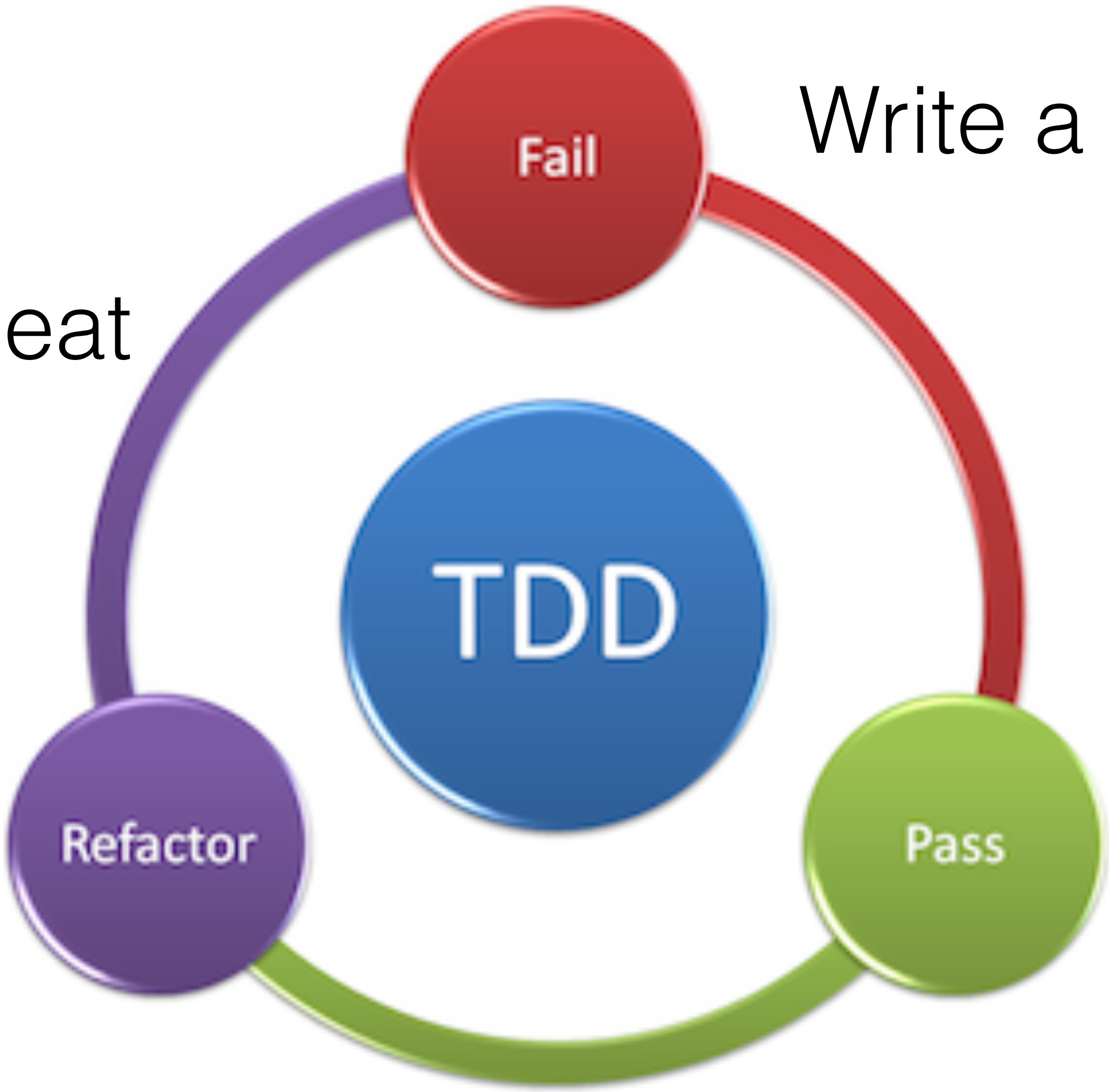
Test Driven Development

- Background
- TDD Workflow
- Getting Started with XCTest
- Testing model, viewcontroller, networks
- Conclusion

Background

1996	Extreme Programming by Kent Beck “You are not allowed write any production code until you write a test”
1998	OCUnit developed by Sen:te, a Swiss Company A port of SUnit, a testing framework written by Kent Beck
2008	Apple added OCUnit to Xcode 2.1 It was used for the development of CoreData, shipped in Tiger, the OS.
2013	Apple introduced XCTest with Xcode 5 Unit testing became a first class citizen in Xcode

