### **Lab Exercise: Unit Testing Programmatically in iOS Swift**

**Objective:**

Learn how to create and use unit tests to validate the functionality of your iOS app.

**Set Up the Project:**

* Create a new Xcode project named UnitTestingDemo.
* Choose "App" under iOS and set the language to Swift.
* Ensure that the "Include Tests" option is selected when creating the project. This will automatically create a test target for your project.

**Implement a Simple Model:**

* Open UnitTestingDemo group in Xcode.
* Create a new Swift file named Calculator.swift.
* Implement the following code in Calculator.swift:

import Foundation

class Calculator {

func add(\_ a: Int, \_ b: Int) -> Int {

return a + b

}

func subtract(\_ a: Int, \_ b: Int) -> Int {

return a - b

}

func multiply(\_ a: Int, \_ b: Int) -> Int {

return a \* b

}

func divide(\_ a: Int, \_ b: Int) -> Int? {

guard b != 0 else { return nil }

return a / b

}

}

**Implement Unit Tests:**

* Open UnitTestingDemoTests group in Xcode.
* Open UnitTestingDemoTests.swift (or create a new Swift file in the test target if it does not exist).
* Replace the contents with the following code:

import XCTest

@testable import UnitTestingDemo

class UnitTestingDemoTests: XCTestCase {

var calculator: Calculator!

override func setUpWithError() throws {

// Put setup code here. This method is called before the invocation of each test method in the class.

calculator = Calculator()

}

override func tearDownWithError() throws {

// Put teardown code here. This method is called after the invocation of each test method in the class.

calculator = nil

}

func testAddition() throws {

XCTAssertEqual(calculator.add(1, 1), 2)

XCTAssertEqual(calculator.add(-1, -1), -2)

XCTAssertEqual(calculator.add(0, 0), 0)

}

func testSubtraction() throws {

XCTAssertEqual(calculator.subtract(2, 1), 1)

XCTAssertEqual(calculator.subtract(-1, -1), 0)

XCTAssertEqual(calculator.subtract(0, 0), 0)

}

func testMultiplication() throws {

XCTAssertEqual(calculator.multiply(2, 3), 6)

XCTAssertEqual(calculator.multiply(-2, -3), 6)

XCTAssertEqual(calculator.multiply(-2, 3), -6)

}

func testDivision() throws {

XCTAssertEqual(calculator.divide(6, 3), 2)

XCTAssertNil(calculator.divide(6, 0)) // Test division by zero

XCTAssertEqual(calculator.divide(-6, -3), 2)

XCTAssertEqual(calculator.divide(-6, 3), -2)

}

}

**Run the Tests:**

* Select the test target (UnitTestingDemoTests) from the target selector in the Xcode toolbar.
* Press Command-U to run all the tests.
* Verify that all tests pass successfully.

**Summary:**

This lab exercise provides hands-on experience with implementing unit tests programmatically in an iOS app using Swift. By completing the exercise, you will gain practical knowledge of setting up and writing unit tests to validate the functionality of your app's components, ensuring they work as expected.