import UIKit

import Charts

class ViewController: UIViewController, FrameExtractorDelegate {

var frameExtractor: FrameExtractor!

@IBOutlet weak var chtChart: LineChartView!

@IBOutlet weak var imageView: UIImageView!

@IBOutlet weak var btnStart: UIButton!

@IBOutlet weak var btnStop: UIButton!

@IBOutlet weak var timer: UILabel!

@IBOutlet weak var hratRate: UILabel!

@IBOutlet weak var atrialFibrillation: UILabel!

var totalTime = 120

var countdownTimer: Timer!

var no : [Double] = []

var result: [Double] = []

var x : Double = 0

override func viewDidLoad() {

super.viewDidLoad()

btnStart.layer.cornerRadius = 5

btnStart.layer.borderWidth = 3

btnStart.layer.borderColor = UIColor.purple.cgColor

btnStart.showsTouchWhenHighlighted = true

btnStop.layer.cornerRadius = 5

btnStop.layer.borderWidth = 3

btnStop.layer.borderColor = UIColor.purple.cgColor

btnStop.showsTouchWhenHighlighted = true

btnStop.isEnabled = false

}

override func didReceiveMemoryWarning() {

super.didReceiveMemoryWarning()

}

@IBAction func btnStart(\_ sender: UIButton) {

btnStop.isEnabled = true

btnStart.isEnabled = false

frameExtractor = FrameExtractor()

frameExtractor.delegate = self

startTimer()

}

func startTimer(){

countdownTimer = Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector(updateTime), userInfo: nil, repeats: true)

}

@objc func updateTime(){

timer.text = "\(timeFormatted(totalSeconds: totalTime))"

if totalTime != 0{

totalTime -= 1

}

else{

endTimer()

}

}

func endTimer(){

countdownTimer.invalidate()

frameExtractor.captureSession.stopRunning()

saveData(data: result)

}

func timeFormatted(totalSeconds: Int) -> String{

let sec: Int = totalSeconds % 60

let min: Int = (totalSeconds / 60) % 60

return String(format: "%02d:%02d", min, sec)

}

@IBAction func btnStop(\_ sender: UIButton) {

btnStop.isEnabled = false

endTimer()

}

func saveData(data: [Double]){

let sData = data

let date = Date()

let calender = Calendar.current

let formatter = DateFormatter()

formatter.dateFormat = "dd.MM.yy"

let hr = calender.component(.hour, from: date)

let min = calender.component(.minute, from: date)

let dt = formatter.string(from: date)

let fileName = "Data\_\(dt)\_\(hr):\(min)"

print(fileName)

let textData = "\(sData)"

if let dir = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first{

let fileURL = dir.appendingPathComponent("\(fileName).txt")

do{

try textData.write(to: fileURL, atomically: false, encoding: .utf8)

}

catch{

print(error)

}

}

let alert = UIAlertController(title: "Data Storage Alert", message: "Data Saved to Text File as \(fileName)", preferredStyle: UIAlertControllerStyle.alert)

alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.default, handler: nil))

self.present(alert, animated: true, completion: nil)

}

func captured(image: UIImage) {

imageView.image = image

let uiImg = image

var red : CGFloat = 0.0

let pixWidth = Int(uiImg.size.width)

let pixHeight = Int(uiImg.size.height)

let pixelData = uiImg.cgImage?.dataProvider?.data

let data: UnsafePointer<UInt8> = CFDataGetBytePtr(pixelData)

for x in 0..<pixWidth{

for y in 0..<pixHeight{

let point = CGPoint(x: x,y:y)

let pixelInfo: Int = ((pixWidth \* Int(point.y)) + Int(point.x)) \* 4

let redp = CGFloat(data[pixelInfo + 1]) / 255.0

red = red + redp

}

}

let finalresult = red / CGFloat(pixHeight\*pixWidth)

x = Double(finalresult)

let dataPlot = Double(x)*//finalresult)*

if(no.count > 100)

{

no.remove(at: 0)

}

no.append(dataPlot)

*//print(finalresult)*

updateGraph()

}

func updateGraph(){

var lineChartEntry = [ChartDataEntry]()

for i in 0..<no.count{

let value = ChartDataEntry(x: Double(i),y: Double(no[i]))

lineChartEntry.append(value)

}

let line1 = LineChartDataSet(values: lineChartEntry, label: "Value")

line1.colors = [NSUIColor.white]

line1.drawCirclesEnabled = false

line1.drawValuesEnabled = false

chtChart.backgroundColor = .red

chtChart.pinchZoomEnabled = true

chtChart.xAxis.enabled = false

chtChart.leftAxis.enabled = false

chtChart.rightAxis.enabled = false

chtChart.chartDescription?.text = "PPG Data"

let data = LineChartData()

data.addDataSet(line1)

chtChart.data = data

}

}