### **Lab -** **Gesture with Code in Swift**

**Lab Exercise: Gesture Handling in iOS with Swift**

**Objective:**

* Learn how to use gesture recognizers in iOS to handle user interactions like taps, swipes, pinches, and rotations.

**Steps:**

**Set Up the Project:**

* Create a new Xcode project named GestureLab.
* Choose "App" under iOS and make sure the language is set to Swift.

**Create the View Controller:**

* Create a new Swift file named GestureViewController.swift.

**Design the Interface Programmatically:**

* In GestureViewController.swift, set up a view with a label and a square view that will respond to gestures.

**Add Gesture Recognizers:**

* Add tap, swipe, pinch, and rotation gesture recognizers to the square view.

**Handle Gestures:**

* Implement methods to handle each gesture and update the UI accordingly.

**Step-by-Step Implementation:**

Step 1: Set Up the Project

* Open Xcode and create a new project.
* Select "App" under iOS and click "Next".
* Name your project GestureLab, set the language to Swift, and make sure "Use SwiftUI" is unchecked.
* Save the project to your desired location.

Step 2: Create the View Controller

* Create a new Swift file named GestureViewController.swift.

import UIKit

class GestureViewController: UIViewController {

// Define properties for the label and square view

let label = UILabel()

let squareView = UIView()

override func viewDidLoad() {

super.viewDidLoad()

// Set up the main view

self.view.backgroundColor = .white

// Set up the label

setupLabel()

// Set up the square view

setupSquareView()

// Add gesture recognizers

addGestureRecognizers()

}

// Method to set up the label

private func setupLabel() {

label.text = "Perform gestures on the blue square"

label.textAlignment = .center

label.translatesAutoresizingMaskIntoConstraints = false

self.view.addSubview(label)

// Set label constraints

NSLayoutConstraint.activate([

label.leadingAnchor.constraint(equalTo: self.view.leadingAnchor, constant: 20),

label.trailingAnchor.constraint(equalTo: self.view.trailingAnchor, constant: -20),

label.topAnchor.constraint(equalTo: self.view.topAnchor, constant: 40)

])

}

// Method to set up the square view

private func setupSquareView() {

squareView.backgroundColor = .systemBlue

squareView.translatesAutoresizingMaskIntoConstraints = false

self.view.addSubview(squareView)

// Set square view constraints

NSLayoutConstraint.activate([

squareView.centerXAnchor.constraint(equalTo: self.view.centerXAnchor),

squareView.centerYAnchor.constraint(equalTo: self.view.centerYAnchor),

squareView.widthAnchor.constraint(equalToConstant: 200),

squareView.heightAnchor.constraint(equalToConstant: 200)

])

}

// Method to add gesture recognizers

private func addGestureRecognizers() {

// Tap Gesture

let tapGesture = UITapGestureRecognizer(target: self, action: #selector(handleTap(\_:)))

squareView.addGestureRecognizer(tapGesture)

// Swipe Gesture

let swipeGesture = UISwipeGestureRecognizer(target: self, action: #selector(handleSwipe(\_:)))

swipeGesture.direction = .right

squareView.addGestureRecognizer(swipeGesture)

// Pinch Gesture

let pinchGesture = UIPinchGestureRecognizer(target: self, action: #selector(handlePinch(\_:)))

squareView.addGestureRecognizer(pinchGesture)

// Rotation Gesture

let rotationGesture = UIRotationGestureRecognizer(target: self, action: #selector(handleRotation(\_:)))

squareView.addGestureRecognizer(rotationGesture)

}

@objc private func handleTap(\_ sender: UITapGestureRecognizer) {

label.text = "Tap gesture recognized"

}

@objc private func handleSwipe(\_ sender: UISwipeGestureRecognizer) {

label.text = "Swipe gesture recognized"

}

@objc private func handlePinch(\_ sender: UIPinchGestureRecognizer) {

label.text = "Pinch gesture recognized with scale: \(sender.scale)"

}

@objc private func handleRotation(\_ sender: UIRotationGestureRecognizer) {

label.text = "Rotation gesture recognized with rotation: \(sender.rotation)"

}

}.