TDD Workflow



Write a Failing Test

Write code to make Test Pass

Write a optimised code

Repeat

Getting Started with XCTest

1. Adding Test Target

   New -> Target -> iOS Unit Testing Bundle

2. Pod Setting

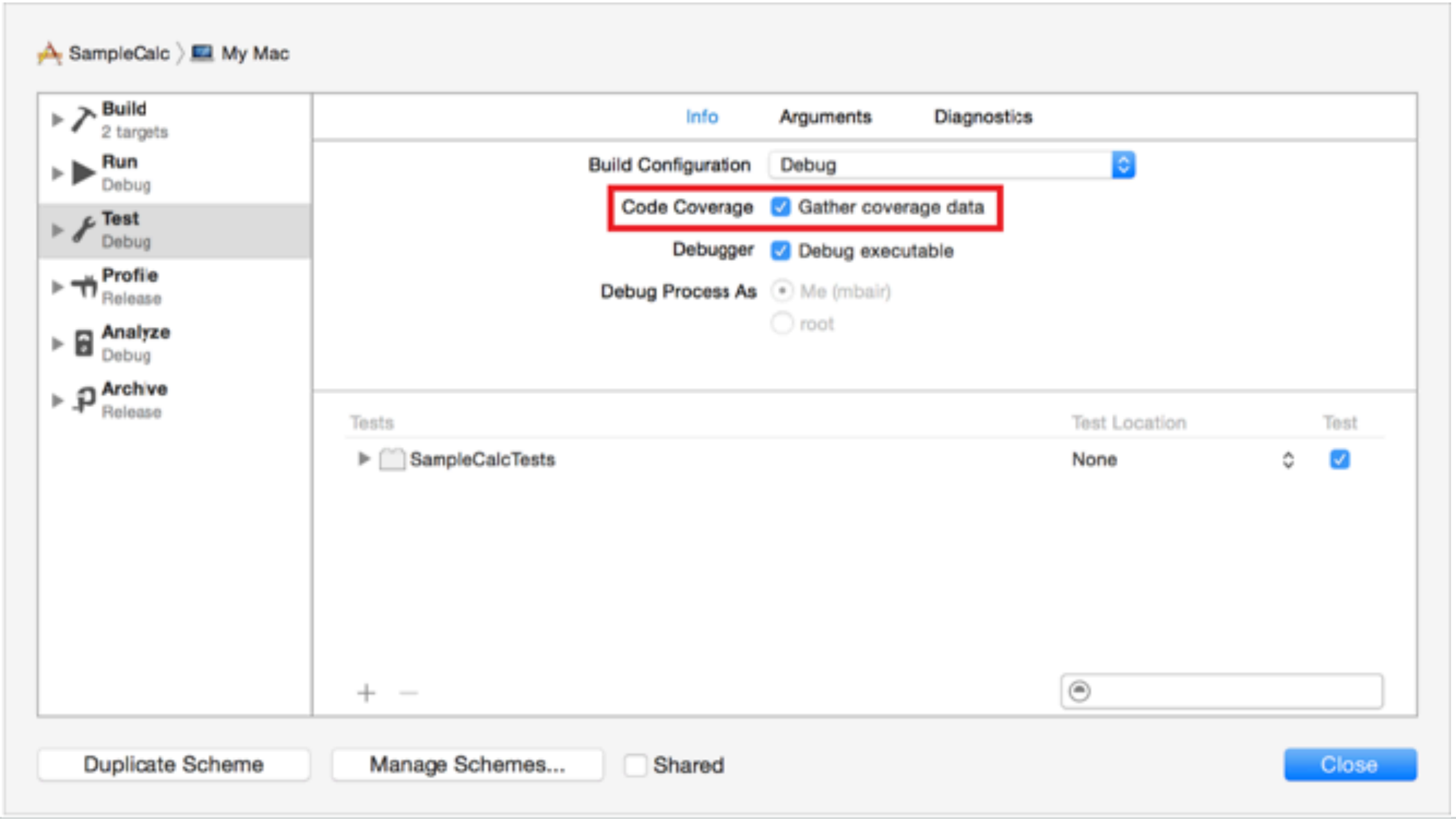
```
inhibit_all_warnings!
use_frameworks!

target :Project do
   pod 'FXBlurView', '~> 1.6'

  target :ProjectTests do
   inherit! :search_paths
  end
```

