HIGHER SCHOOL OF ECONOMICS UNIVERSITY

HOMEWORK #3

dedicated to Research Seminar
«Development of applications for the Apple iOS platform»

MOSCOW 2022

Theme and Objective

Theme	Custom classes
Objective	Develop the functionality of the app by introducing new classes

Task description

You should implement 3 sliders to change app's background color.

Task requirements

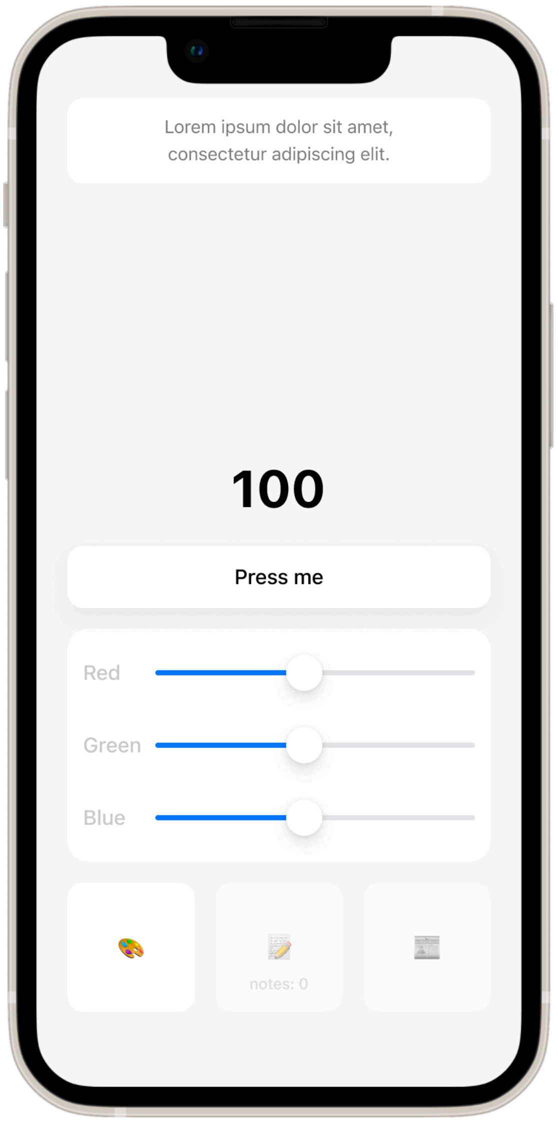
During this task:

- You are not allowed to use React, SwiftUI, Objc-C or anything except vanilla Swift.
- You cannot use StoryBoard to layout the app.
- You are allowed to use any architecture for this app.

Grading

Grade	Task
0	The task was not submitted or it contains nothing
1	The task has nothing to do with the assignment.
2	View appears when palette button is pressed
3	View consists of vertical stackview with 3 (R,G,B) horizontal stackviews (label & slider)
4	Sliders are interactive
5	App background changes color corresponding to sliders drags
6	View with colors can be closed using second tap on palette button
7	Color change is done using animation
8	Sliders values represents background color when it is tapped more than once
9	The app looks similar on devices with different sizes and elements don't overlap
10	Color palette button triggers haptic engine on press

Visualisation



Tutorial

Point 2

First lets create new file with class for ColorPaletteView with with stackView for controls and variable in which we will store color of the background.

```swift
final class ColorPaletteView: UIControl {
    private let stackView = UIStackView()
    private(set) var chosenColor: UIColor = .systemGray6

    init() {
        super.init(frame: .zero)

        setupView()
    }

    @available(*, unavailable)
    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }

    private func setupView() {

    }
}
```

