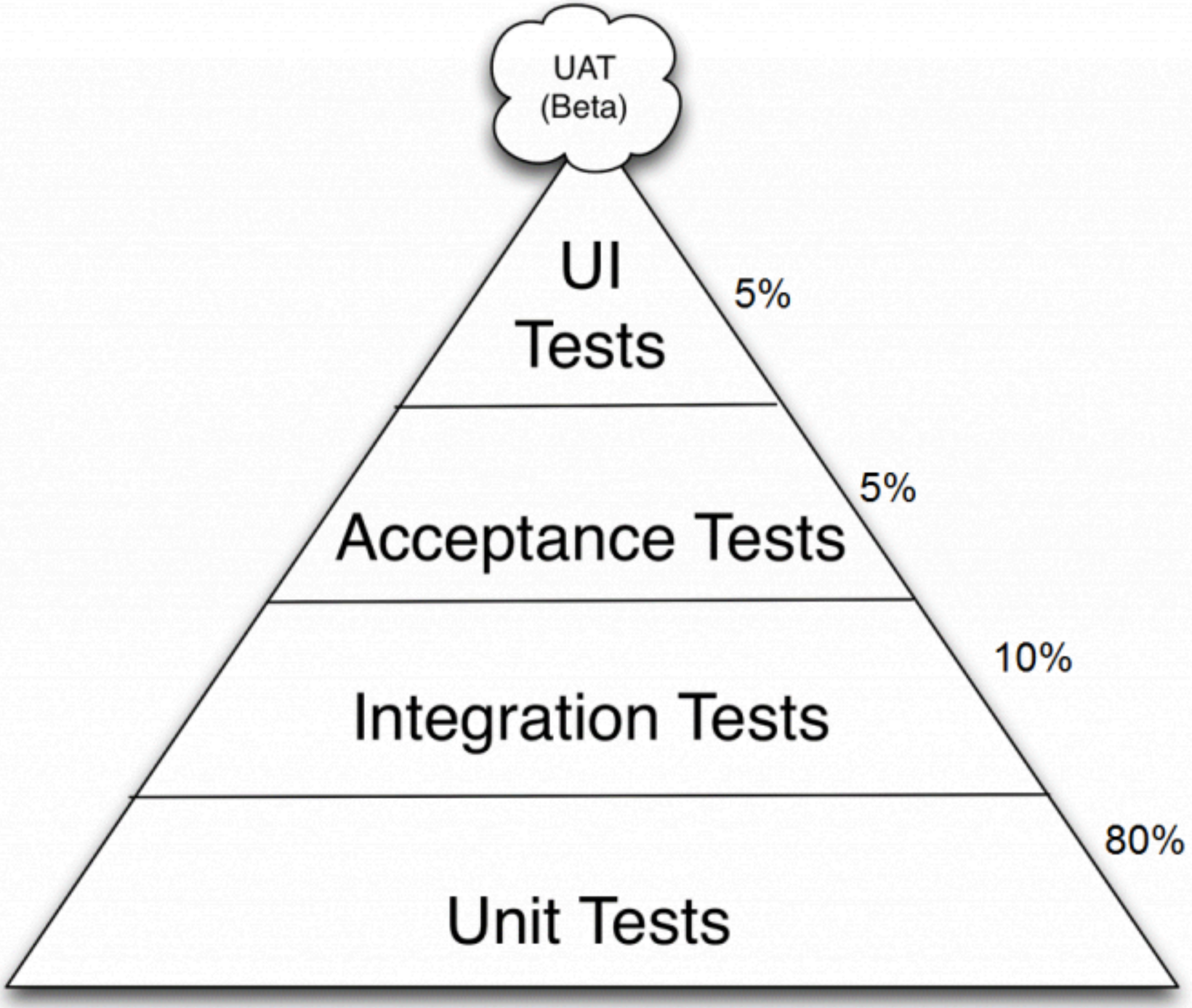
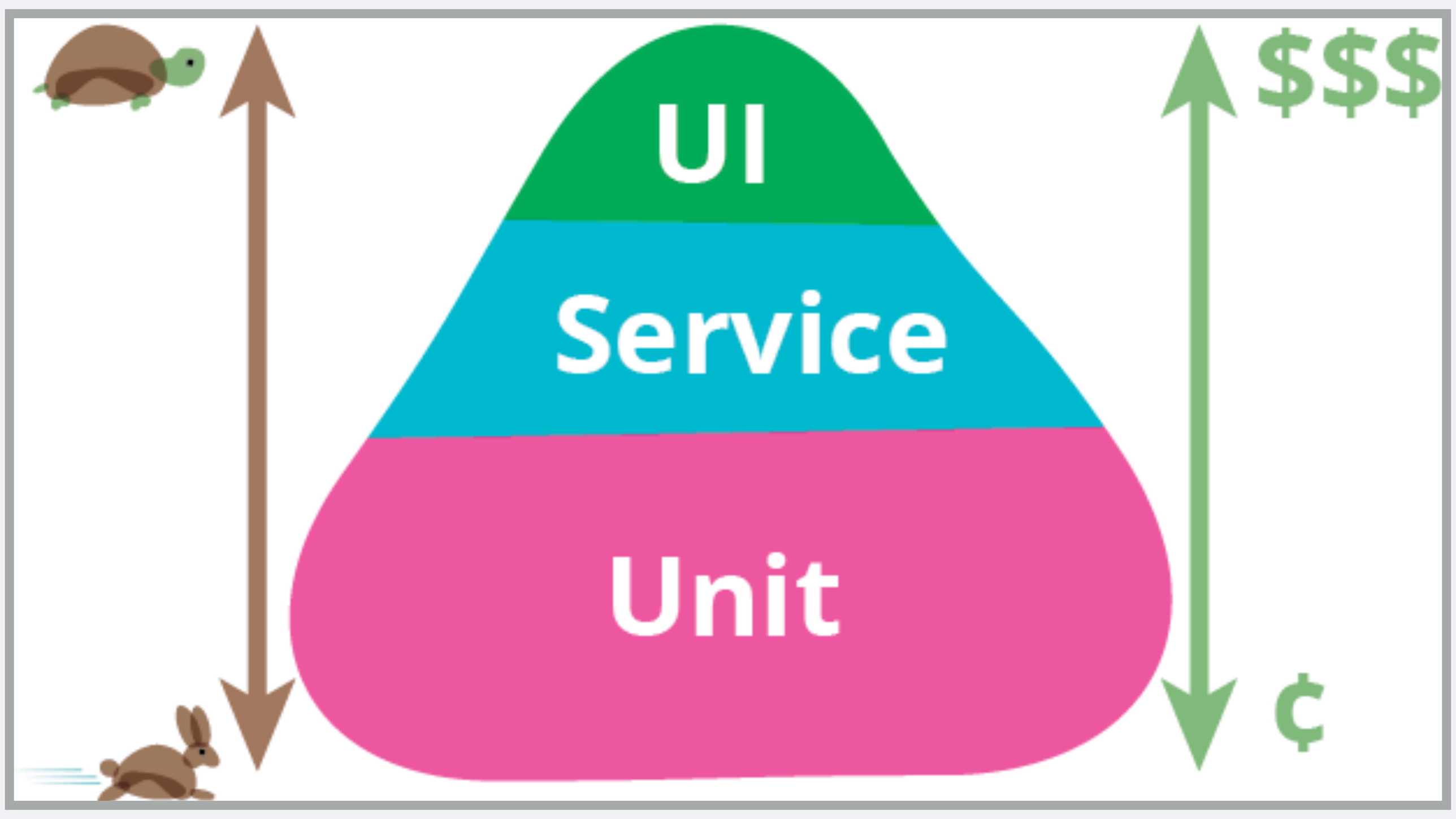
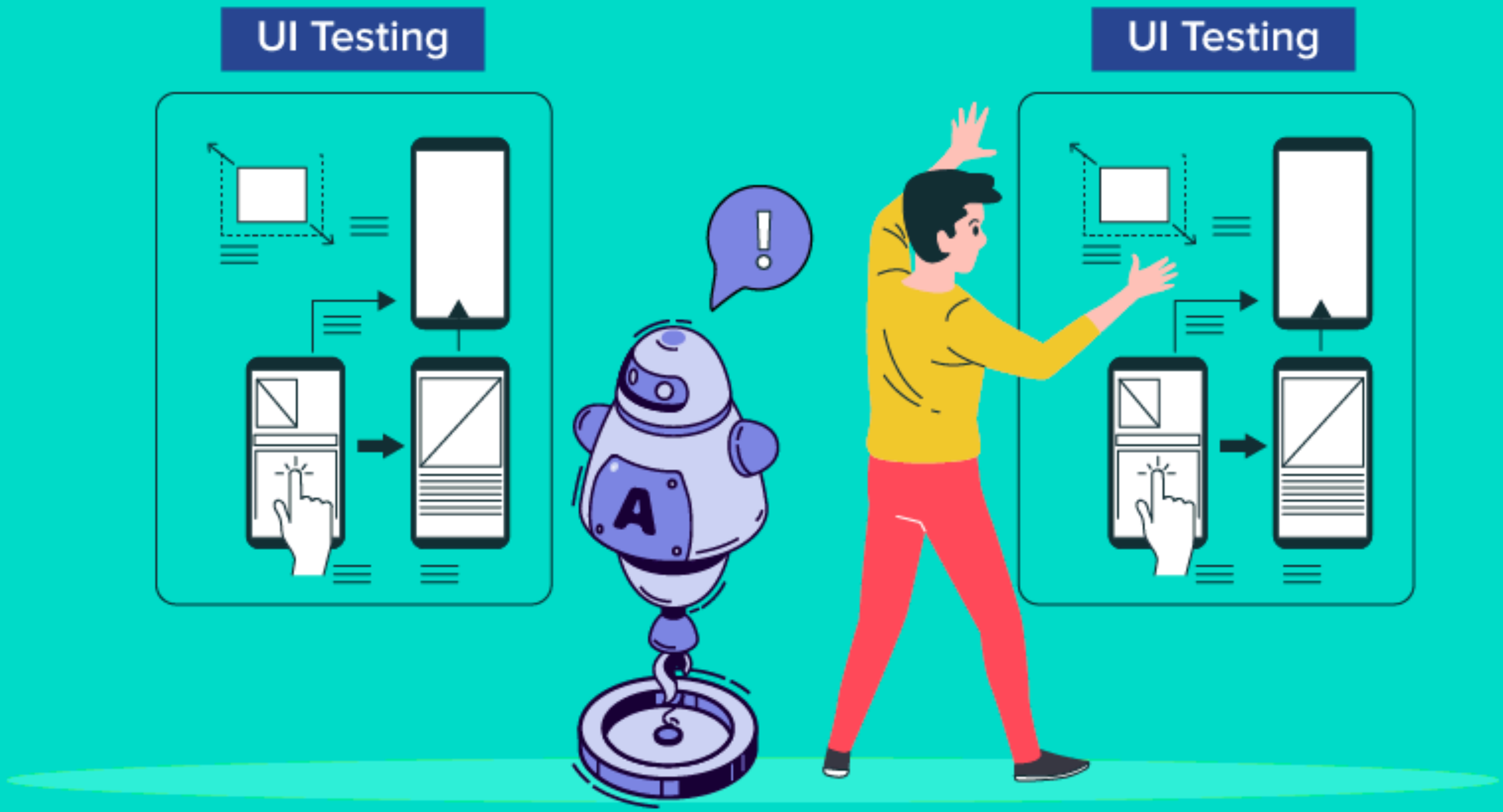
UI testing


