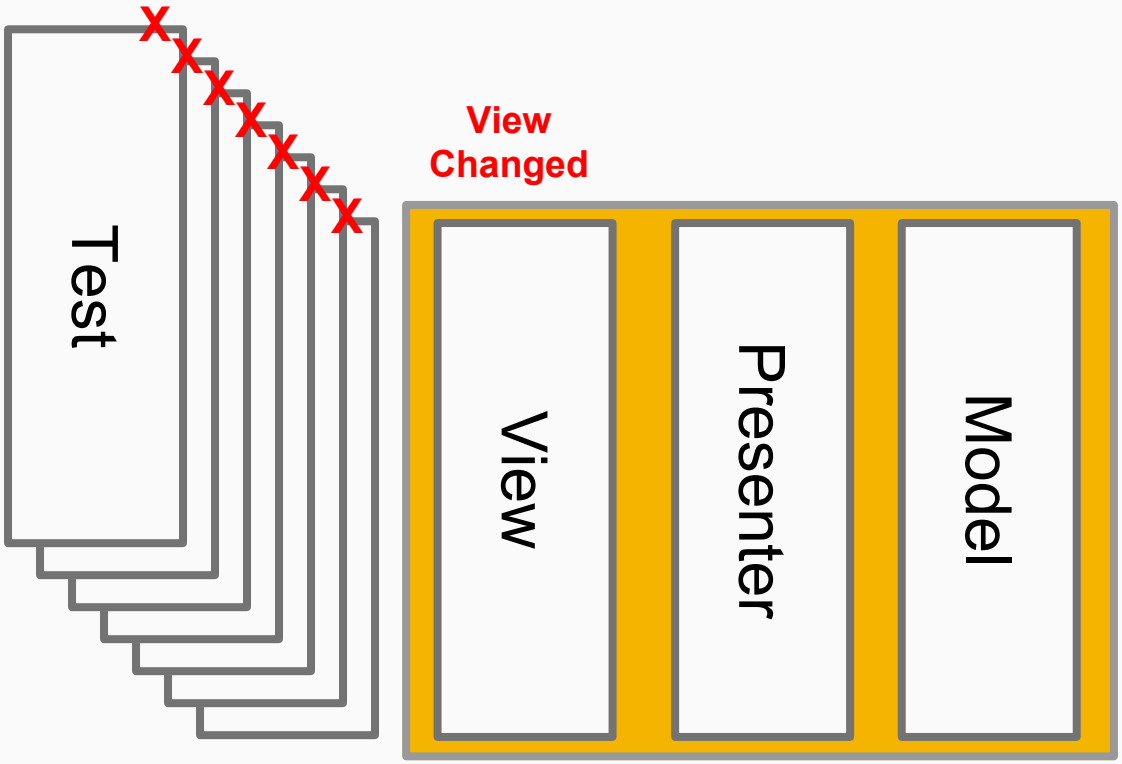
Robot Testing Pattern



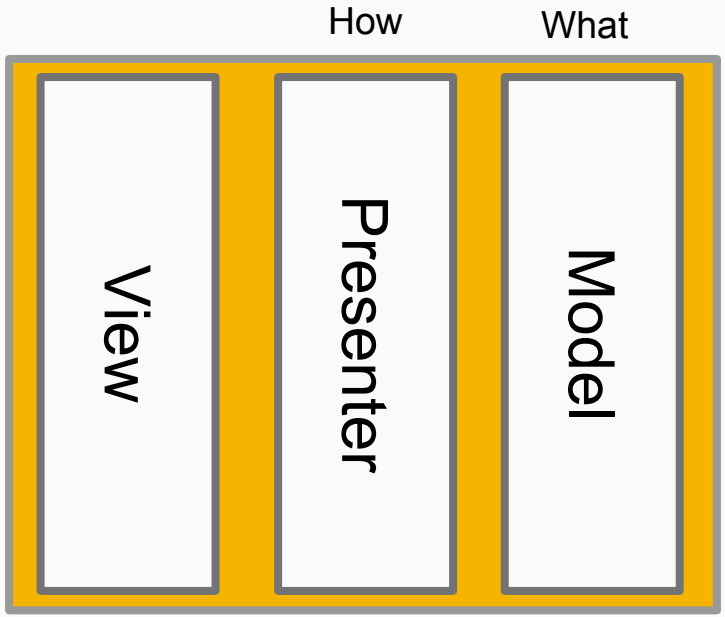
Robot Testing Pattern



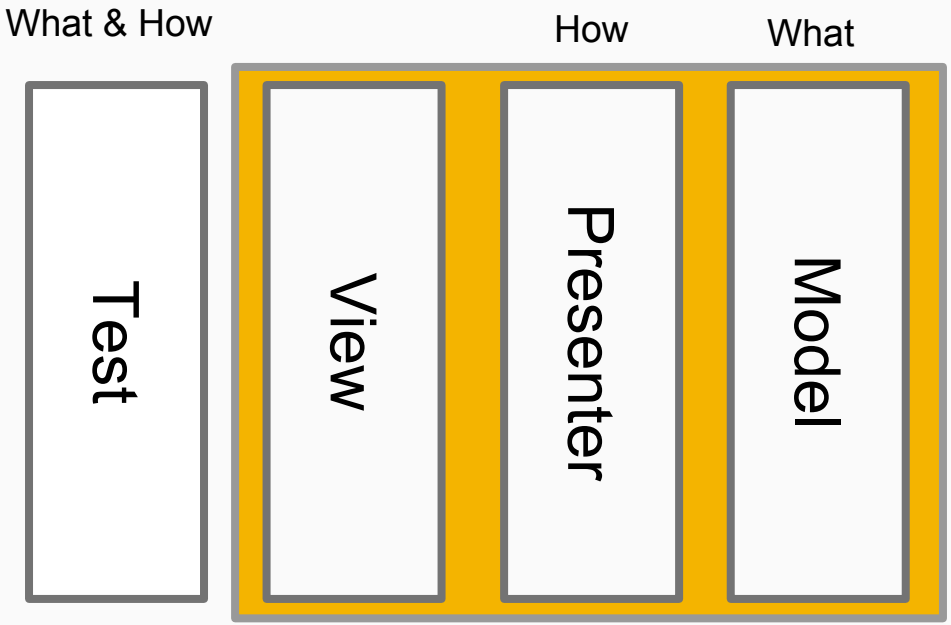
Robot Testing Pattern



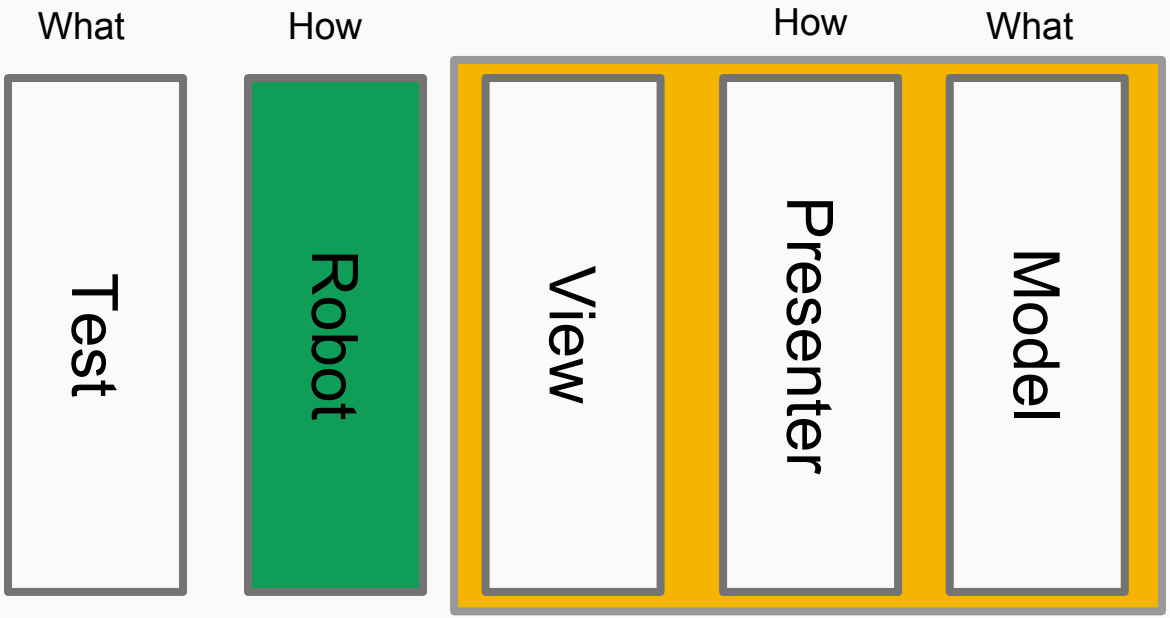
Robot Testing Pattern



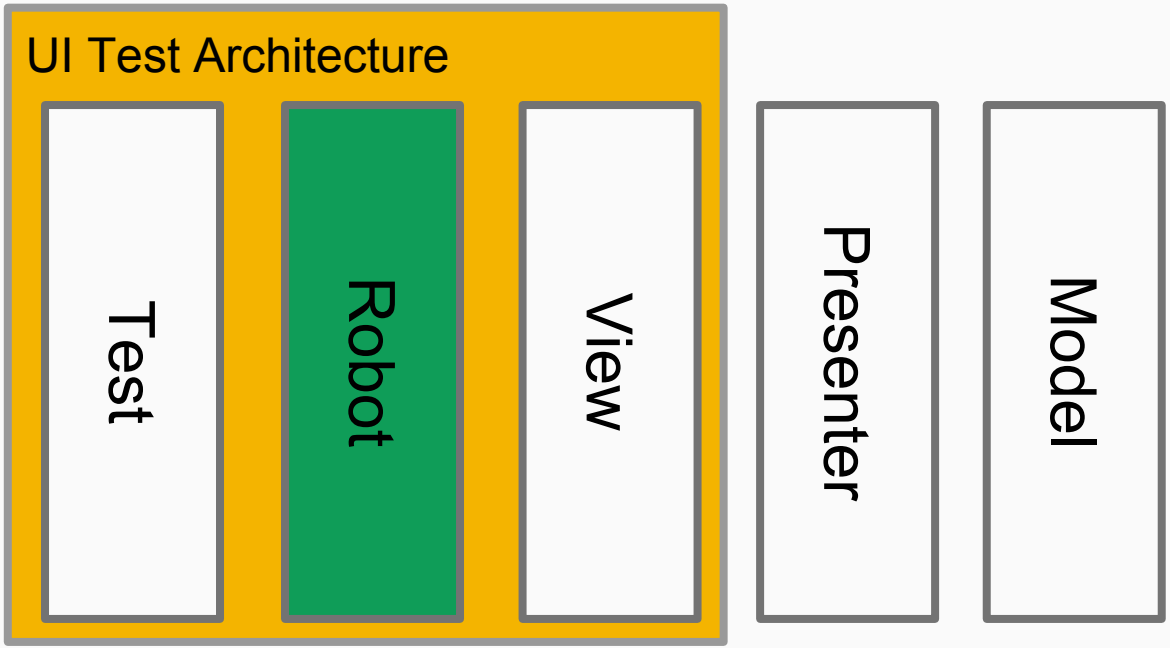
Robot Testing Pattern



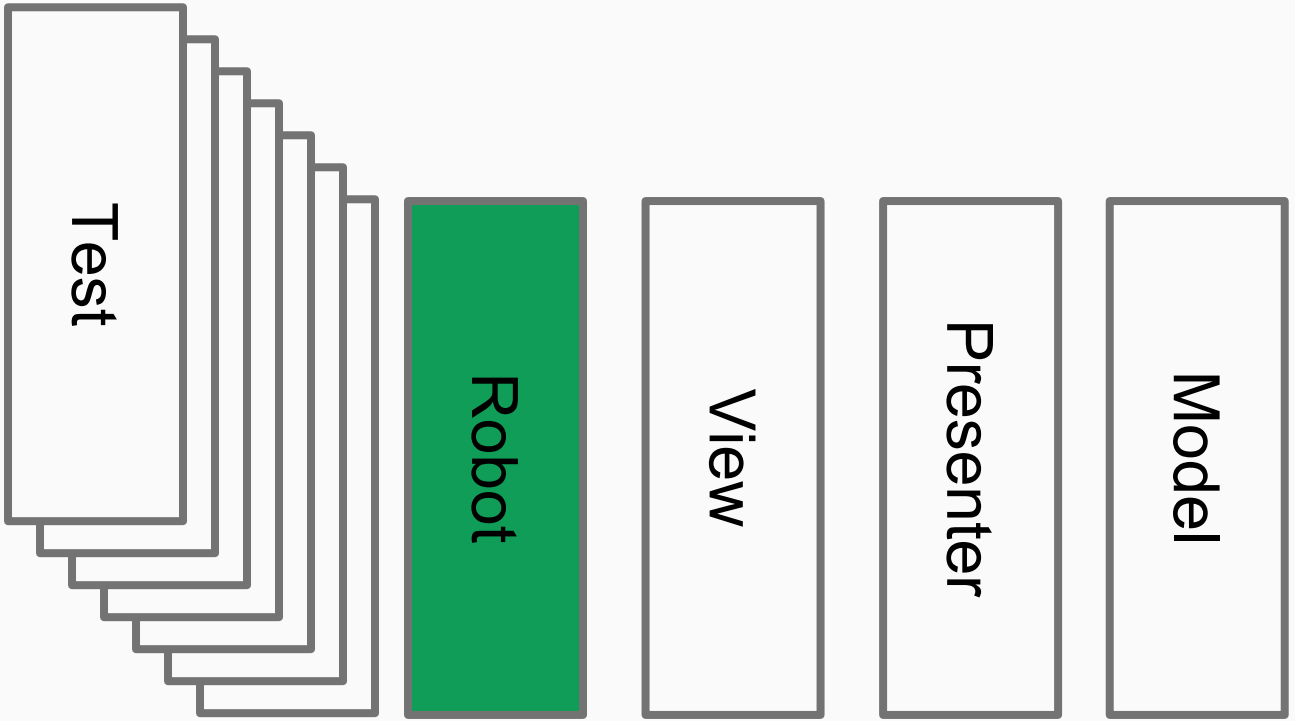
Robot Testing Pattern



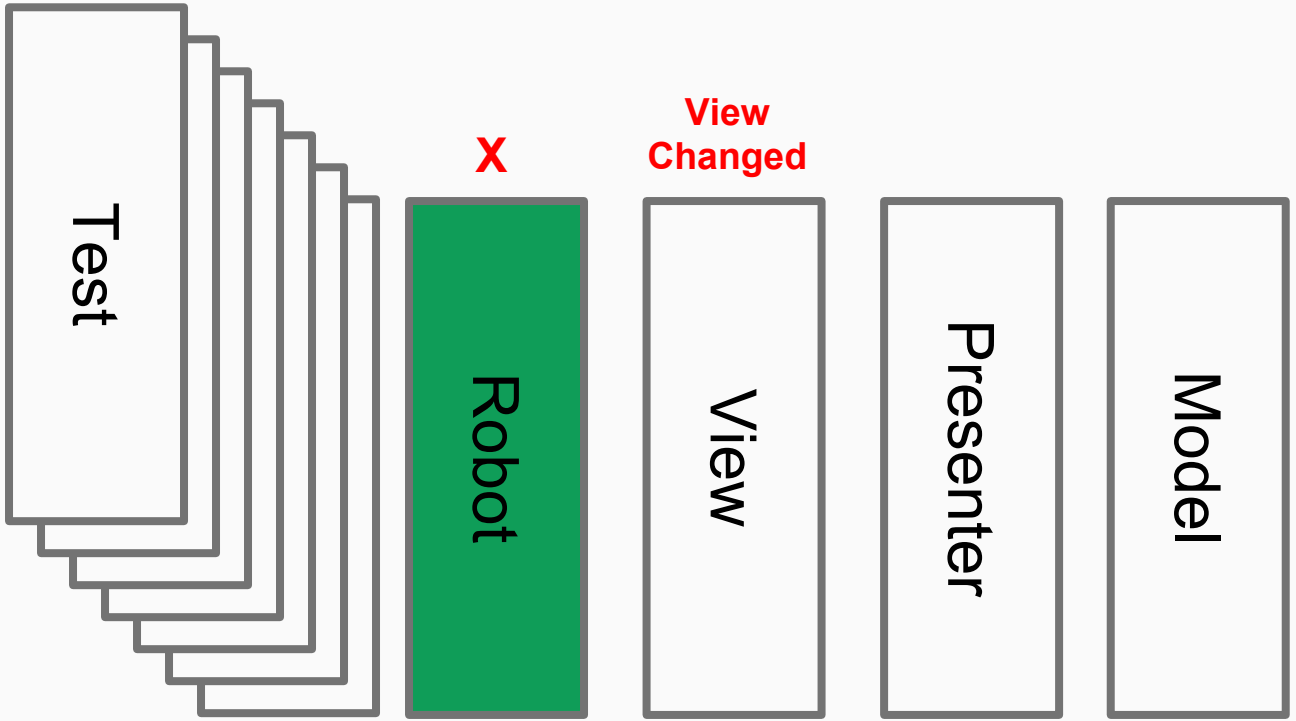
Robot Testing Pattern



Robot Testing Pattern



Robot Testing Pattern



Typical Espresso Test

```
@RunWith(AndroidJUnit4.class)
public class LoginTest {
   @Rule
   public ActivityTestRule<LoginActivity> activityRule = new ActivityTestRule(LoginActivity.class);

   @Test
   public void testLogin(){
      onView(withText("LOGIN")).check(matches(not(isEnabled())));
      onView(withId(R.id.username)).perform(typeText("sam"));
      onView(withText("LOGIN")).perform(click());
      onView(withId(R.id.home_view_pager)).check(matches(isDisplayed()));
   }
}
```

Typical Espresso Test With Screenshots

```
@RunWith(AndroidJUnit4.class)
public class LoginTest {
  @Rule
  public ActivityTestRule<LoginActivity> activityRule = new ActivityTestRule(LoginActivity.class);

  @Test
  public void testLogin(){
      onView(withText("LOGIN")).check(matches(not(isEnabled())));
      takeScreenshot("login_disabled");
      onView(withId(R.id.username)).perform(typeText("sam"));