Getting Started with XCTest

3. Product -> Scheme-> Edit Scheme



SampleCalc 〉 My Mac

Build Configuration: Debug

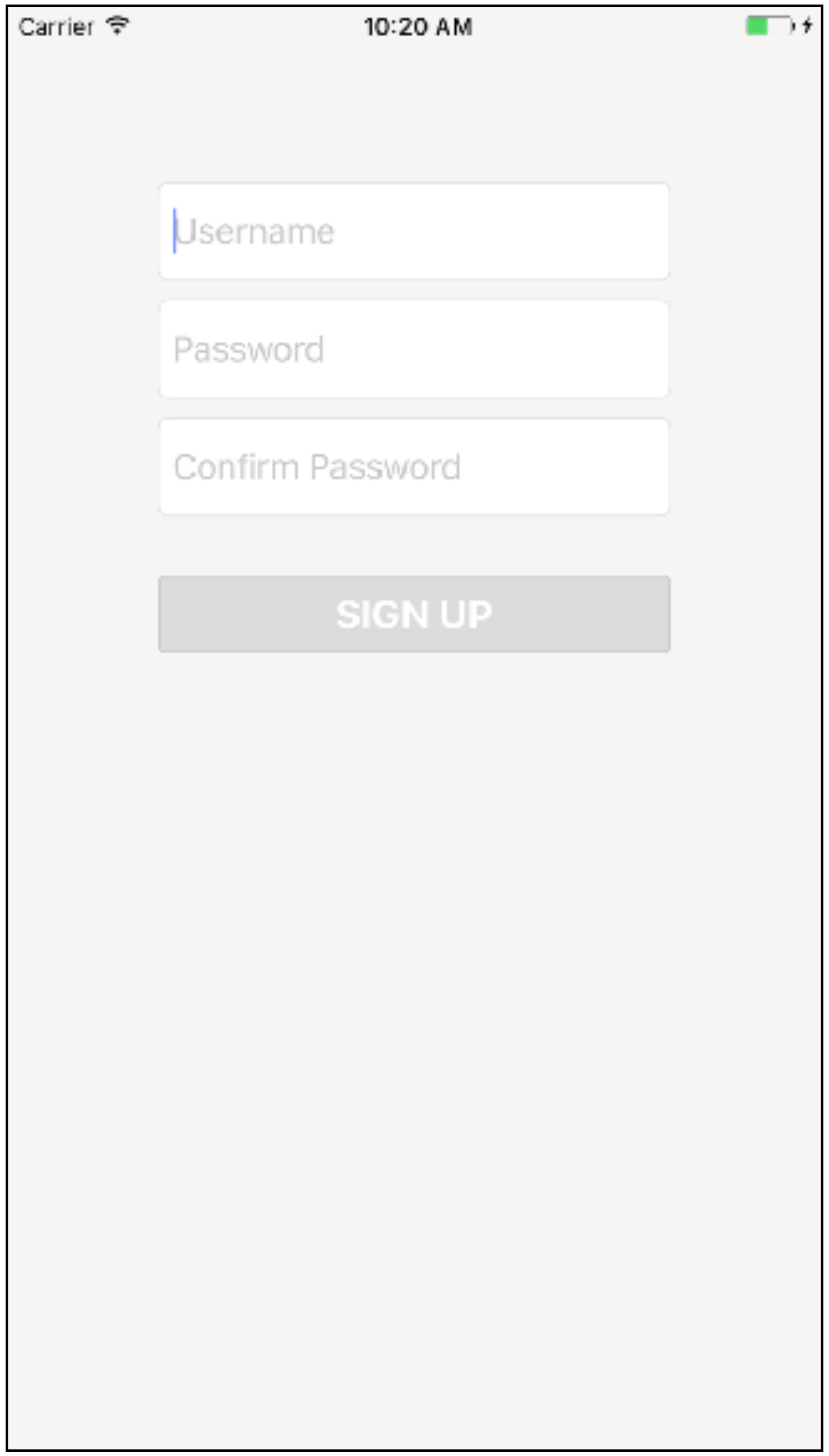
Code Coverage: ✓ Gather coverage data

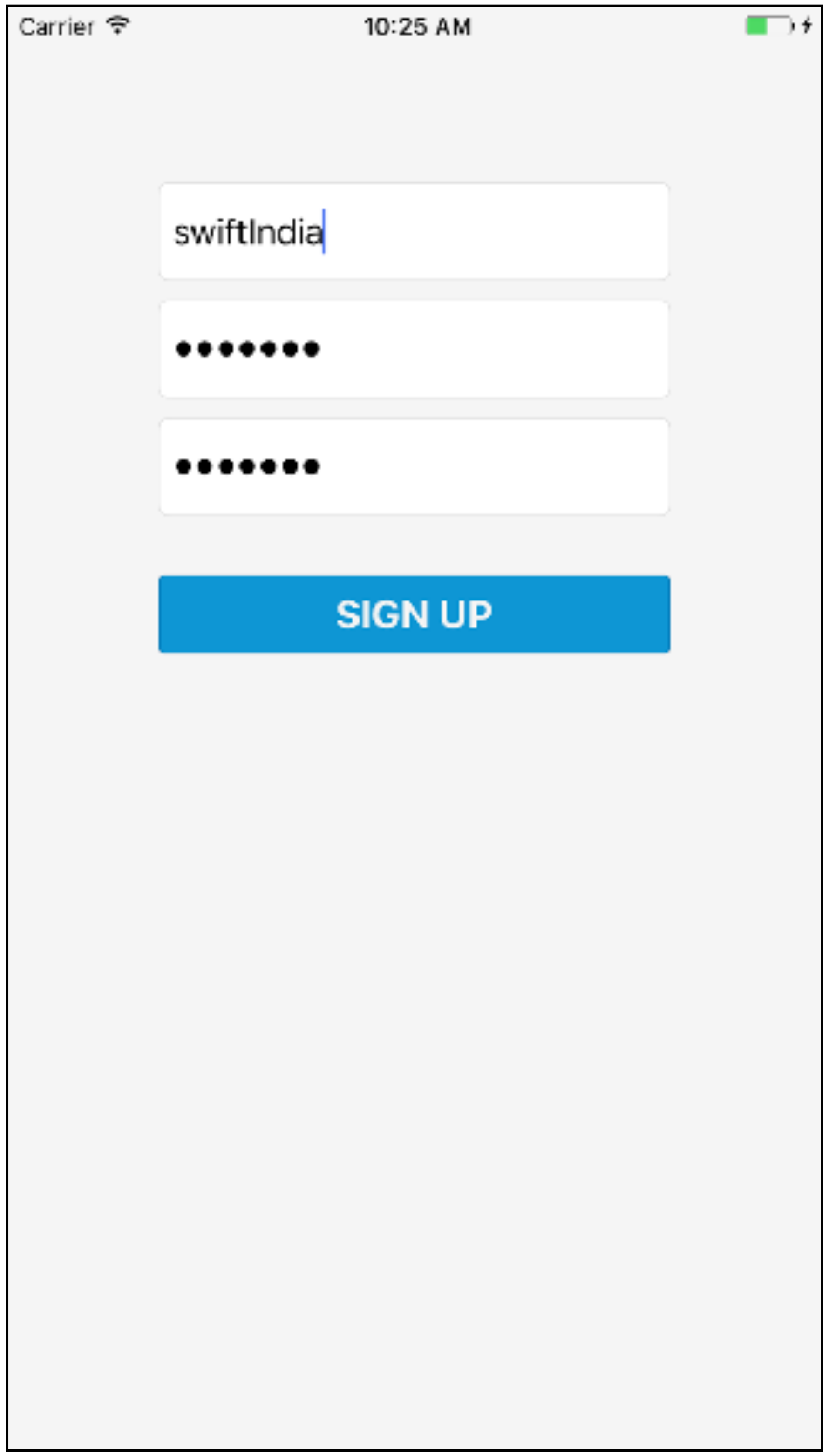
Debugger: ✓ Debug executable

Debug Process As: ● Me (mbair) ○ root

Tests	Test Location	Test
SampleCalcTests	None	✓

Let’s do TDD


Carrier
10:20 AM
Username
Password
Confirm Password
SIGN UP


Carrier
10:25 AM
swiftIndia
•••••••
•••••••
SIGN UP

What to test?

Define the use case:

“As a user I want to be able to sign up to the app”