UAT
(Beta)
UI
Tests
5%
Acceptance Tests
5%
Integration Tests
10%
Unit Tests
80%


UI
Service
Unit
$$$
¢

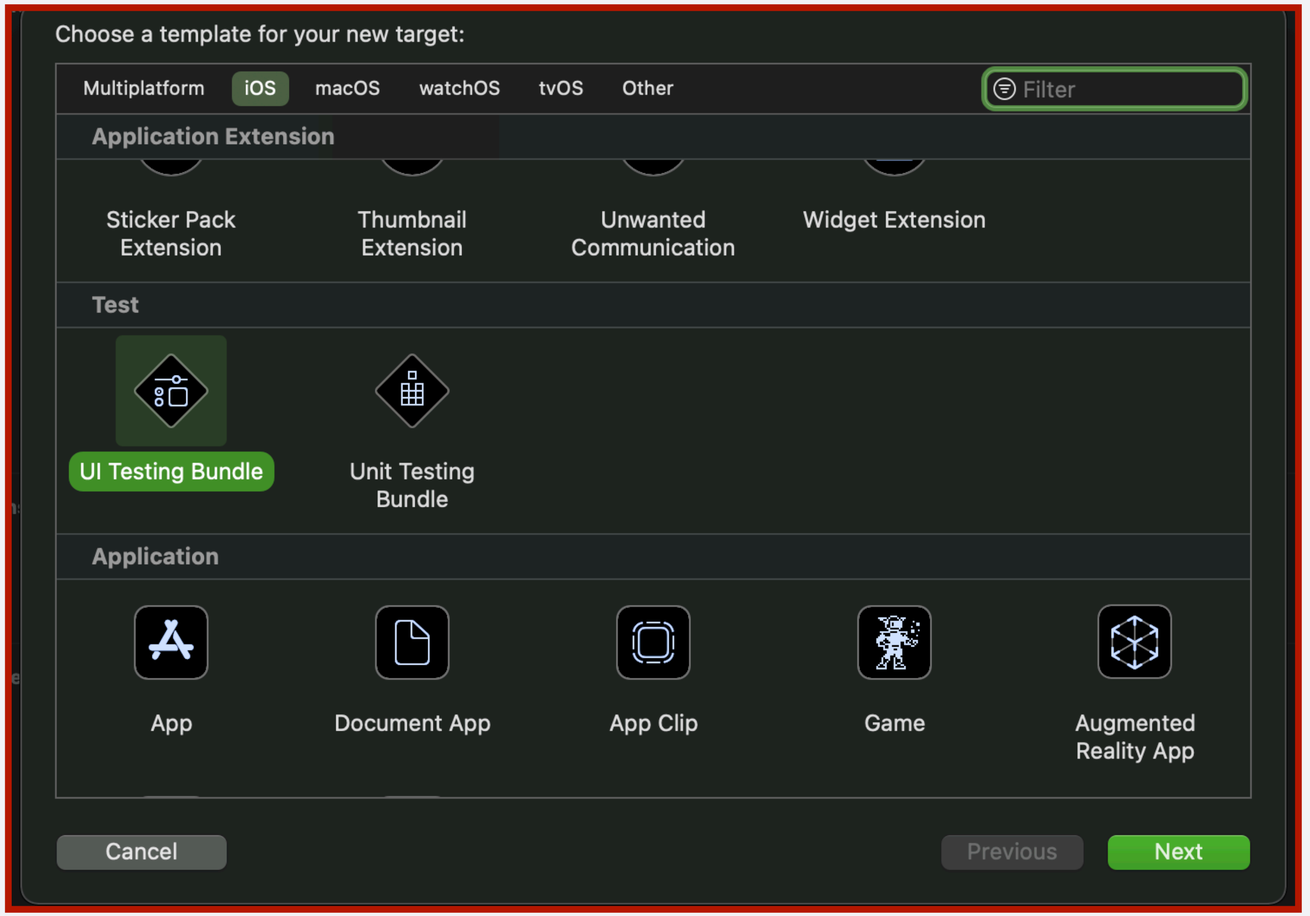
UI vs Unit

1. Хрупкие
2. Медленней пишутся
3. Медленней дают обратную связь


UI Testing
UI Testing
!
A

UI tests

- XCTest
- Accessibility



UI tests

- XCUI Application
- XCUIElement
- XCUIElementQuery

UI tests. XCUIElement

```swift
/**
 * @class XCUIElement (/seealso XCUIElementAttributes)
 * Elements are objects encapsulating the information needed to dynamically locate a user interface
 * element in an application. Elements are described in terms of queries /seealso XCUIElementQuery.
 */
open class XCUIElement : NSObject, XCUIElementAttributes, XCUIElementTypeQueryProvider {


    /** Test to determine if the element exists. */
    open var exists: Bool { get }


    /** Waits the specified amount of time for the element's exist property to be true and returns false if the timeout expires witho
        the element coming into existence. */
    open func waitForExistence(timeout: TimeInterval) -> Bool


    /** Whether or not a hit point can be computed for the element for the purpose of synthesizing events. */
    open var isHittable: Bool { get }


    /** Returns a query for all descendants of the element matching the specified type. */
    open func descendants(matching type: XCUIElement.ElementType) -> XCUIElementQuery


    /** Returns a query for direct children of the element matching the specified type. */
    open func children(matching type: XCUIElement.ElementType) -> XCUIElementQuery


    /** Creates and returns a new coordinate that will compute its screen point by adding the offset multiplied by the size of the
        element's frame to the origin of the element's frame. */
    open func coordinate(withNormalizedOffset normalizedOffset: CGVector) -> XCUICoordinate
```

Сбербанк Онлайн

Devteam

UI tests. XCUIElement

```
extension XCUIElement {


    /**
     * Sends a tap event to a hittable point computed for the element.
     */
    open func tap()


    /**
     * Sends a double tap event to a hittable point computed for the element.
     */
    open func doubleTap()


    /**
     * Sends a long press gesture to a hittable point computed for the element, holding for the specified duration.
     *
     * @param duration
     * Duration in seconds.
     */
    open func press(forDuration duration: TimeInterval)


    /**
     * Initiates a press-and-hold gesture that then drags to another element, suitable for table cell reordering and similar operations.
     * @param duration
     * Duration of the initial press-and-hold.
     * @param otherElement
     * The element to finish the drag gesture over. In the example of table cell reordering, this would be the reorder element of the
       destination row.
     */
    open func press(forDuration duration: TimeInterval, thenDragTo otherElement: XCUIElement)
```

Сбербанк Онлайн

Devteam

UI tests

```
final class UITests: XCTestCase {
    var app: XCUIApplication!

    override func setUp() {
        continueAfterFailure = false
        app = XCUIApplication()
        app.launchArguments = ["testing"]
        app.launch()
    }

    func testThatWelcomeMessageIsCorrect() {
        XCTAssertTrue(app.staticTexts["Hello World!"].exists)
    }
}
```

UI tests

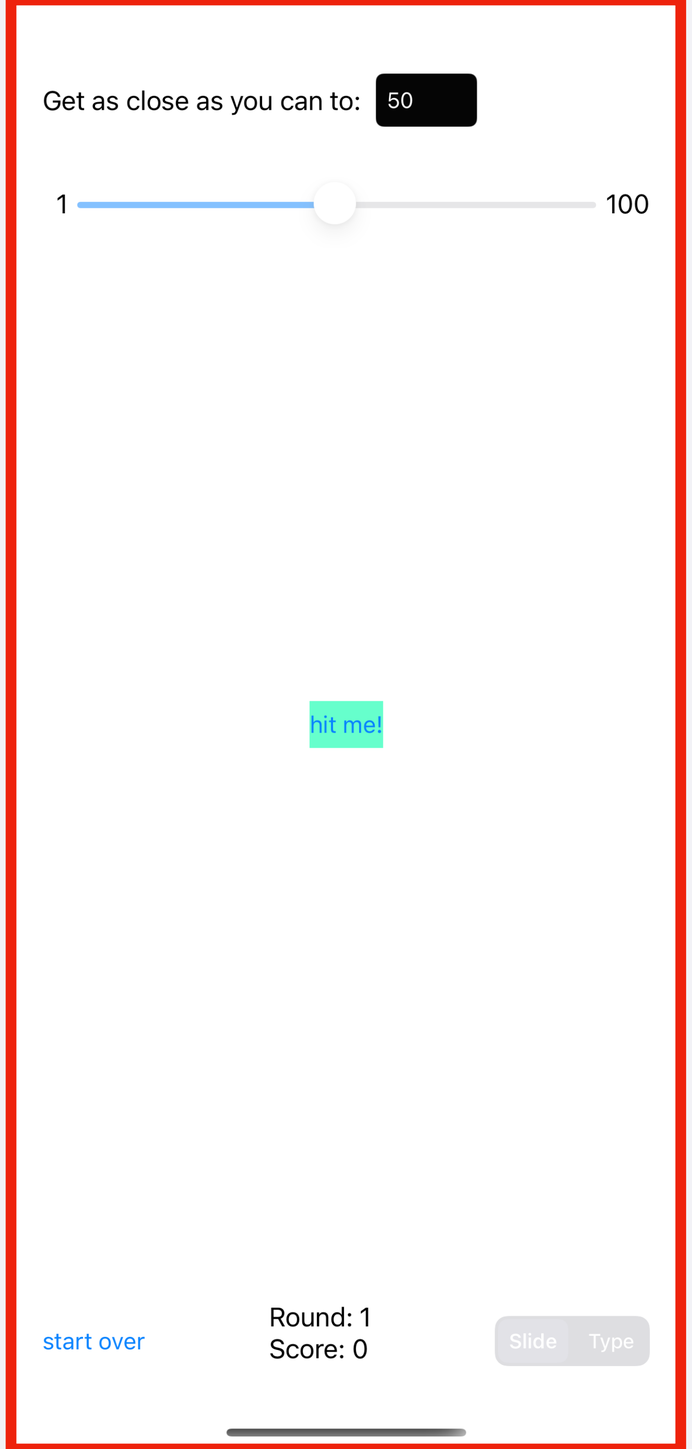
- XCUI Application
- XCUIElement
- XCUIElementQuery