      takeScreenshot("entered_username");
      onView(withText("LOGIN")).perform(click());
      takeScreenshot("logged_in_and_at_homescreen");
      onView(withId(R.id.home_view_pager)).check(matches(isDisplayed()));
  }
}
```

Espresso Test With a Robot

```
@RunWith(AndroidJUnit4.class)
public class LoginTest {
   @Rule
   public ActivityTestRule<LoginActivity> activityRule = new ActivityTestRule(LoginActivity.class);

   @Test
   public void testLogin(){
      new LoginRobot().assertLoginDisabled().username("sam").login().assertHomeScreenShown();
   }
}
```

Example Robot

```
static final Matcher<View> VIEW_MATCHER_USERNAME_EDIT_TEXT = withId(R.id.username);
static final Matcher<View> VIEW_MATCHER_LOGIN_BUTTON = withText("LOGIN");
static final Matcher<View> VIEW_MATCHER_HOME_SCREEN = withId(R.id.home_view_pager);

public LoginRobot assertLoginDisabled() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(not(isEnabled())));
  takeScreenshot("login_disabled");
  return this;
}

public LoginRobot username(String username) {
  onView(VIEW_MATCHER_USERNAME_EDIT_TEXT).perform(clearText(), typeText(username), closeSoftKeyboard());
  takeScreenshot("username_entered");
  return this;
}

public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}

public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

Example Robot

```
static final Matcher<View> VIEW_MATCHER_USERNAME_EDIT_TEXT = withId(R.id.username);