Point 3

We need to initialise ColorPaletteView in our WelcomeViewController and provide function which opens and hides it.

```
let colorPaletteView = ColorPaletteView()
//

private func setupView() {
        view.backgroundColor = .systemGray6
        commentView.isHidden = true
        colorPaletteView.isHidden = true

        setupIncrementButton()
        setupValueLabel()
        setupMenuButtons()
        setupColorControlSV()
    }
//

private func setupMenuButtons() {
        let colorsButton = makeMenuButton(title: "🎨")
        colorsButton.addTarget(self, action: 
#selector(paletteButtonPressed), for: .touchUpInside)
//
}
```

```
private func setupColorControlSV() {
        colorPaletteView.isHidden = true
        view.addSubview(colorPaletteView)

colorPaletteView.translatesAutoresizingMaskIntoConstraints = 
false

        NSLayoutConstraint.activate([
            colorPaletteView.topAnchor.constraint(equalTo: 
incrementButton.bottomAnchor, constant: 8),
            colorPaletteView.leadingAnchor.constraint(equalTo: 
view.safeAreaLayoutGuide.leadingAnchor, constant: 24),
            colorPaletteView.trailingAnchor.constraint(equalTo: 
view.safeAreaLayoutGuide.trailingAnchor, constant: -24),
            colorPaletteView.bottomAnchor.constraint(equalTo: 
buttonsSV.topAnchor, constant: -8)
        ])
    }
//

@objc
    private func paletteButtonPressed() {
        colorPaletteView.isHidden = false
        let generator = UIImpactFeedbackGenerator(style: .medium)
        generator.impactOccurred()
    }
```

Point 4

To implement stackView with sliders lets write an extension to ColorPaletteView. It will be safer.

```
extension ColorPaletteView {
    private final class ColorSliderView: UIControl {
        private let slider = UISlider()
        private let colorLabel = UILabel()

        private(set) var value: Float

        init(colorName: String, value: Float) {
            self.value = value
            super.init(frame: .zero)

            slider.value = value
            colorLabel.text = colorName
            setupView()

        @available(*, unavailable)
        required init?(coder: NSCoder) {
            fatalError("init(coder:) has not been implemented")
        }

        private func setupView() {
            let stackView = UIStackView(arrangedSubviews:
[colorLabel, slider])
            stackView.axis = .horizontal
            stackView.spacing = 8

            addSubview(stackView)
            stackView.pin(to: self, [.left: 12, .top: 12, .right:
12, .bottom: 12])
        }
}
```

Point 5

To make them functionally enabled we need to add objc function and attach it to WelcomeViewController.

```
extension ColorPaletteView {
    private final class ColorSliderView: UIControl {
        private let slider = UISlider()
        private let colorLabel = UILabel()

        private(set) var value: Float

        init(colorName: String, value: Float) {
            self.value = value
            super.init(frame: .zero)

            slider.value = value
            colorLabel.text = colorName
            setupView()
            slider.addTarget(self, action:
#selector(sliderMoved(_:)), for: .touchDragInside)
        }

        @available(*, unavailable)
        required init?(coder: NSCoder) {
            fatalError("init(coder:) has not been implemented")
        }

        private func setupView() {
            let stackView = UIStackView(arrangedSubviews:
[colorLabel, slider])
            stackView.axis = .horizontal
            stackView.spacing = 8

            addSubview(stackView)
            stackView.pin(to: self, [.left: 12, .top: 12, .right:
12, .bottom: 12])
        }

        @objc
        private func sliderMoved(_ slider: UISlider) {
            self.value = slider.value
            sendActions(for: .touchDragInside)
        }
    }
}
```

```swift
//Color Palette View

private func setupView() {
        let redControl = ColorSliderView(colorName: "R", value:
Float(chosenColor.redComponent))
        let greenControl = ColorSliderView(colorName: "G", value:
Float(chosenColor.greenComponent))
        let blueControl = ColorSliderView(colorName: "B", value:
Float(chosenColor.blueComponent))
        redControl.tag = 0
        greenControl.tag = 1
        blueControl.tag = 2

        stackView.axis = .vertical
        stackView.distribution = .equalSpacing
        stackView.addArrangedSubview(redControl)
        stackView.addArrangedSubview(greenControl)
        stackView.addArrangedSubview(blueControl)
        stackView.backgroundColor = .white
        stackView.layer.cornerRadius = 12

        [redControl, greenControl, blueControl].forEach {
            $0.addTarget(self, action: #selector(sliderMoved(slider:)),
for: .touchDragInside)
        }
        addSubview(stackView)
        stackView.pin(to: self)
    }
    @objc
    private func sliderMoved(slider: ColorSliderView) {
        switch slider.tag {
        case 0:
            self.chosenColor = UIColor(
                red: CGFloat(slider.value),
                green: chosenColor.greenComponent,
                blue: chosenColor.blueComponent,
                alpha: chosenColor.alphaComponent
            )
        case 1:
            self.chosenColor = UIColor(
                red: chosenColor.redComponent,
                green: CGFloat(slider.value),
                blue: chosenColor.blueComponent,
                alpha: chosenColor.alphaComponent
            )
        default:
            self.chosenColor = UIColor(
                red: chosenColor.redComponent,
                green: chosenColor.greenComponent,
                blue: CGFloat(slider.value),
                alpha: chosenColor.alphaComponent
            )
        }
        sendActions(for: .touchDragInside)
    }
```