- Stub manager
- Page object pattern
- Snapshot

Сбербанк Онлайн

Devteam

UI tests



Сбербанк Онлайн

Devteam

UI tests

```
var app: XCUIApplication!
```

```
class BullsEyeUITests: XCTestCase {
  var app: XCUIApplication!

  override func setUpWithError() throws {
    try super.setUpWithError()
    continueAfterFailure = false

    app = XCUIApplication()
    app.launch()
  }

  func testThatGameStyleSwitchChangesStateCorrectly() throws {

    // arrange
    let slideButton = app.segmentedControls.buttons["Slide"]
    let typeButton = app.segmentedControls.buttons["Type"]
    let slideLabel = app.staticTexts["Get as close as you can to: "]
    let typeLabel = app.staticTexts["Guess where the slider is: "]

    // assert
    if slideButton.isSelected {
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)

      typeButton.tap()
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)
    } else if typeButton.isSelected {
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)

      slideButton.tap()
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)
    }
  }
}
```

Сбербанк Онлайн

UI tests

```
/**
 * The most recently observed state of the application. Applications are passively monitored to update
 * this property as they change state. Consequently, updates to this property are inherently asynchronous.
 *
 * Some guarantees are made, however:
 *
 *     - When -launch and -activate return, if they were successful, the state of the application will be
 *       XCUIApplicationStateRunningBackground or XCUIApplicationStateRunningForeground, whichever is
 *       appropriate for the application. Most applications will be XCUIApplicationStateRunningForeground
 *       after launch or activation.
 *
 *     - When -terminate returns, if it was successful, the state of the application will be
 *       XCUIApplicationStateNotRunning.
 */
open var state: XCUIApplication.State { get }
```

Сбербанк Онлайн

Devteam

UI tests

```
/**
 * Waits for the application to become a specific state, giving up after a number of seconds.
 *
 * Returns YES if the application is currently in or transitions to the desired state within the specified
 * timeout period.
 */
open func wait(for state: XCUIApplication.State, timeout: TimeInterval) -> Bool
```

Сбербанк Онлайн

Devteam

UI tests

```
override func setUpWithError() throws {
  try super.setUpWithError()
  continueAfterFailure = false

  app = XCUIApplication()
  app.launch()
}
```

```
class BullsEyeUITests: XCTestCase {
  var app: XCUIApplication!

  override func setUpWithError() throws {
    try super.setUpWithError()
    continueAfterFailure = false

    app = XCUIApplication()
    app.launch()
  }

  func testThatGameStyleSwitchChangesStateCorrectly() throws {

    // arrange
    let slideButton = app.segmentedControls.buttons["Slide"]
    let typeButton = app.segmentedControls.buttons["Type"]
    let slideLabel = app.staticTexts["Get as close as you can to: "]
    let typeLabel = app.staticTexts["Guess where the slider is: "]

    // assert
    if slideButton.isSelected {
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)

      typeButton.tap()
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)
    } else if typeButton.isSelected {
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)

      slideButton.tap()
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)
    }
  }
}
```

Сбербанк Онлайн

UI tests

```
/**
 * @property continueAfterFailure
 * Determines whether the test method continues execution after an XCTAssert fails.
 *
 * By default, this property is YES, meaning the test method will complete regardless of how many
 * XCTAssert failures occur. Setting this to NO causes the test method to end execution immediately
 * after the first failure occurs, but does not affect remaining test methods in the suite.
 *
 * If XCTAssert failures in the test method indicate problems with state or determinism, additional
 * failures may be not be helpful information. Setting `continueAfterFailure` to NO can reduce the
 * noise in the test report for these kinds of tests.
 */
open var continueAfterFailure: Bool
```

Сбербанк Онлайн

Devteam

UI tests

```
func testThatGameStyleSwitchChangesStateCorrectly() throws {

  // arrange
  let slideButton = app.segmentedControls.buttons["Slide"]
  let typeButton = app.segmentedControls.buttons["Type"]
  let slideLabel = app.staticTexts["Get as close as you can to: "]
  let typeLabel = app.staticTexts["Guess where the slider is: "]
}
```

```
class BullsEyeUITests: XCTestCase {
  var app: XCUIApplication!

  override func setUpWithError() throws {
    try super.setUpWithError()
    continueAfterFailure = false

    app = XCUIApplication()
    app.launch()
  }

  func testThatGameStyleSwitchChangesSt     Correctly() throws {

    // arrange
    let slideButton = app.segmentedControls.buttons["Slide"]
    let typeButton = app.segmentedControls.buttons["Type"]
    let slideLabel = app.staticTexts["Get as close as you can to: "]
    let typeLabel = app.staticTexts["Guess where the slider is: "]

    // assert
    if slideButton.isSelected {
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)

      typeButton.tap()
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)
    } else if typeButton.isSelected {
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)

      slideButton.tap()
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)
    }
  }
}
```

