HIGHER SCHOOL OF ECONOMICS UNIVERSITY

HOMEWORK #2

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

MOSCOW 2022

Theme and Objective

Theme	Simple project creation
Objective	Mastering programming layout creation and stack view usage

Task description

You should create a simple app without storyboard.

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	The project has button and label (with integer value)
3	When the button is pressed, label changes (number increases)
4	Button can be used more than once
5	«Comment» label is created (pinned to the top of the device)
6	«Comment» label changes corresponding to the value of the integer label
7	«Comment» label changes with animation
8	Stack view with buttons is created
9	The app looks similar on devices with different sizes
10	Shadows for button is added using CALayer extension /or/ any button triggers haptic response on click

Visualisation

Tutorial:

Point 1

First, we need to delete storyboard from our project and make it possible to set UI without storyboard at all. We need to delete storyboard from our left menu. Then change SceneDelegate file as follows:

```
var window: UIWindow?


        func scene(_ scene: UIScene, willConnectTo session:
UISceneSession, options connectionOptions:
UIScene.ConnectionOptions) {
            guard let windowScene = (scene as? UIWindowScene)
else { return }
            let window = UIWindow(windowScene: windowScene)

            let navigationController =
UINavigationController(rootViewController:
WelcomeViewController())
            window.rootViewController = navigationController
            self.window = window
            window.makeKeyAndVisible()
        }
```

Besides that, we should delete Main Storyboard from info.plist and delete it from target > Build Settings > Info.plist Values.

Point 2

For convenience, It's better to setup every distinct element in a separate function

```
final class WelcomeViewController: UIViewController {

    private let commentLabel = UILabel()
    private let valueLabel = UILabel()

     private let value: Int = 0
}
```

For convenience, It's better to setup every distinct element in a separate function. And call each of them in setupView() function, which will be called in viewDidLoad().

```
private func setupIncrementButton() {
        incrementButton.setTitle("Increment", for: .normal)
        incrementButton.setTitleColor(.black, for: .normal)
        incrementButton.layer.cornerRadius = 12
        incrementButton.titleLabel?.font = .systemFont(ofSize:
16.0, weight: .medium)
        incrementButton.backgroundColor = .white
        incrementButton.layer.applyShadow()

        self.view.addSubview(incrementButton)
        incrementButton.setHeight(to: 48)
        incrementButton.pinTop(to: self.view.centerYAnchor)
        incrementButton.pin(to: self.view, [.left: 24, .right:
24])

        incrementButton.addTarget(self, action:
#selector(incrementButtonPressed), for: .touchUpInside)
    }
```

```
private func setupValueLabel() {
        valueLabel.font = .systemFont(ofSize: 40.0,
weight: .bold)
        valueLabel.textColor = .black
        valueLabel.text = "\(value)"

        self.view.addSubview(valueLabel)
        valueLabel.pinBottom(to: incrementButton.topAnchor, 16)
        valueLabel.pinCenter(to: self.view.centerXAnchor)
    }
```

```
private func setupView() {
        view.backgroundColor = .systemGray6

        setupIncrementButton()
        setupValueLabel()
    }
```

Point 3

The next task is to change the label, when we press the button. We need to implement the function which will be incrementing our value and updating the appearance. Function should be linked with the button we created previously.

```
@objc
private func incrementButtonPressed() {
    value += 1
    updateUI()
}
```

Point 4

Already done in previous point.

Point 5

We need to create the view, attach label inside of it. The view should be constrained with respect of device screen size, so we need to take it into account.

```
private func setupCommentView() -> UIView {
        let commentView = UIView()
        commentView.backgroundColor = .white
        commentView.layer.cornerRadius = 12

        view.addSubview(commentView)
        commentView.pinTop(to:
self.view.safeAreaLayoutGuide.topAnchor)
        commentView.pin(to: self.view, [.left: 24, .right: 24])

        commentLabel.font = .systemFont(ofSize: 14.0,
weight: .regular)
        commentLabel.textColor = .systemGray
        commentLabel.numberOfLines = 0
        commentLabel.textAlignment = .center

        commentView.addSubview(commentLabel)
        commentLabel.pin(to: commentView, [.top: 16, .left:
16, .bottom: 16, .right: 16])

        return commentView
    }
```

Point 6

Comment view should contain motivational message to inspire the user to tap the button more and more. For such task it would be cool to change the message in the comment based on the value passed.

```
func updateCommentLabel(value: Int) {
        switch value {
        case 0...10:
            commentLabel.text = «1"
        case 10...20:
            commentLabel.text = "2"
        case 20...30:
            commentLabel.text = "3"
        case 30...40:
            commentLabel.text = "4"
        case 40...50:
            commentLabel.text = "🎉🎉🎉🎉🎉🎉🎉🎉🎉🎉"
        case 50...60:
            commentLabel.text = "big boy"
        case 60...70:
            commentLabel.text = "70 70 70 moreeeee"
        case 70...80:
            commentLabel.text = "⭐⭐⭐⭐⭐⭐⭐⭐⭐"
        case 80...90:
            commentLabel.text = "80+\n go higher!"
        case 90...100:
            commentLabel.text = "100!! to the moon!!"
        default:
            break
        }
    }
```