static final Matcher<View> VIEW_MATCHER_LOGIN_BUTTON = withText("LOGIN");
static final Matcher<View> VIEW_MATCHER_HOME_SCREEN = withId(R.id.home_view_pager);

public LoginRobot assertLoginDisabled() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(not(isEnabled())));
  takeScreenshot("login_disabled");
  return this;
}

public LoginRobot username(String username) {
  onView(VIEW_MATCHER_USERNAME_EDIT_TEXT).perform(clearText(), typeText(username), closeSoftKeyboard());
  takeScreenshot("username_entered");
  return this;
}

public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}

public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

Example Robot

```
static final Matcher<View> VIEW_MATCHER_USERNAME_EDIT_TEXT = withId(R.id.username);
static final Matcher<View> VIEW_MATCHER_LOGIN_BUTTON = withText("LOGIN");
static final Matcher<View> VIEW_MATCHER_HOME_SCREEN = withId(R.id.home_view_pager);

public LoginRobot assertLoginDisabled() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(not(isEnabled())));
  takeScreenshot("login_disabled");
  return this;
}

public LoginRobot username(String username) {
  onView(VIEW_MATCHER_USERNAME_EDIT_TEXT).perform(clearText(), typeText(username), closeSoftKeyboard());
  takeScreenshot("username_entered");
  return this;
}

public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}

public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

Example Robot