Сбербанк Онлайн

UI tests. XCUIElementQuery

Сбербанк Онлайн

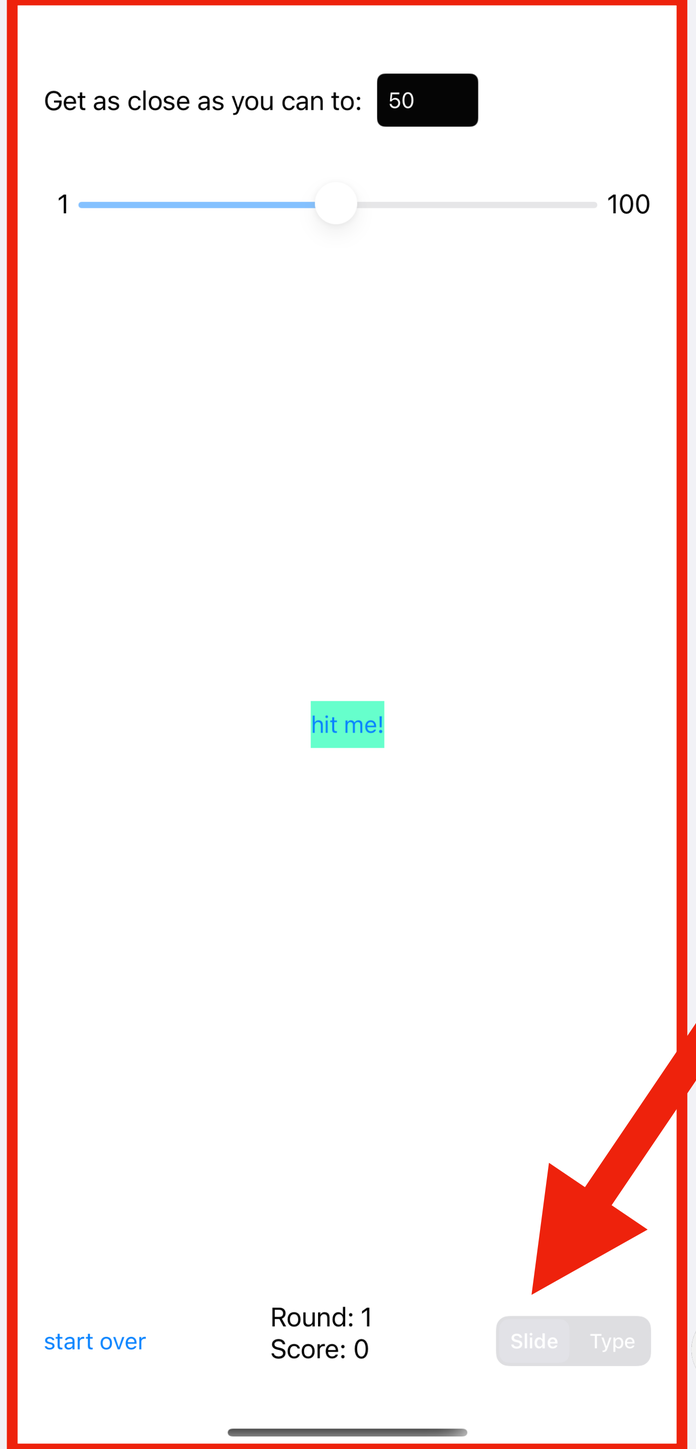
Devteam

UI tests

```
let app = XCUIApplication()
app.buttons["Slide"].tap()
```

UI tests

```
let slideButton = app.segmentedControls.buttons["Slide"]
```



Сбербанк Онлайн

Devteam

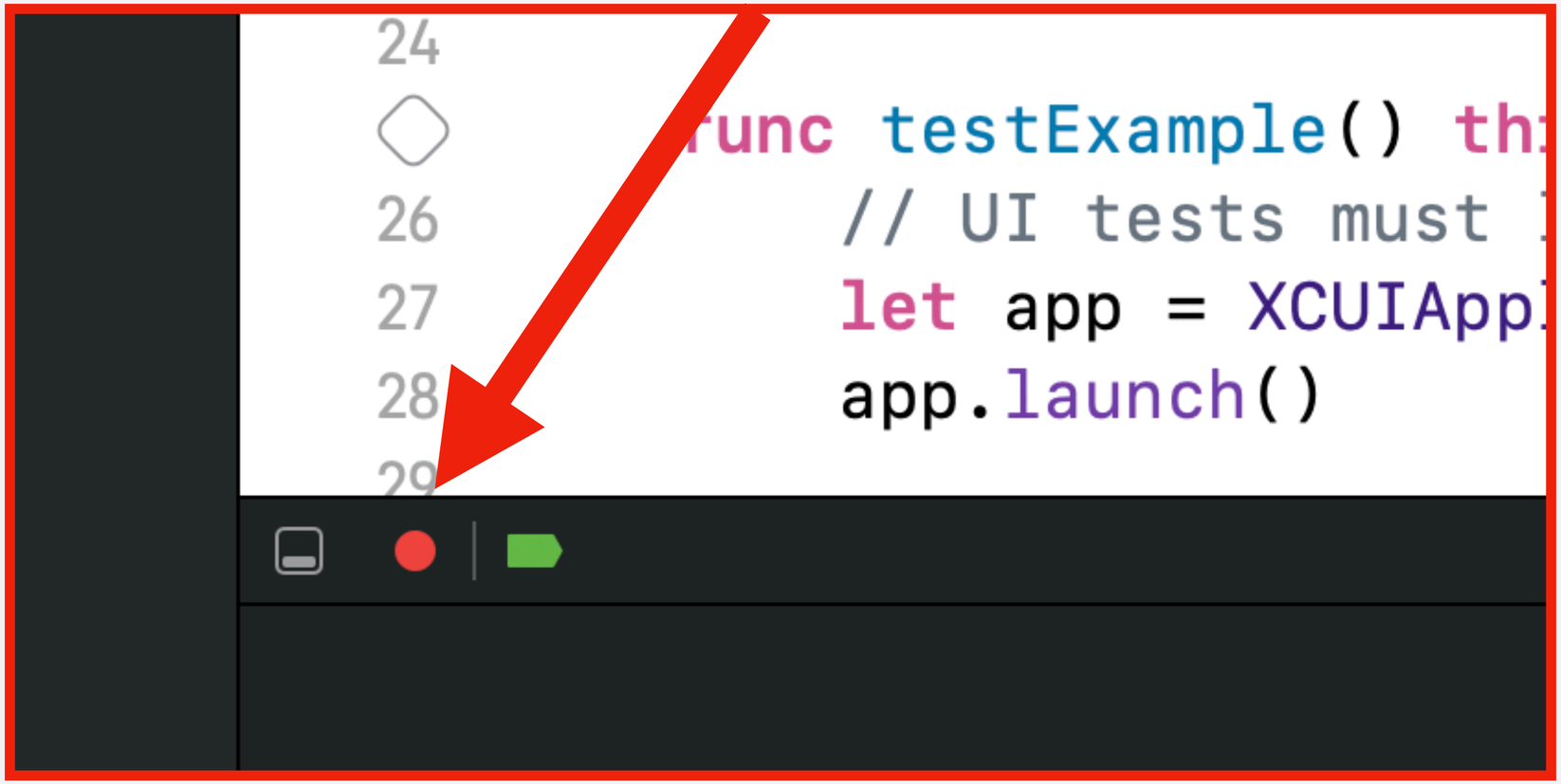
UI tests

```
if slideButton.isSelected {
  XCTAssertTrue(slideLabel.exists)
  XCTAssertFalse(typeLabel.exists)

  typeButton.tap()
  XCTAssertTrue(typeLabel.exists)
  XCTAssertFalse(slideLabel.exists)
} else if typeButton.isSelected {
  XCTAssertTrue(typeLabel.exists)
  XCTAssertFalse(slideLabel.exists)

  slideButton.tap()
  XCTAssertTrue(slideLabel.exists)
  XCTAssertFalse(typeLabel.exists)
}
```

Сбербанк Онлайн

```
class BullsEyeUITests: XCTestCase {
  var app: XCUIApplication!

  override func setUpWithError() throws {
    try super.setUpWithError()
    continueAfterFailure = false

    app = XCUIApplication()
    app.launch()
  }

  func testThatGameStyleSwitchChange tateCorrectly() throws {

    // arrange
    let slideButton = app.segmen edControls.buttons["Slide"]
    let typeButton = app.segme edControls.buttons["Type"]
    let slideLabel = app.stat cTexts["Get as close as you can to: "]
    let typeLabel = app.s cTexts["Guess where the slider is: "]

    // assert
    if slideButton.isSelected {
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)

      typeButton.tap()
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)
    } else if typeButton.isSelected {
      XCTAssertTrue(typeLabel.exists)
      XCTAssertFalse(slideLabel.exists)

      slideButton.tap()
      XCTAssertTrue(slideLabel.exists)
      XCTAssertFalse(typeLabel.exists)
    }
  }
}
```

UI tests



Сбербанк Онлайн

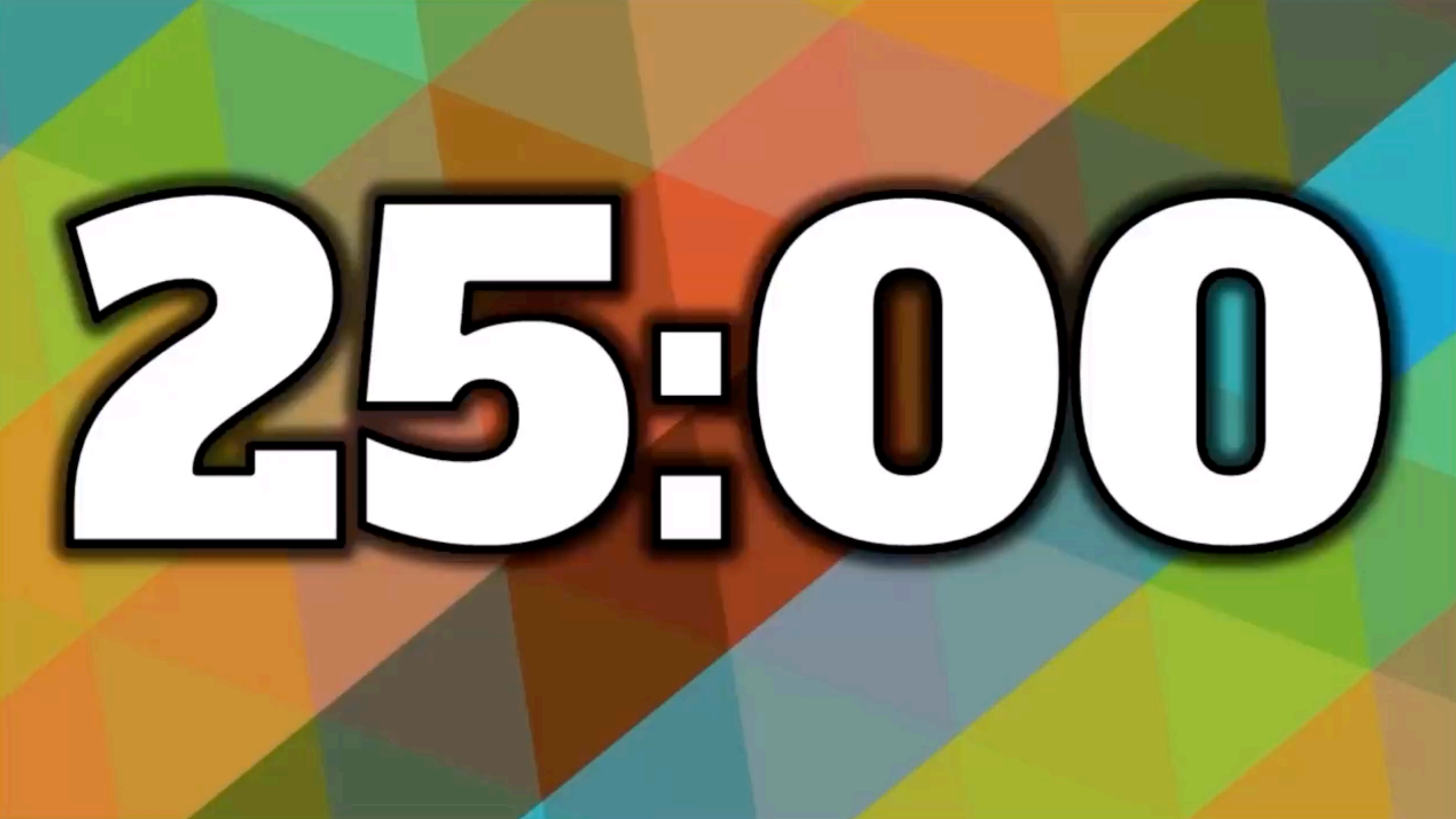
Devteam

ЗАДАНИЕ

Реализовать какой-нибудь UI-тест (индивидуально)

Сбербанк Онлайн

Devteam


25:00

UI tests

- XCUI Application
- XCUIElement
- XCUIElementQuery

- Stub manager
- Page object pattern
- Snapshot

Сбербанк Онлайн

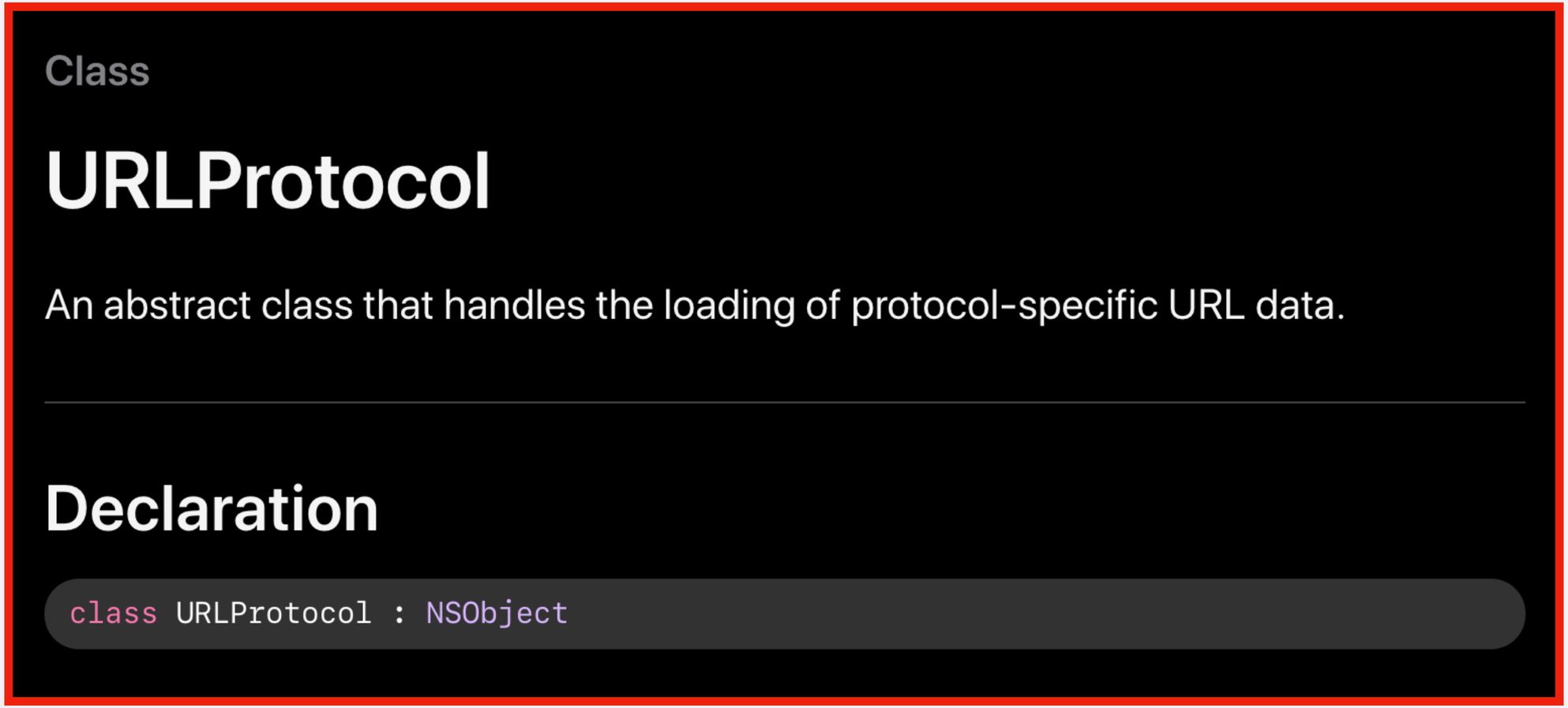
Devteam


10:00

UI tests. Stub manager

- создание класса-наследника NSURLProtocol
- регистрация его в системе загрузки URL

UI tests. Stub manager. NSURLProtocol



Сбербанк Онлайн

Devteam

UI tests. Stub manager. NSURLProtocol

```
startLoading {

    cachedData: Data

    if cachedData {
        response: NSHTTPURLResponse
        client(URLProtocol: didReceiveResponse: cacheStoragePolicy:)
        client(URLProtocol:self didLoadData:cachedData)
        client(URLProtocolDidFinishLoading:)
    } else {
        client(URLProtocol:self didFailWithError:создаем ошибку)
    }
}
```

Сбербанк Онлайн

Devteam

UI tests. Stub manager. NSURLProtocol

Instance Property

protocolClasses

An array of extra protocol subclasses that handle requests in a session.

