### **Lab Exercise: Design a Login Form Using Code and Perform UI Testing in iOS Swift**

**Objective:**

Create a login form programmatically and write UI tests to verify its functionality.

**Part 1: Create a Login Form Programmatically**

**Set Up the Project:**

* Open Xcode and create a new project.
* Choose "App" under iOS and set the language to Swift.
* Name the project LoginFormUITest.

**Design the Login Form Programmatically:**

* Open ViewController.swift and replace its content with the following code to create the login form programmatically.

import UIKit

class ViewController: UIViewController {

let usernameTextField: UITextField = {

let textField = UITextField()

textField.placeholder = "Username"

textField.borderStyle = .roundedRect

textField.translatesAutoresizingMaskIntoConstraints = false

return textField

}()

let passwordTextField: UITextField = {

let textField = UITextField()

textField.placeholder = "Password"

textField.borderStyle = .roundedRect

textField.isSecureTextEntry = true

textField.translatesAutoresizingMaskIntoConstraints = false

return textField

}()

let loginButton: UIButton = {

let button = UIButton(type: .system)

button.setTitle("Login", for: .normal)

button.translatesAutoresizingMaskIntoConstraints = false

button.addTarget(self, action: #selector(loginButtonTapped), for: .touchUpInside)

return button

}()

let resultLabel: UILabel = {

let label = UILabel()

label.textColor = .black

label.translatesAutoresizingMaskIntoConstraints = false

return label

}()

override func viewDidLoad() {

super.viewDidLoad()

view.backgroundColor = .white

view.addSubview(usernameTextField)

view.addSubview(passwordTextField)

view.addSubview(loginButton)

view.addSubview(resultLabel)

setupConstraints()

}

func setupConstraints() {

NSLayoutConstraint.activate([

usernameTextField.centerXAnchor.constraint(equalTo: view.centerXAnchor),

usernameTextField.topAnchor.constraint(equalTo: view.topAnchor, constant: 100),

usernameTextField.widthAnchor.constraint(equalToConstant: 200),

passwordTextField.centerXAnchor.constraint(equalTo: view.centerXAnchor),

passwordTextField.topAnchor.constraint(equalTo: usernameTextField.bottomAnchor, constant: 20),

passwordTextField.widthAnchor.constraint(equalToConstant: 200),

loginButton.centerXAnchor.constraint(equalTo: view.centerXAnchor),

loginButton.topAnchor.constraint(equalTo: passwordTextField.bottomAnchor, constant: 20),

resultLabel.centerXAnchor.constraint(equalTo: view.centerXAnchor),

resultLabel.topAnchor.constraint(equalTo: loginButton.bottomAnchor, constant: 20)

])

}

@objc func loginButtonTapped() {

let username = usernameTextField.text ?? ""

let password = passwordTextField.text ?? ""

if username == "user" && password == "password" {

resultLabel.text = "Login successful"

resultLabel.textColor = .green

} else {

resultLabel.text = "Login failed"

resultLabel.textColor = .red

}

}

}

**Run the App:**

Build and run the app to ensure the login form appears correctly.

**Part 2: Perform UI Testing**

**Add a New UI Test Target:**

* In Xcode, go to File > New > Target.
* Choose UI Testing Bundle and click Next.
* Name the target LoginFormUITests.

**Write UI Tests:**

* Open LoginFormUITests.swift (created by Xcode) and replace its content with the following code to write UI tests for the login form.

import XCTest

class LoginFormUITests: XCTestCase {

override func setUpWithError() throws {

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

continueAfterFailure = false

// UI tests must launch the application that they test.

let app = XCUIApplication()

app.launch()

}

func testSuccessfulLogin() throws {

let app = XCUIApplication()

let usernameTextField = app.textFields["Username"]

let passwordTextField = app.secureTextFields["Password"]

let loginButton = app.buttons["Login"]

let resultLabel = app.staticTexts["Login successful"]

// Ensure initial state

XCTAssertTrue(usernameTextField.exists)

XCTAssertTrue(passwordTextField.exists)

XCTAssertTrue(loginButton.exists)

// Perform the login

usernameTextField.tap()

usernameTextField.typeText("user")

passwordTextField.tap()

passwordTextField.typeText("password")

loginButton.tap()

// Check for successful login

XCTAssertTrue(resultLabel.exists)

}

func testFailedLogin() throws {

let app = XCUIApplication()

let usernameTextField = app.textFields["Username"]

let passwordTextField = app.secureTextFields["Password"]

let loginButton = app.buttons["Login"]

let resultLabel = app.staticTexts["Login failed"]

// Ensure initial state

XCTAssertTrue(usernameTextField.exists)

XCTAssertTrue(passwordTextField.exists)

XCTAssertTrue(loginButton.exists)

// Perform the login with incorrect credentials

usernameTextField.tap()

usernameTextField.typeText("wronguser")

passwordTextField.tap()

passwordTextField.typeText("wrongpassword")

loginButton.tap()

// Check for failed login

XCTAssertTrue(resultLabel.exists)

}

}

**Run the UI Tests:**

* In Xcode, select the LoginFormUITests scheme.
* Go to Product > Test or press Cmd+U to run the tests.
* Ensure all tests pass successfully.

**Summary:**

This lab exercise demonstrates how to create a login form programmatically in Swift and write UI tests to verify its functionality. You learned how to:

* Create UI elements and constraints programmatically.
* Write @IBOutlet and @IBAction to handle user interactions.
* Write UI tests using XCTest to automate testing of the login functionality.
* By completing this exercise, you will gain practical knowledge in programmatically creating UI components and ensuring they function correctly through automated UI testing.