```
static final Matcher<View> VIEW_MATCHER_USERNAME_EDIT_TEXT = withId(R.id.username);
static final Matcher<View> VIEW_MATCHER_LOGIN_BUTTON = withText("LOGIN");
static final Matcher<View> VIEW_MATCHER_HOME_SCREEN = withId(R.id.home_view_pager);

public LoginRobot assertLoginDisabled() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(not(isEnabled())));
  takeScreenshot("login_disabled");
  return this;
}

public LoginRobot username(String username) {
  onView(VIEW_MATCHER_USERNAME_EDIT_TEXT).perform(clearText(), typeText(username), closeSoftKeyboard());
  takeScreenshot("username_entered");
  return this;
}

public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}

public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

Example Robot

```
static final Matcher<View> VIEW_MATCHER_USERNAME_EDIT_TEXT = withId(R.id.username);
static final Matcher<View> VIEW_MATCHER_LOGIN_BUTTON = withText("LOGIN");
static final Matcher<View> VIEW_MATCHER_HOME_SCREEN = withId(R.id.home_view_pager);

public LoginRobot assertLoginDisabled() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(not(isEnabled())));
  takeScreenshot("login_disabled");
  return this;
}

public LoginRobot username(String username) {
  onView(VIEW_MATCHER_USERNAME_EDIT_TEXT).perform(clearText(), typeText(username), closeSoftKeyboard());
  takeScreenshot("username_entered");
  return this;
}

public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}

public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

When to Take Screenshots

- **Assertions**
  - Assertion(s)
  - SCREENSHOT

```
public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

- **Actions (Alters screen state)**
  - Displayed Assertion
  - SCREENSHOT
  - Action

```
public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}
```

When to Take Screenshots

- **Assertions**
  - Assertion(s)
  - SCREENSHOT

```
public LoginRobot assertHomeScreenShown() {
  onView(VIEW_MATCHER_HOME_SCREEN).check(matches(isDisplayed()));
  takeScreenshot("home_screen_shown");
  return this;
}
```

- **Actions (Alters screen state)**
  - Displayed Assertion
  - SCREENSHOT
  - Action

```
public LoginRobot login() {
  onView(VIEW_MATCHER_LOGIN_BUTTON).check(matches(isDisplayed()));
  takeScreenshot("logging_in");
  onView(VIEW_MATCHER_LOGIN_BUTTON).perform(click());
  return this;
}
```

Tips and Tricks

NEVER Thread.sleep()

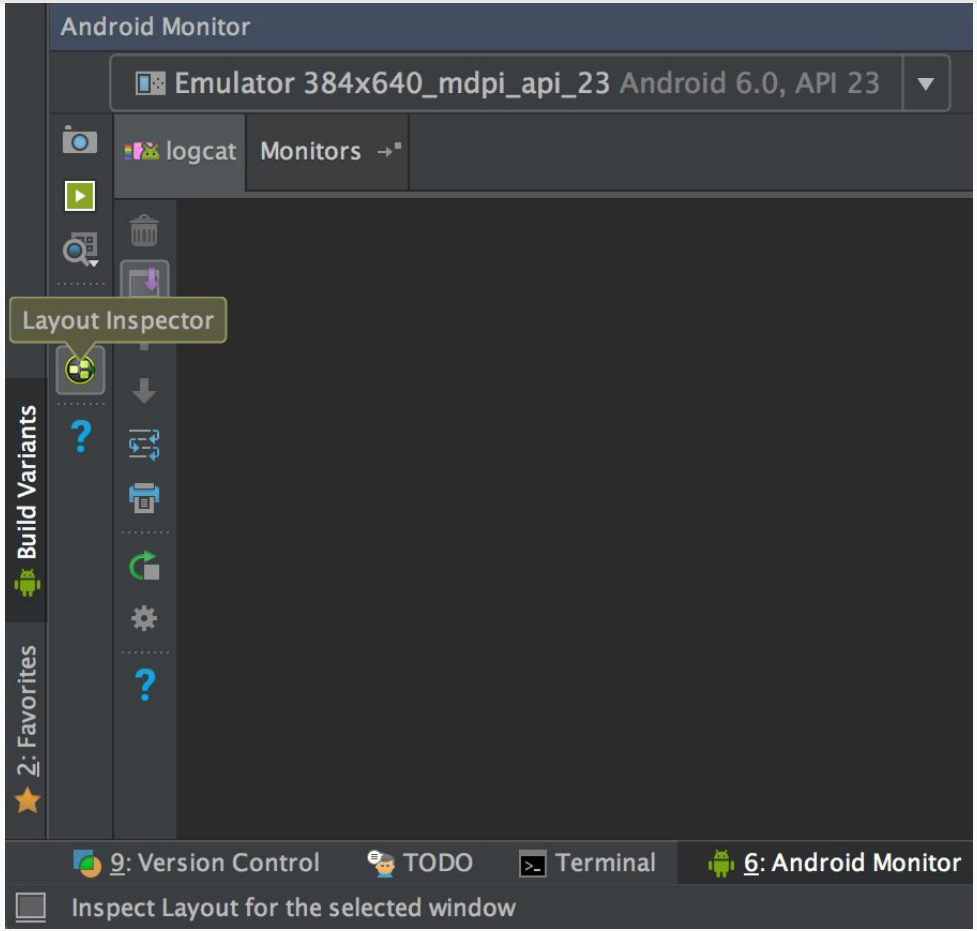
- AsyncTask Threadpool
  - Leverage for Networking and RxJava
- Idling Resources - Use CountingIdlingResource
  - increment(); and decrement();

https://developer.android.com/reference/android/support/test/espresso/contrib/CountingIdlingResource.html