Declaration

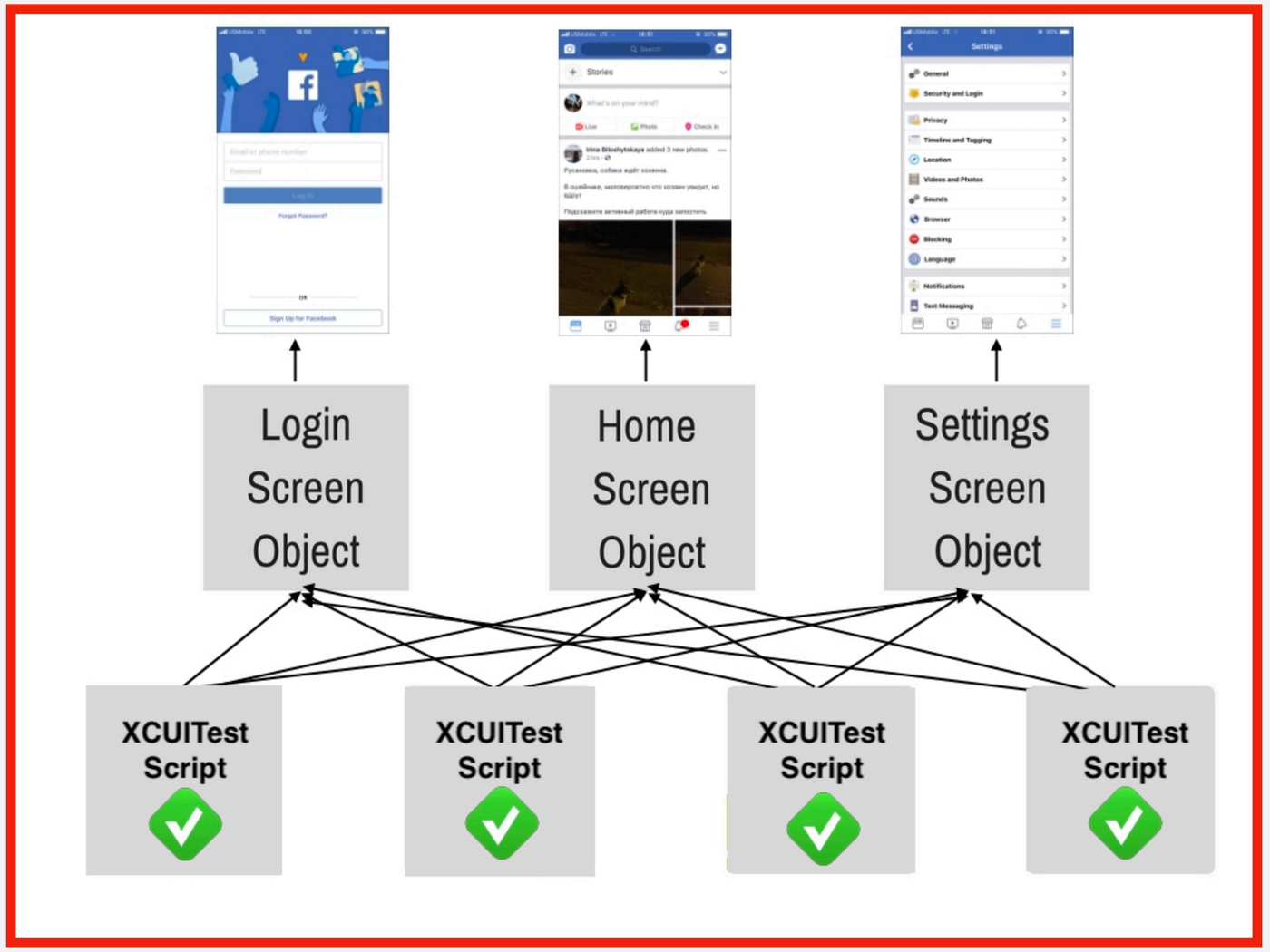
```
var protocolClasses: [AnyClass]? { get set }
```

Сбербанк Онлайн

https://developer.apple.com/documentation/foundation/urlsessionconfiguration/1411050-protocolclasses

Devteam

UI tests. Page object pattern



Сбербанк Онлайн

Devteam

UI tests. Page object pattern

```
func testExample() {

    let app = XCUIApplication()

    let signUpPage = WelcomePage(app: app)

    let completePage = signUpPage.
        .typeEmail("smith@example.com")
        .typePassword("jg28hwm90iflz")
        .typeName("Smith")
        .tapCreateAccountButton()

    _ = completePage
        .then { XCTAssertEqual($0.headingText, "Welcome, Smith!") }
        .pickCompany("Japan")

    // ...
}
```

Сбербанк Онлайн

Devteam

UI tests. Page object pattern

```
public final class NewOnlineConsultationsPage: PageWithWorker, BackNavigation {
    public init() {}

    public let screenName = "Онлайн-консультации"

    public var openElement: XCUIElement { app.otherElements["OnlineServicesViewController"] }

    public let openTimeout: Timeout = .apiRequest

    public var paidConsultationsHeader: XCUIElement { app.otherElements["Доступные консультации"] }

    public var availableConsultationsHeader: XCUIElement { app.otherElements["Можно приобрести"] }

    public private(set) lazy var pageWorker = PageWorker(page: self)

    // MARK: - Elements
    public var backButton: XCUIElement { app.buttons["Главная"] }
    public var searchButton: XCUIElement { app.navigationBars.buttons.element(boundBy: 1) }
    public var promocodeButton: XCUIElement { app.images["online_services_promo"].firstMatch }
    public var tutorialButton: XCUIElement { app.images["online_services_tutorial"].firstMatch }
    public var firstSpecialityListItem: XCUIElement { app.cells.containing(.image, identifier:
        "disclosure_arrow_16").firstMatch }
    public var firstBoughtSpecialityCard: XCUIElement {
        app.otherElements.containing(.staticText, identifier: "bought_speciality").firstMatch
    }
}
```

Сбербанк Онлайн

Devteam

UI tests. Page object pattern

```
protocol Page {
    var app: XCUIApplication { get }
    var view: XCUIElement { get }

    init(app: XCUIApplication)
}
```

UI tests. Page object pattern

```
class WelcomePage: Page {
    // ...
    private var loginButton: XCUIElement { return view.buttons["loginButton"] }
    private var signUpButton: XCUIElement { return view.buttons["signUpButton"] }

    func tapLoginButton() -> LoginPage {
        loginButton.tap()
        return LoginPage(app: app)
    }

    func tapSignUpButton() -> SignUpPage {
        signUpButton.tap()
        return SignUpPage(app: app)
    }
}
```

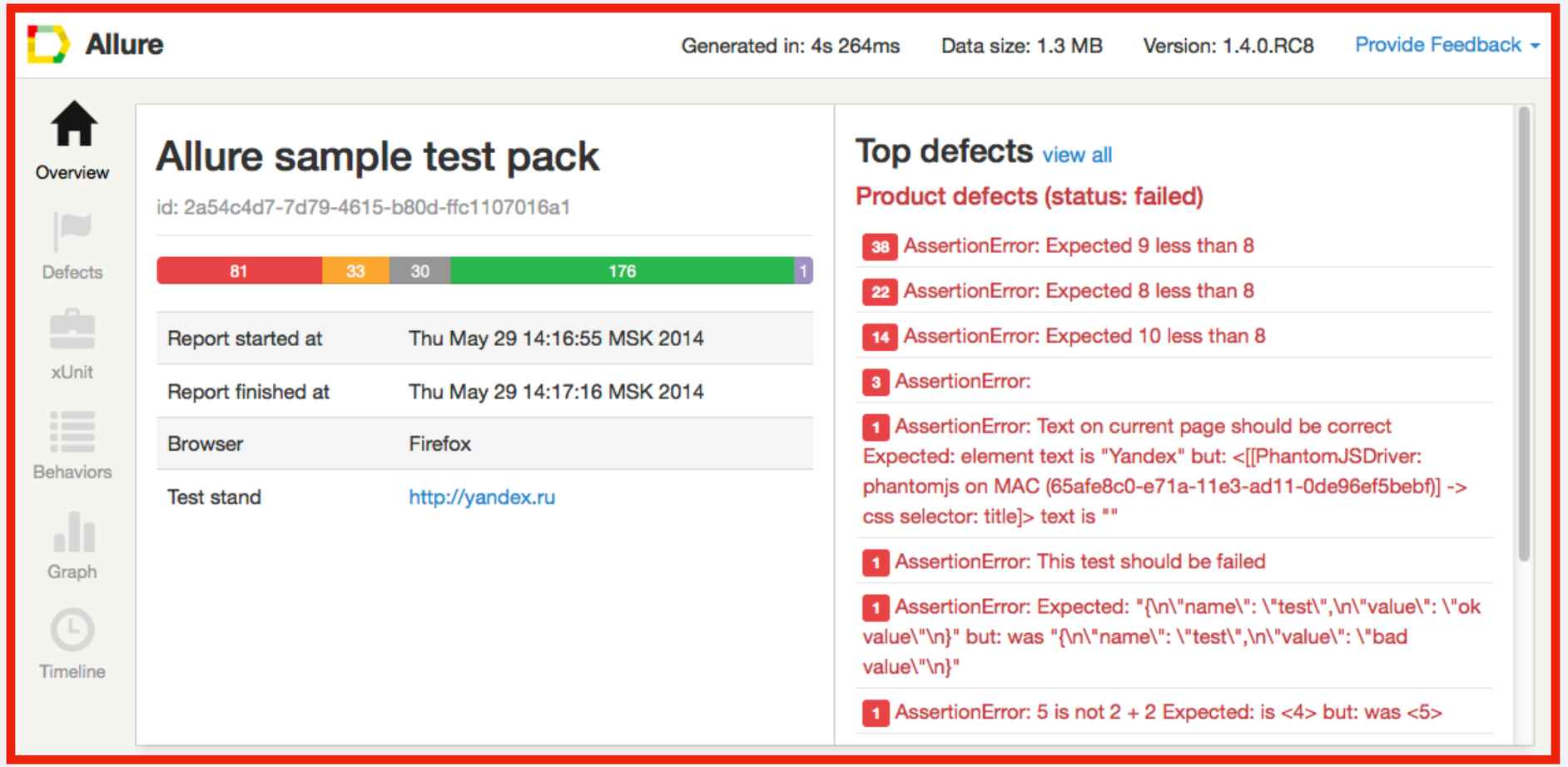
UI tests. Page object pattern

```
signUpPage
      .typeEmail("")
      .typePassword("3i&wu,iu87n")
      .then { XCTAssertEqual(signUpPage.errorMessage,
              “Email can't be blank.") }
```