Point 7

To make ui updates more comfortable for the user we should add animations for comment view changes.

```
@objc
    private func incrementButtonPressed() {
        value += 1

        let generator = UIImpactFeedbackGenerator(style: .light)
        generator.impactOccurred()

        UIView.animate(withDuration: 1) {
            self.updateUI()
        }
    }
```

Point 8

There should be 3 buttons (for further development). Up to now we should implement just blanc buttons with no functions triggered by them. Buttons should be located in the stack view.

First, lets create function which produces the UIButton with desired title.

```
    private func makeMenuButton(title: String) -> UIButton {
        let button = UIButton()
        button.setTitle(title, for: .normal)
        button.setTitleColor(.black, for: .normal)
        button.layer.cornerRadius = 12
        button.titleLabel?.font = .systemFont(ofSize: 16.0,
weight: .medium)
        button.backgroundColor = .white
        button.heightAnchor.constraint(equalTo:
button.widthAnchor).isActive = true

        return button
    }
```

Then, we should organise it nicely in UIStackView()

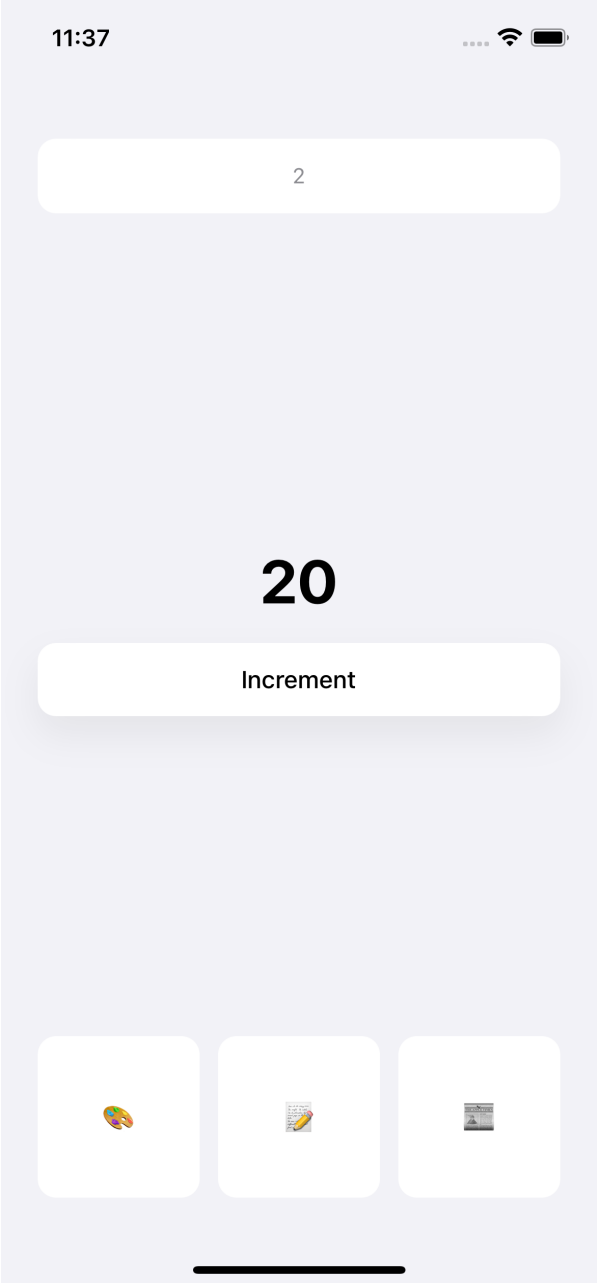
```
private func setupMenuButtons() {
        let colorsButton = makeMenuButton(title: "🎨")
        let notesButton = makeMenuButton(title: "📝")
        let newsButton = makeMenuButton(title: "🏞")

        let buttonsSV = UIStackView(arrangedSubviews:
[colorsButton, notesButton, newsButton])
        buttonsSV.spacing = 12
        buttonsSV.axis = .horizontal
        buttonsSV.distribution = .fillEqually

        self.view.addSubview(buttonsSV)
        buttonsSV.pin(to: self.view, [.left: 24, .right: 24])
        buttonsSV.pinBottom(to:
self.view.safeAreaLayoutGuide.bottomAnchor, 24)
    }
```

Conclusion

As a result, we learnt how to build a pretty looking app without storyboard at all and filled the app with basic functionality.



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.