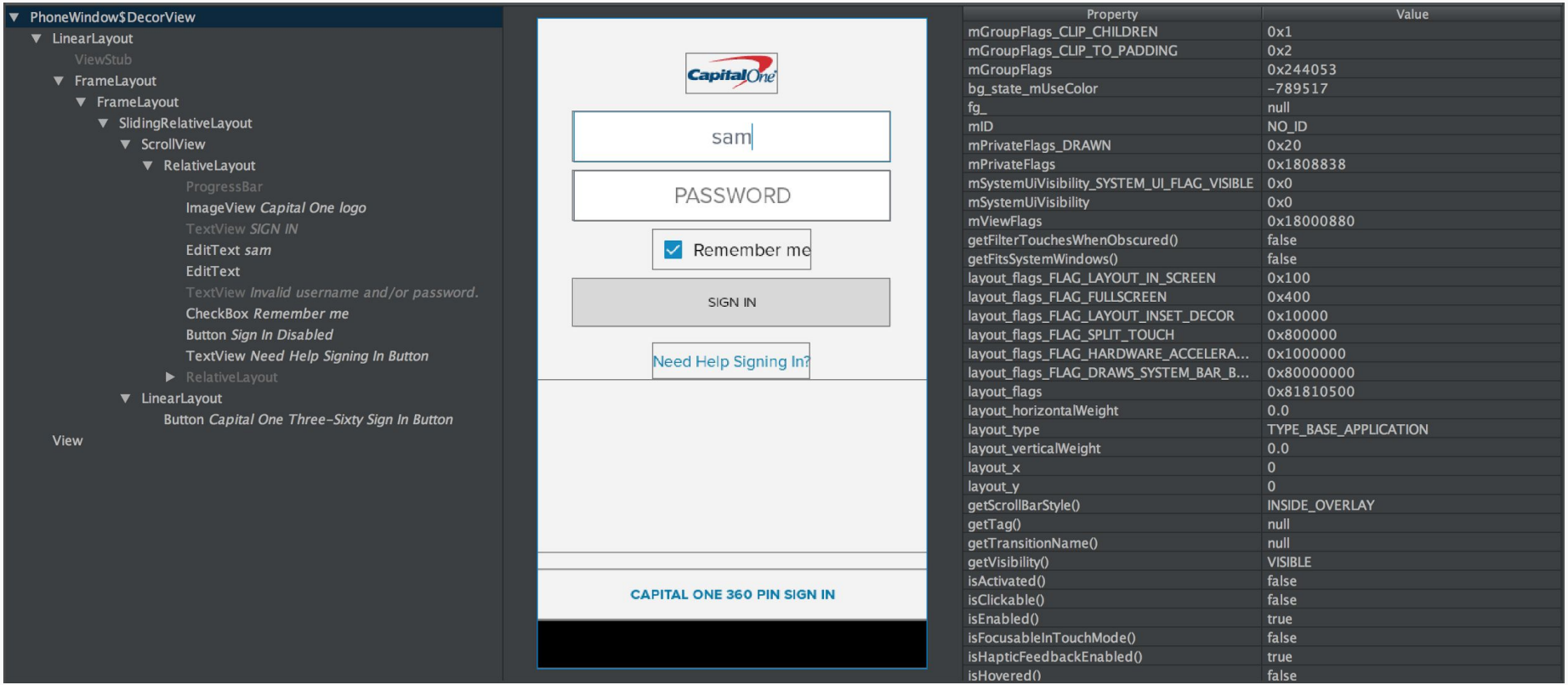
Define Your “LOW BAR” Device

- Android Emulator Virtual Device
- 384x640
- mdpi - 160 dots per inch
- Able to run in continuous integration

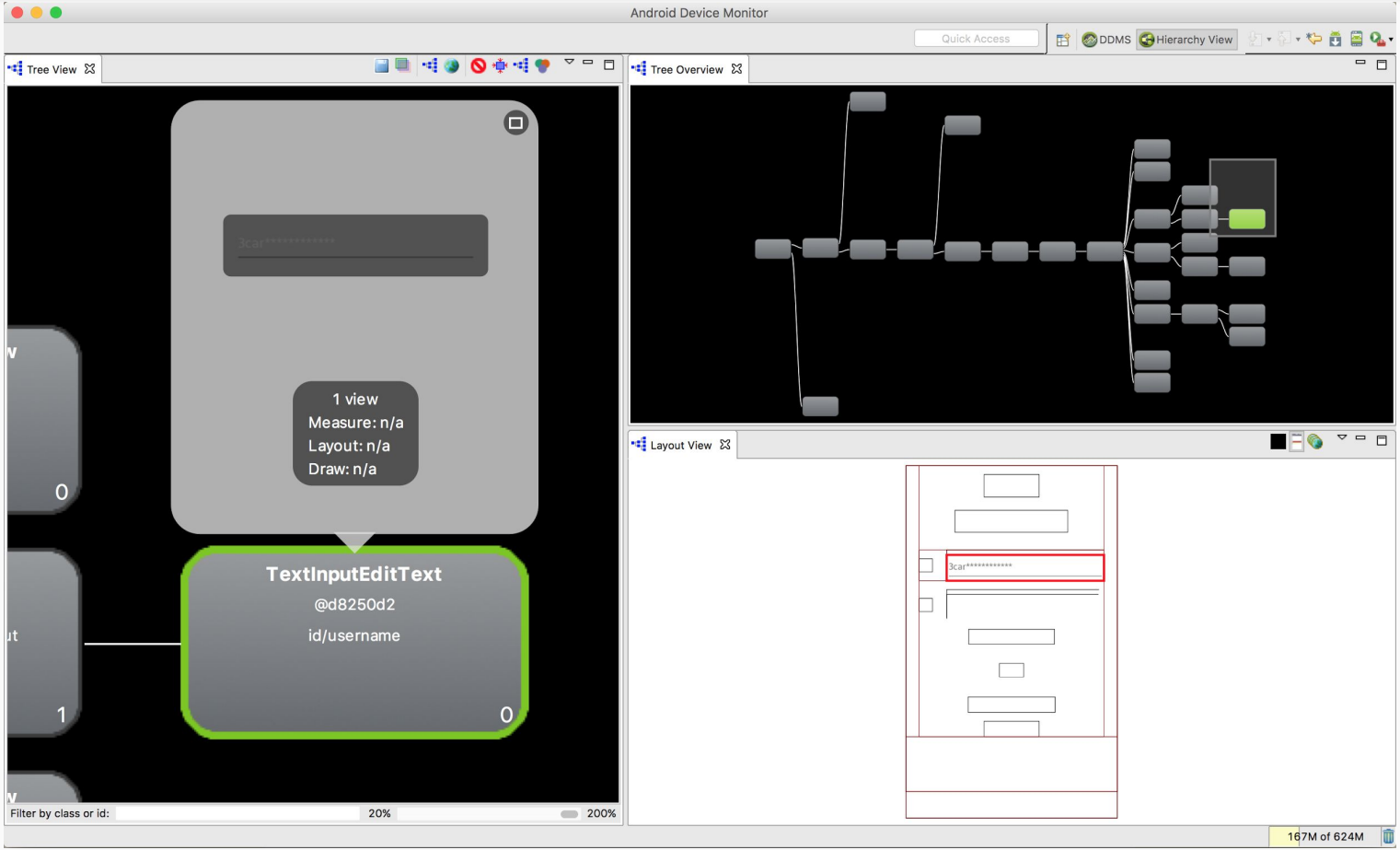
Android Studio - Layout Inspector



Android Studio - Layout Inspector



Android SDK Hierarchy Viewer



Leverage Continuous Integration

- Jenkins
- Circle CI
- BuddyBuild
- etc

Create a Custom Test Runner

- Control test runner behavior without recompiling
- Turn on/off all screenshots with a command-line parameter
  - --e screenshots=true
  - --e screenshots=false

Create a Custom Test Runner

```
defaultConfig {
   testInstrumentationRunner "com.handstandsam.MyAndroidJUnitRunner"
}
```