Сбербанк Онлайн

Devteam

UI tests. Snapshot

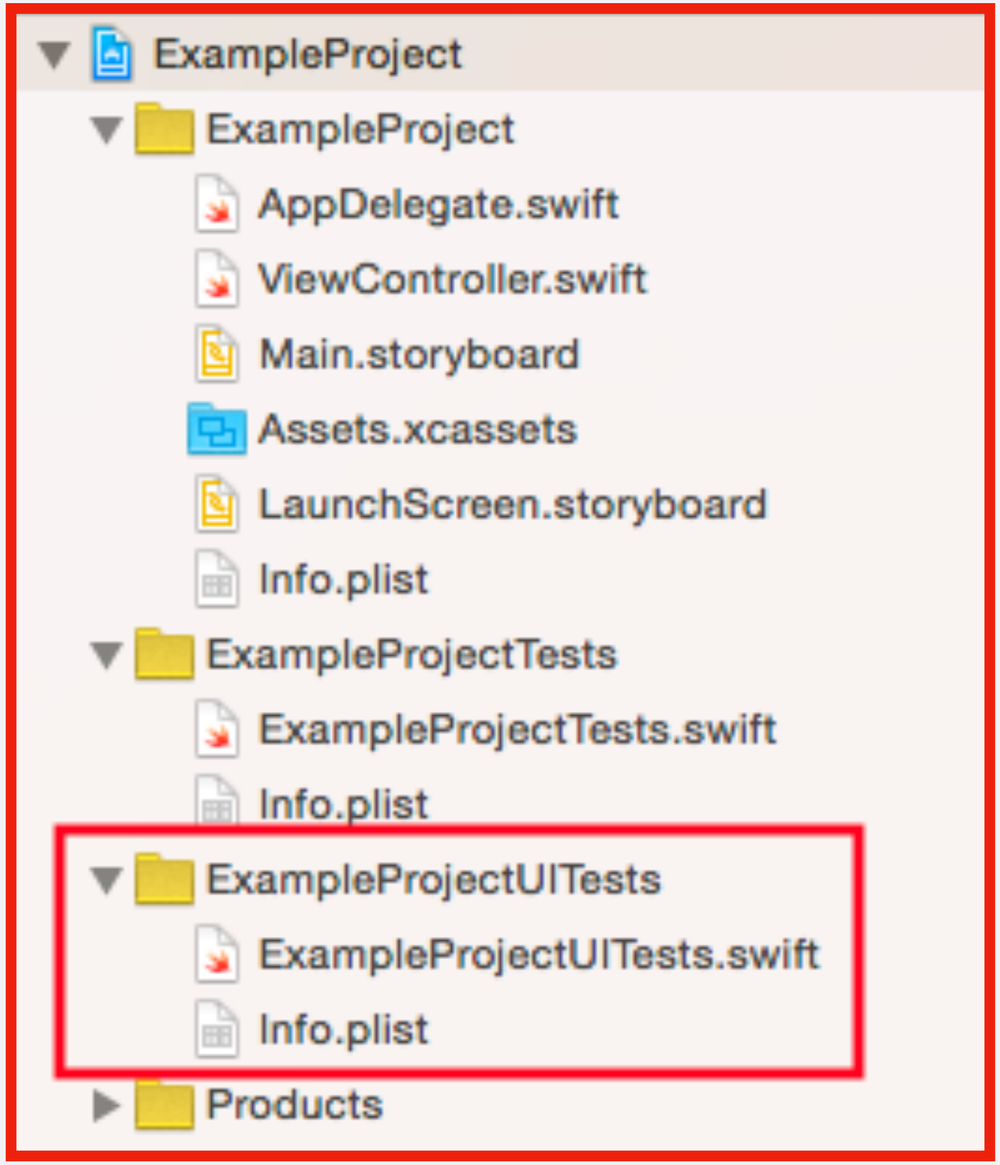


Сбербанк Онлайн

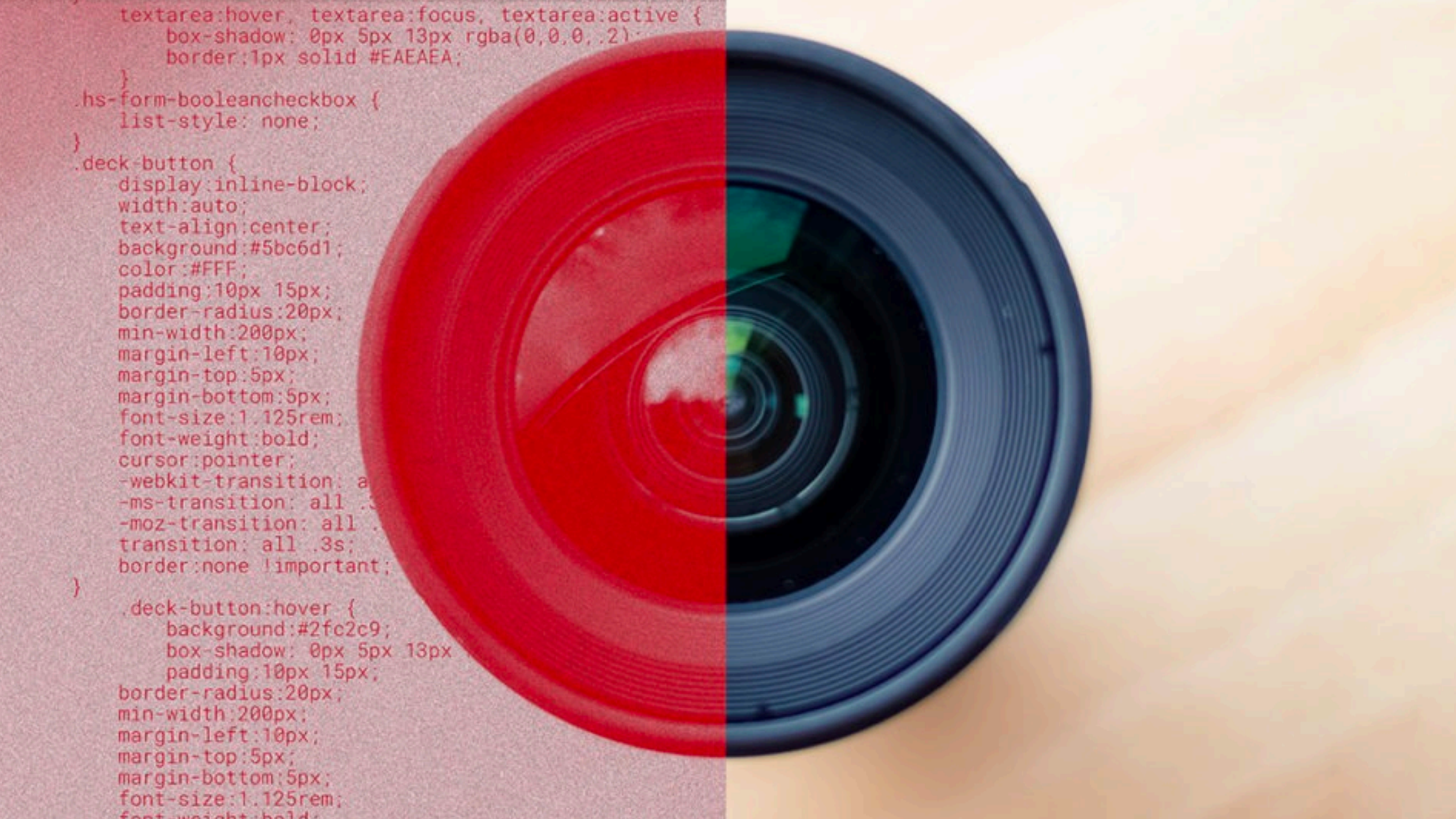
Devteam

UI tests

- Stub manager
- Page object pattern
- Snapshot



Snapshot testing


textarea:hover, textarea:focus, textarea:active {
box-shadow: 0px 5px 13px rgba(0,0,0,.2);
border:1px solid #EAEAEA;
}
.hs-form-booleancheckbox {
list-style: none;
}
.deck-button {
display:inline-block;
width:auto;
text-align:center;
background:#5bc6d1;
color:#FFF;
padding:10px 15px;
border-radius:20px;
min-width:200px;
margin-left:10px;
margin-top:5px;
margin-bottom:5px;
font-size:1.125rem;
font-weight:bold;
cursor:pointer;
-webkit-transition: a
-ms-transition: all .
-moz-transition: all .
transition: all .3s;
border:none !important;
}
.deck-button:hover {
background:#2fc2c9;
box-shadow: 0px 5px 13px
padding:10px 15px;
border-radius:20px;
min-width:200px;
margin-left:10px;
margin-top:5px;
margin-bottom:5px;
font-size:1.125rem;