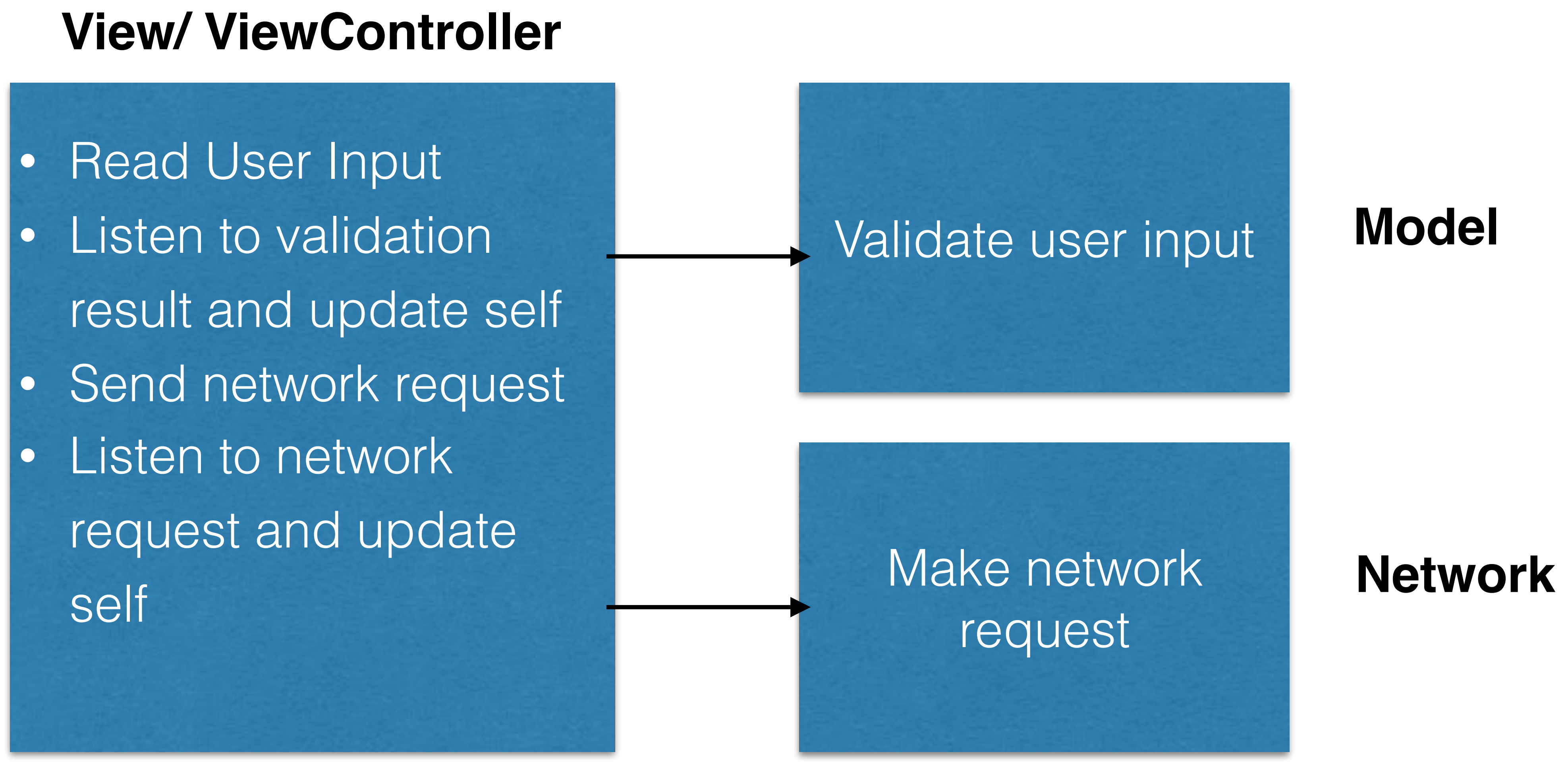
What to test?

Define the responsibilities:

- Read user input
- Validate user input
- Update view based on user inputs
- Send network call after validating user
- Update view to show logged in state

What to test?

Distribute the responsibilities:



Test Structure

Set up
Exercise
Assert
Clean up

Testing Model

FormField

Value

Validator

isValid() -> Bool

Form

FormFields

Dependency Validator

isValid() -> Bool

Testing ViewController

```
   var sut: PharmacyViewController!

     override func setUp() {
         super.setUp()
         let storyboard = UIStoryboard(name: "Main", bundle: Bundle.main)
    sut = storyboard.instantiateViewController(withIdentifier:
"PharmacyViewControllerID") as! PharmacyViewController
    _ = sut.view

}

   func testTableViewNotNil() {
    XCTAssertNotNil(sut.tableView)
   }
```

`_ = sut.view triggers viewDidLoad`

Testing Networking Call

```
  func testAuthorizationCall() {

    let asyncExpectation = expectation(description: "long
running method")
    var authToken = ""
    authorizeUser(username: "swifty", password: "123456") { string
in
      authToken = string
      asyncExpectation.fulfill()
    }

    self.waitForExpectations(timeout: 2.0) { error in
      XCTAssertFalse(authToken.isEmpty)
    }
  }
```

Conclusion

- Shapes your thought process. You think in terms of dependencies and focus on decoupling.
- Documents your code.
- Protects you from regression issues
- Gives you confidence in refactoring stuff

So let’s do a favour to ourselves

WRITE TESTS

Susmita Horrow
@SusmitaHorrow

Demo Project