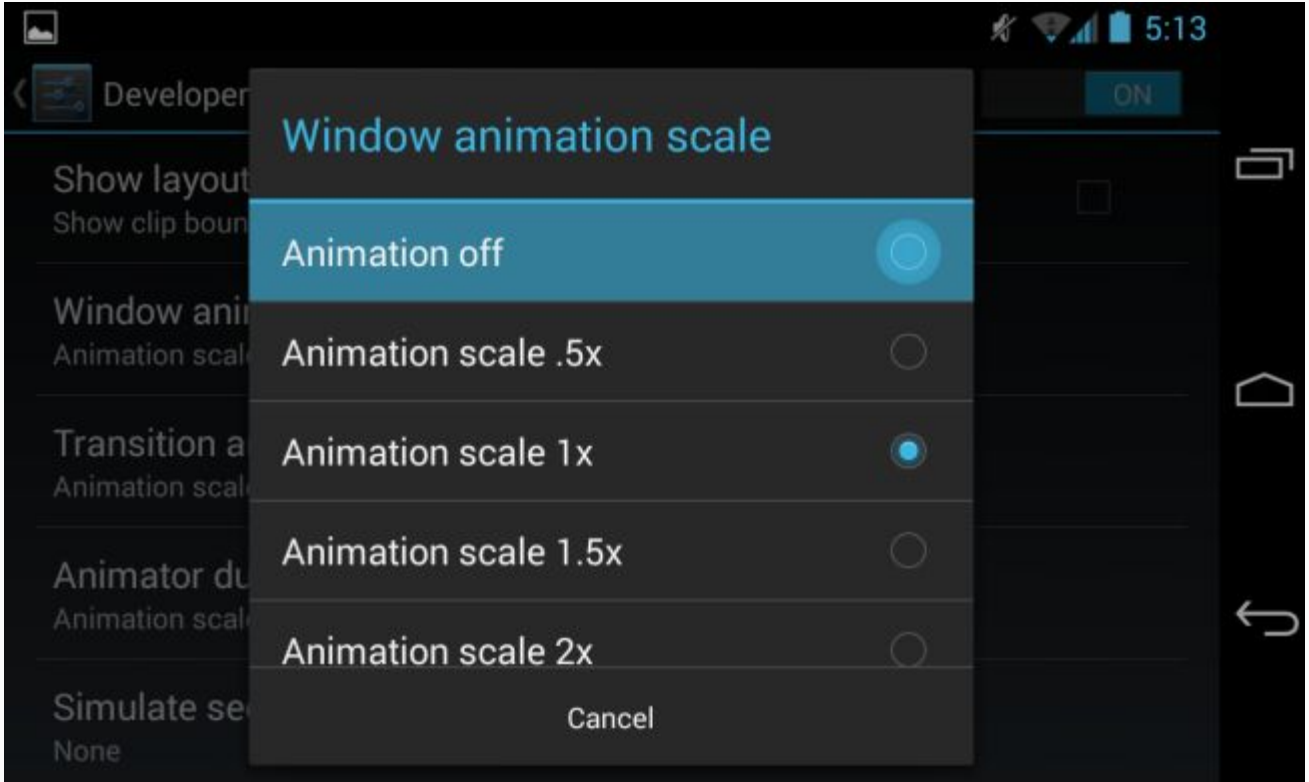
```
public class MyAndroidJUnitRunner extends AndroidJUnitRunner {
  @Override
  public void onCreate(Bundle arguments) {
    // process you parameters here.
    super.onCreate(arguments);
  }
}
```

Create a Custom Test Runner

```
defaultConfig {
    testInstrumentationRunner "com.handstandsam.MyAndroidJUnitRunner"
}
```

```
public class MyAndroidJUnitRunner extends AndroidJUnitRunner {
  @Override
  public void onCreate(Bundle arguments) {
    // process you parameters here.
    super.onCreate(arguments);
  }
}
```

Disable Animations



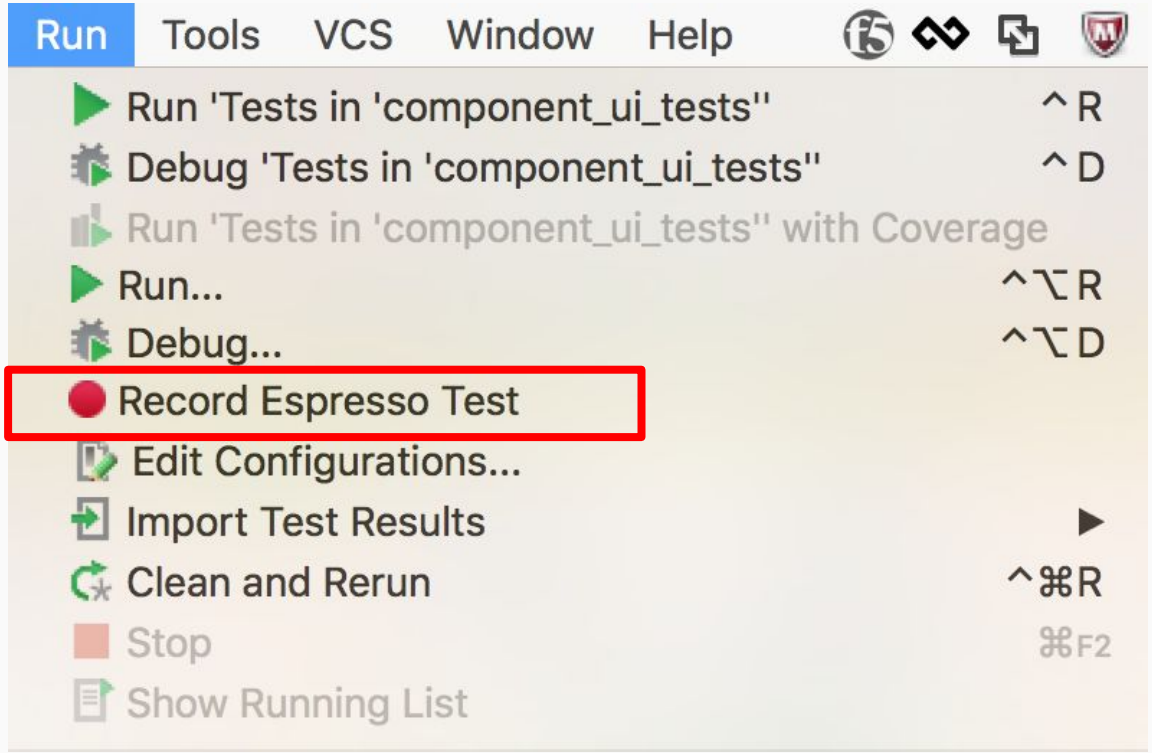
Test Butler by LinkedIn
“Reliable Android testing, at your service.”