Login

Email

Password

Login

SIGN IN

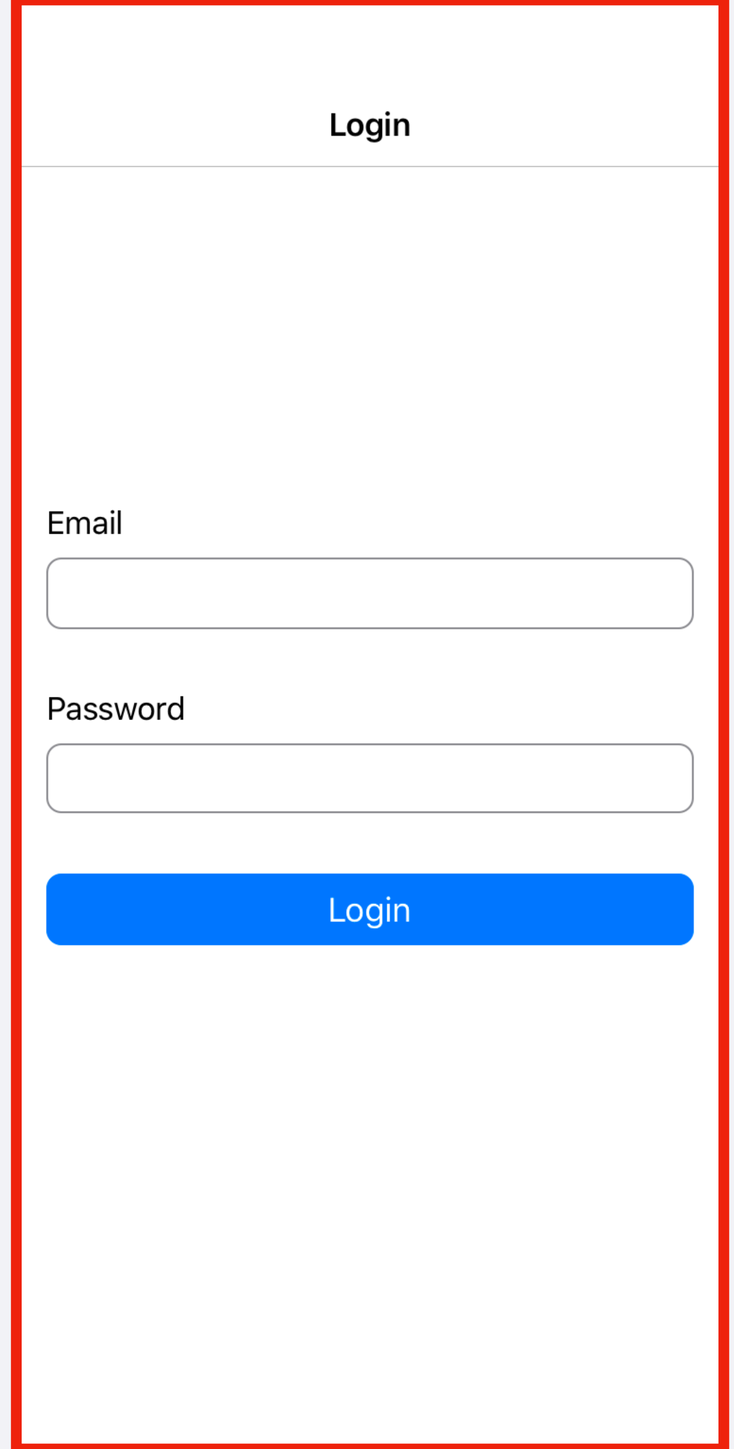
EMAIL

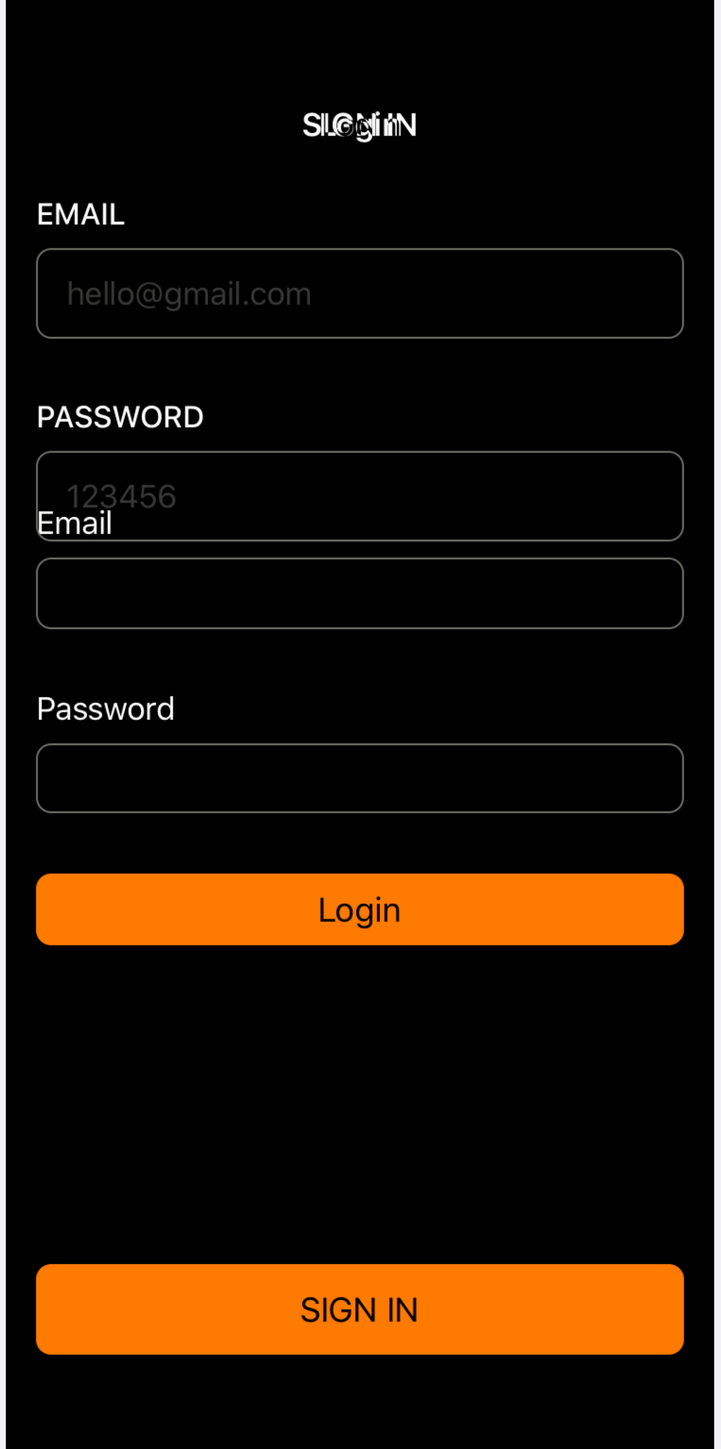
hello@gmail.com

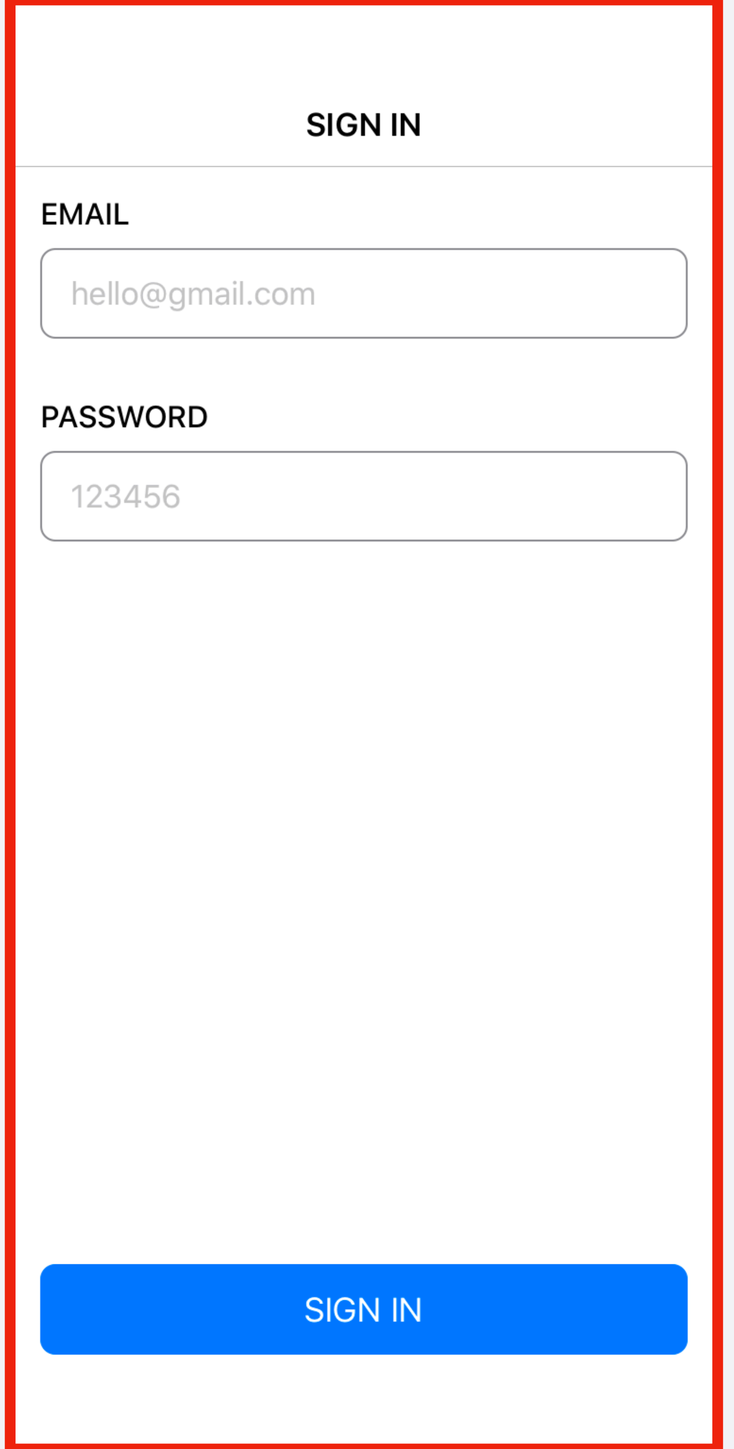
PASSWORD

123456

SIGN IN


Login
Email
Password
Login


SIGN IN
EMAIL
hello@gmail.com
PASSWORD
123456
Email
Password
Login
SIGN IN


SIGN IN
EMAIL
hello@gmail.com
PASSWORD
123456
SIGN IN

Snapshot testing

- https://github.com/pointfreeco/swift-snapshot-testing
- https://github.com/uber/ios-snapshot-test-case/ (**obsolete**)

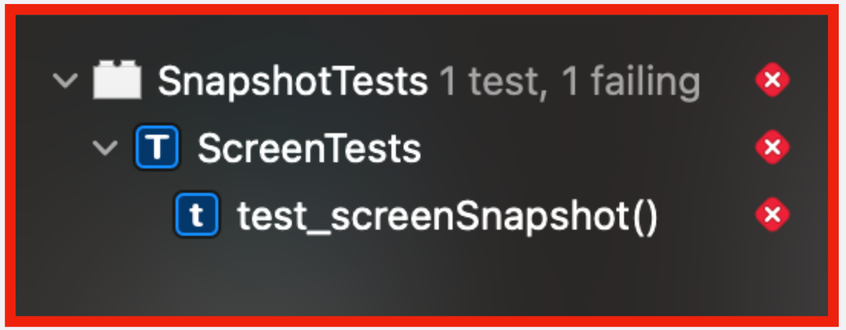
Snapshot testing. PointFreeCo

```
class Screen: UIViewController {

  override func viewDidLoad() {
    super.viewDidLoad()
    view.backgroundColor = .red
  }
}
```

Snapshot testing. PointFreeCo

```
import XCTest
import SnapshotTesting

@testable import Snapshot

final class ScreenTests: XCTestCase {
  var sut: Screen!

  override func setUp() {
    sut = Screen()
  }

  func test_screenSnapshot() {
    assertSnapshot(matching: sut, as: .image(on: .iPhone8))
  }
}
```

Snapshot testing. PointFreeCo




```
  func test_screenSnapshot() {
    assertSnapshot(matching: sut, as: .image(on: .iPhone8))
  }
}
```

failed - No reference was found on disk. Automatically recorded snapshot: ...

open "/Users/docdoc/Desktop/Examples/Snapshot/SnapshotTests/__Snapshots__/ScreenTests/test_screenSnapshot.1.png"

Re-run "test_screenSnapshot" to test against the newly-recorded snapshot.