Point 6

To enable such functionality we need to create function in WelcomeViewController in which we will toggle isHidden state.

```swift
@objc
    private func paletteButtonPressed() {
        colorPaletteView.isHidden.toggle()
        let generator = UIImpactFeedbackGenerator(style: .medium)
        generator.impactOccurred()
    }
```

Point 7

Lets higher the UX by adding animation to background color change.

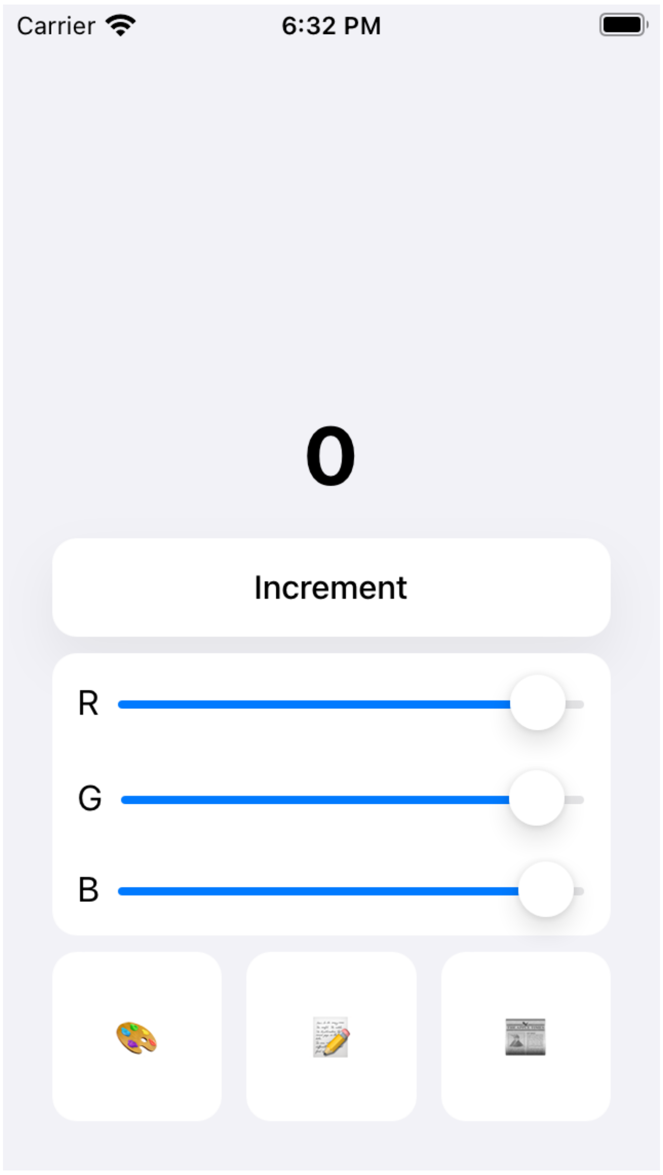
```swift
@objc
    private func changeColor(_ slider: ColorPaletteView) {
        UIView.animate(withDuration: 0.5) {
            self.view.backgroundColor = slider.chosenColor
        }
    }
```

Point 8

Such functionality is a already provided in previous points.

Conclusion

We created a nice looking interactive app.



Info

If your project contains interesting solutions, exceptional code style, good presentation, unique references or well designed clean code (patterns, paradigms) up to two bonus points might be added. This means the maximum grade a student can receive per homework is 12. If interesting solution repeats in different projects bonus points won't be added. It is pointless to try and prove to us that your solution is interesting. There won't be any appeals regarding bonus points.

If the task is overdue, for each day after deadline your maximum grade goes down by 10%. Bonus points might not be added for projects submitted late. The maximum penalty is 50%. Outdated submission can get you up to 5 points, but if you already submitted and received a grade you can submit points you didn't manage to do in time. Therefor you can make your grades better.

The grade will be decreased for code style violation and bad code. If you stumble across any questions feel free to contact teachers or assistants! Our goal is to help you develop skills required to create iOS Apps.