- Stabilizes the Android emulator
  - Disables animations
  - Disables crash & ANR dialogs
  - Locks the keyguard, WiFi radio, and CPU to ensure they don't go to sleep unexpectedly while tests are running.
- Handles changing global emulator settings and holds relevant permissions so your app doesn't have to.
  - Enable/disable WiFi
  - Change device orientation
  - Set location services mode
  - Set application locale

https://github.com/linkedin/test-butler

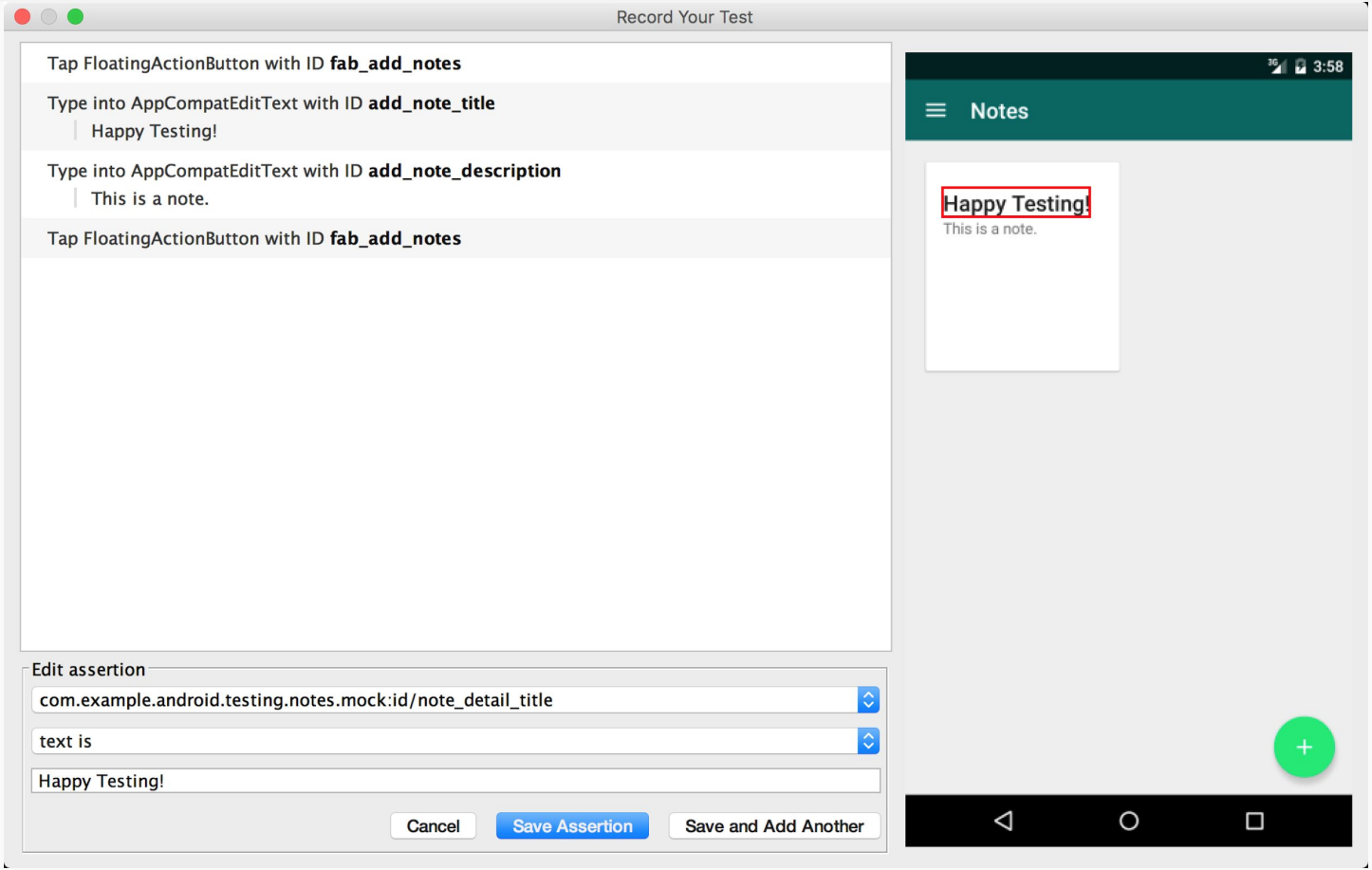
How do I get started?

Espresso Test Recorder in Android Studio 2.2+

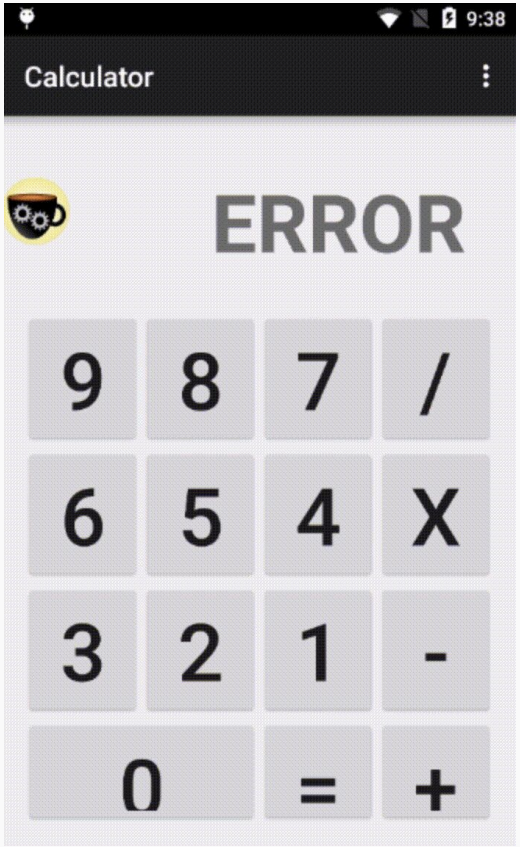


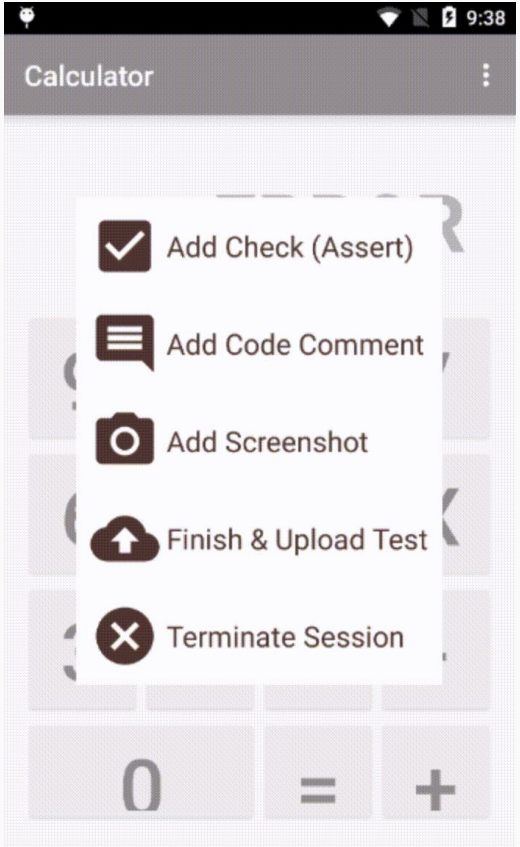
https://developer.android.com/studio/test/espresso-test-recorder.html