Snapshot testing. PointFreeCo

```
19
    func test_screenSnapshot() {
21      assertSnapshot(matching: sut, as: .image(on: .iPhone8))
22    }
23  }
24
```

failed - Snapshot does not match reference.

@−
"/Users/docdoc/Desktop/Examples/Snapshot/SnapshotTests/
__Snapshots__/ScreenTests/test_screenSnapshot.1.png"
@+
"/Users/docdoc/Library/Developer/CoreSimulator/Devices/
49B6E336-4D6E-428E-8FC3-E8234F355679/data/Containers/Data/
Application/5C42EE4F-7FD0-4093-A873-E6A4A61D94B4/tmp/
ScreenTests/test_screenSnapshot.1.png"

Newly-taken snapshot does not match reference.

Snapshot testing. PointFreeCo

ScreenTests SnapshotTests 1 failed (100 %) in 1s

test_screenSnapshot() 1s

Attached Failure Diff (Start)

reference

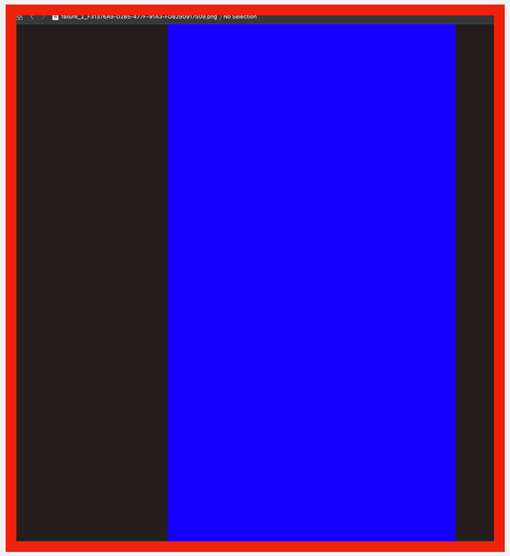
failure

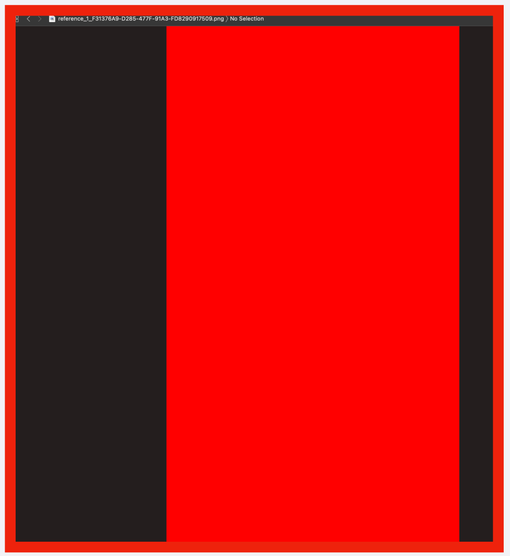
difference

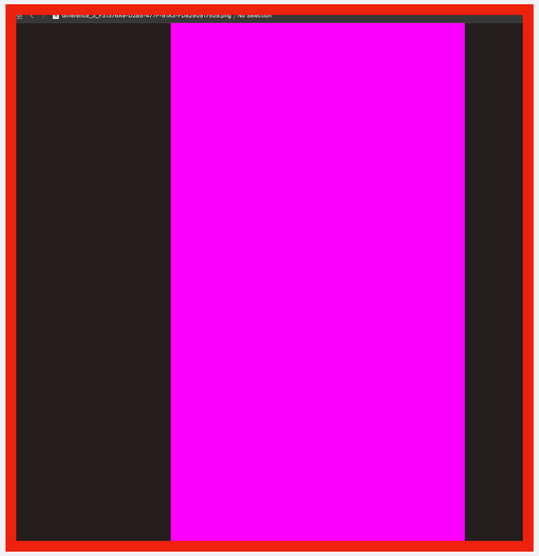
Assertion Failure at ScreenTests.swift:21:
failed - Snapshot does not match reference.

@−
"/Users/docdoc/Desktop/Examples/Snapshot/SnapshotTests/__Snapshots__/ScreenTests/test_screenSnapshot.1.png"
@+
"/Users/docdoc/Library/Developer/CoreSimulator/Devices/49B6E336-4D6E-428E-8FC3-E8234F355679/data/Containers/Data/Application/5C42EE4F-7FD0-4093-A873-E6A4A61D94B4/tmp/ScreenTests/test_screenSnapshot.1.png"

Newly-taken snapshot does not match reference.







Сбербанк Онлайн

Devteam

Read

- https://www.raywenderlich.com/21020457-ios-unit-testing-and-ui-testing-tutorial#toc-anchor-015
- https://swiftwithmajid.com/2021/03/18/ui-testing-in-swift-with-xctest-framework/
- https://habr.com/ru/company/oleg-bunin/blog/353276/
- https://www.hackingwithswift.com/articles/148/xcode-ui-testing-cheat-sheet
- https://medium.com/wantedly-engineering/introducing-page-object-pattern-in-ios-74e46c664d26
- https://medium.com/dev-jam/snapshot-testing-in-swift-9d52cbec075c
- https://jon-gabilondo-angulo-7635.medium.com/why-is-xcuitest-so-bad-84917569e28b

- https://github.com/pointfreeco/swift-snapshot-testing

Homework

Реализовать в выпускном проекте

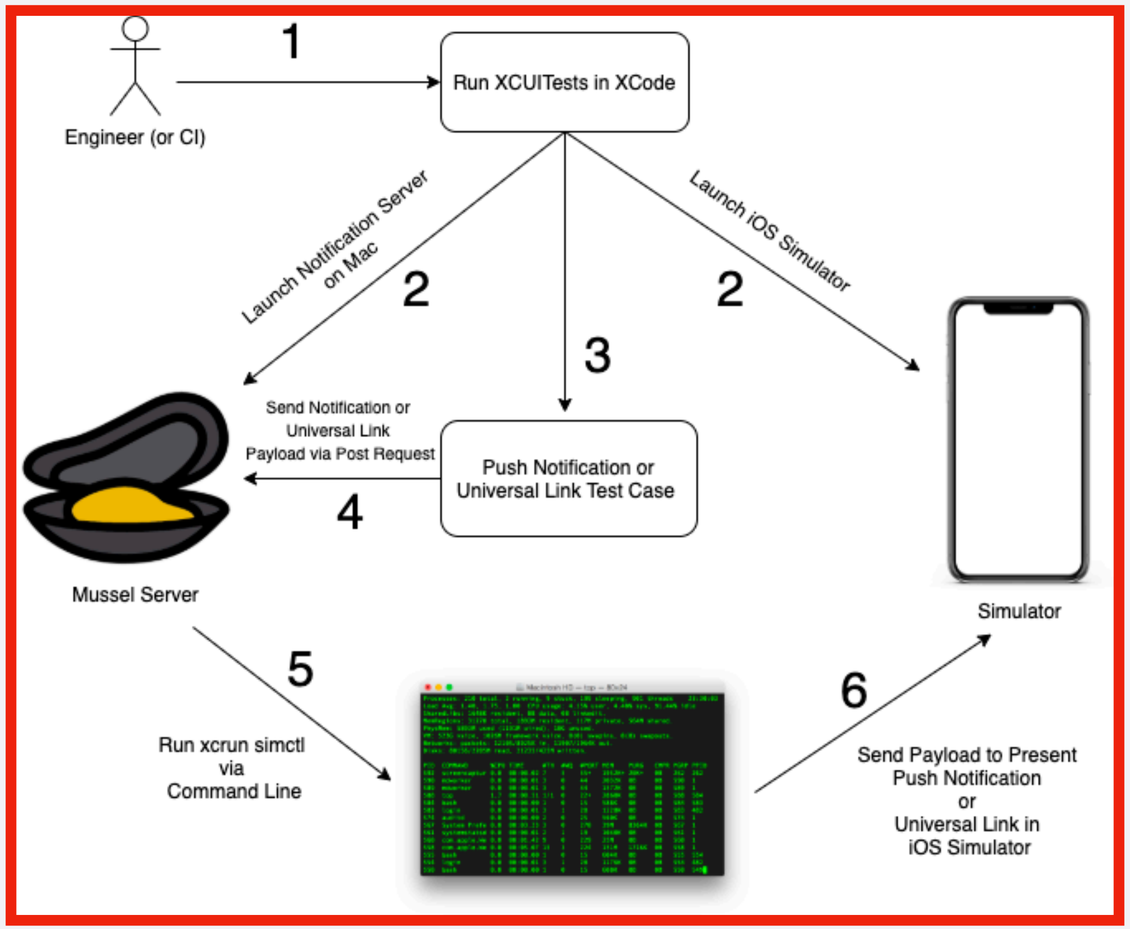
- хотя бы один UI-тест через **page object**
- хотя бы один snapshot test (⚠️ iPhone SE 2)

Прислать на проверку до защиты.

ROBOT Pattern

https://medium.com/zendesk-engineering/ui-testing-ios-apps-with-the-robot-pattern-dd839b59fed1

UI tests. Stub manager. Mussels



https://github.com/UrbanCompass/Mussel

Сбербанк Онлайн

Devteam

Test plans