Calculator App-> A simple and handy calculator application developed using Swift.

Developing Languages, Tools, and Techniques Needed:

Xcode 14.0.1 iOS 16 iPhone 14 Pro Simulator

https://developer.apple.com/documentation/xcode-release-notes/xcode-14_0_1-release-notes

Swift5 https://www.swift.org/blog/swift-5-released/

SwiftUI https://developer.apple.com/xcode/swiftui/

Synchronous Developing Notes

Add IBoutlets in `ViewController.swift`:

```
@IBOutlet var holder: UIView!
```

Add buttons:

Set up the zero button and make it display at the bottom center:

```
  private func setupNumberPad() {
        let buttonSize = view.frame.size.width / 4
        let zeroButton = UIButton(frame: CGRect(x: 0, y:
holder.frame.size.height-buttonSize, width: buttonSize*3, height:
buttonSize))
        zeroButton.setTitleColor(.black, for: .normal)
        zeroButton.backgroundColor = .white
        zeroButton.setTitle("0", for: .normal)
        holder.addSubview(zeroButton)
    }
```

button zero displayed.PNG

Now add 1 2 3 buttons above using a for loop:

```
     for x in 0..<3 {
        let button1 = UIButton(frame: CGRect(x: buttonSize *
CGFloat(x), y: holder.frame.size.height-(buttonSize*2), width:
buttonSize, height: buttonSize))
        button1.setTitleColor(.black, for: .normal)
        button1.backgroundColor = .white
        button1.setTitle("\(x+1)", for: .normal)
        holder.addSubview(button1)
    }
```

button 1 2 3 displayed.PNG

Copy and paste the for loops twice and we got 0-9 buttons:

0-9 buttons all displayed.PNG

Add CLEAR ALL button:

```
let clearButton = UIButton(
frame: CGRect(x: 0, y: holder.frame.size.height-(buttonSize*5),
   width: view.frame.size.width, height: buttonSize))
    clearButton.setTitleColor(.black, for: .normal)
    clearButton.backgroundColor = .white
    clearButton.setTitle("Clear ALL", for: .normal)
    holder.addSubview(clearButton)
```

Add mathematical operations:

```
    let operations = ["+", "-", "x", "/"]
    for x in 0..<4 {
        let button4 = UIButton(frame: CGRect(x: buttonSize * 3, y:
holder.frame.size.height-(buttonSize*CGFloat(x+1)), width: buttonSize,
height: buttonSize))
        button4.setTitleColor(.white, for: .normal)
        button4.backgroundColor = .orange
        button4.setTitle(operations[x], for: .normal)
        holder.addSubview(button4)
    }
```

Now CLEAR ALL and all mathematical operations displayed:

clear all and all mathematical operations displayed.PNG

Now to let the initial result label display, define result label:

```
 private var resultLabel: UILabel = {
        let label = UILabel()
        label.text = "0"
        label.textColor = .white
        label.textAlignment = .right
        label.font = UIFont(name: "Helvetica", size: 100)
        return label
    }()
```

Confine the result label:

```
  resultLabel.frame = CGRect(x: 20, y: clearButton.frame.origin.y -
110.0, width: view.frame.size.width-40 , height:100 )
  holder.addSubview(resultLabel)
```

Now the initial result label as 0 showing:

initial result label displayed.PNG

Add a = button and make all actions align better:

```
 clearButton.addTarget(self, action: #selector(clearResult), for:
.touchUpInside)
   }
    @objc func clearResult(){
        resultLabel.text="0"
```

better aligned calculator.PNG

Make number pressed responding:

```
    @objc func numberPressed(_ sender: UIButton){
        let tag = sender.tag - 1
        if resultLabel.text == "0"{
            resultLabel.text = "\(tag)"
        }
        else if let text = resultLabel.text {
            resultLabel.text = "\(text)\(tag)"
```

Also add targets to all buttons:

```
buttonX.addTarget(self, action: #selector(numberPressed(_:)), for:
.touchUpInside)
```

Now the number buttons we pressed are responding and displaying on screen:

number pressed responded and displayed.PNG

Use switch statement to enable each action:

```
            switch operation {
                case .add:
                    let result = firstNumber + secondNumber
                    resultLabel.text = "\(result)"
                    break
                case .subtract:
                    let result = firstNumber - secondNumber
                    resultLabel.text = "\(result)"
                    break
                case .multiply:
                    let result = firstNumber * secondNumber
                    resultLabel.text = "\(result)"
                    break
                case .divide:
                    let result = firstNumber / secondNumber
                    resultLabel.text = "\(result)"
                    break
                }
```

Now all mathematical operations work:

8+9=17 addition works.PNG

55-6=49 subtraction works.PNG

25x4=100 multiplication works.PNG

260/2=130 division works.PNG