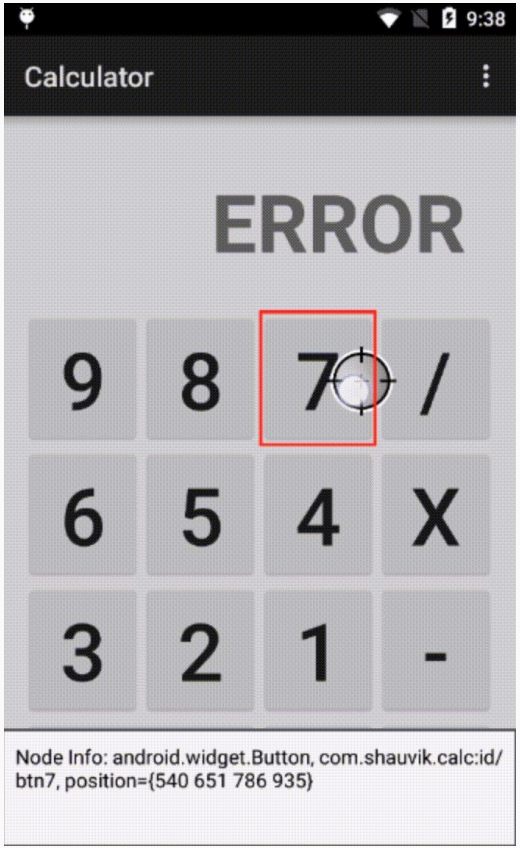
Espresso Test Recorder in Android Studio 2.2+



Barista by MoQuality



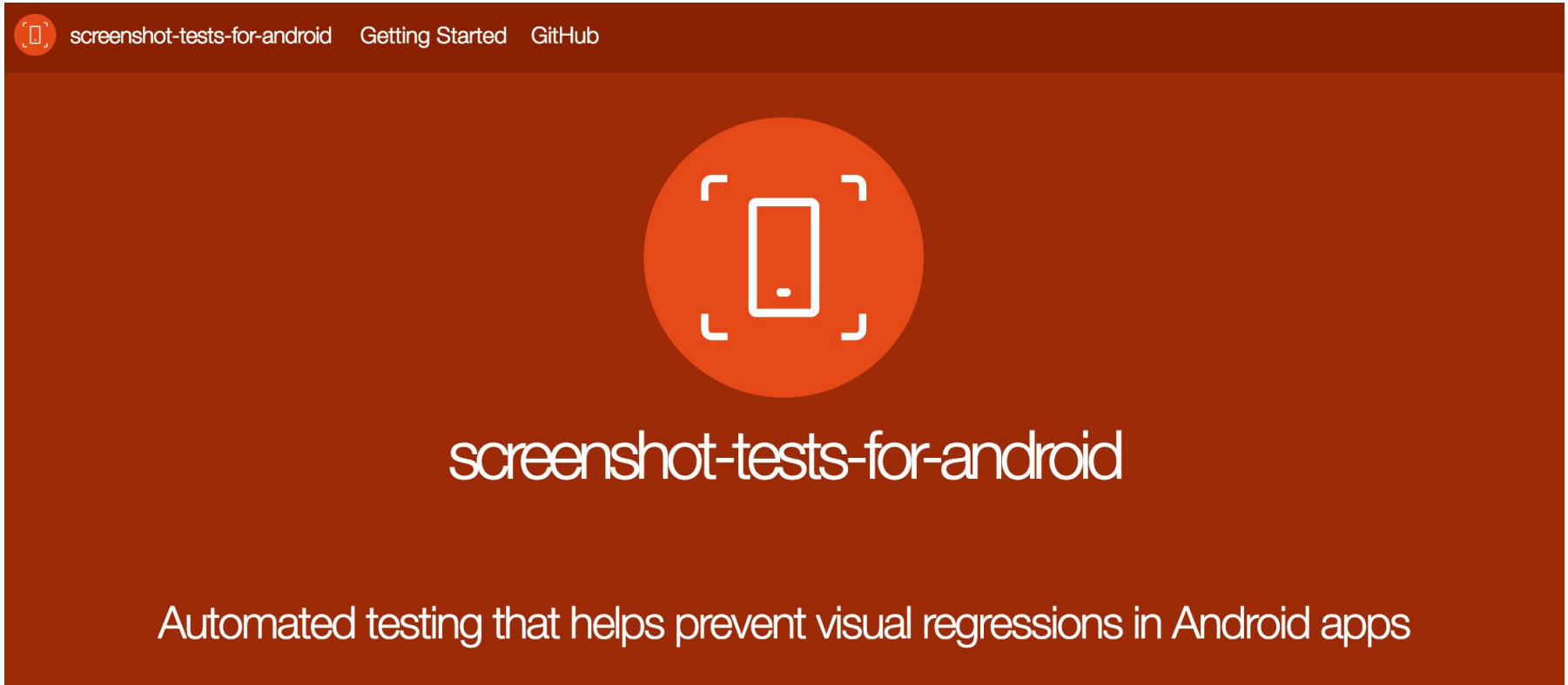




https://moquality.com/barista/

Pixel by Pixel
Visual Regression Testing

Deterministic View Screenshot Comparison



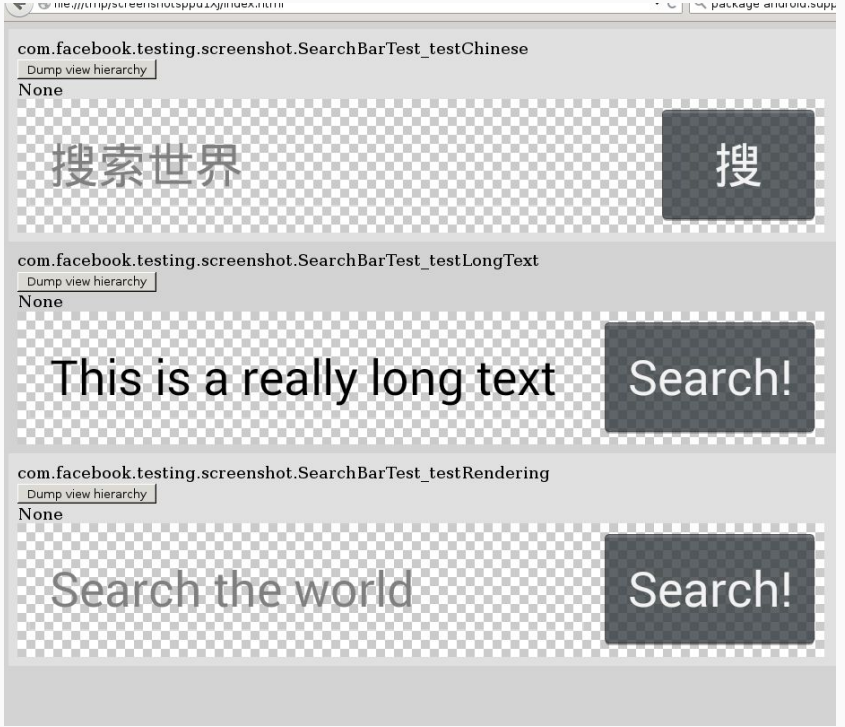
Screenshot-tests-for-android by Facebook

```
public class MyTests {
  @Test
  public void doScreenshot() {

    View view =
        mLayoutInflater.inflate(R.layout.my_layout, null, false);

    ViewHelpers.setupView(view)
      .setExactWidthDp(300)
      .layout();

    Screenshot.snap(view)
      .record();
  }
}
```



Sam Edwards

@HandstandSam