

// Storyboards

Home.storyboard

Search.storyboard

NewPost.storyboard & make initial view for every story board, set class, and storyboard ID

Profile.storyboard

Activity.storyboard

// View Controllers

HomeViewController.swift

SearchViewController.swift

NewPostViewController.swift

ProfileViewController.swift

ActivityViewController.swift

// set up the tab bar in AppDelegate

// FeedData.swift (data)

import UIKit

struct Post {

var postImage: UIImage

var postComment: String

var user: User

var commentCount: Int

var likesCount: Int

var datePosted: String

}

struct Story {

var post: Post

}

class Model {

var postList: [Post] = [Post]()

var storyList: [Story] = [Story]()

init() {

let user1 = User(name: "John Carmack", profileImage: UIImage(named: "user1")!)

↓ user 2

let post1 = Post(postImage: UIImage(named: "destination1")!, postComment: "----", user: user1, commentCount: 5, likesCount: 10, datePosted: "3 Sept")

postList.append(post1)

↓

AppDelegate.swift

```
import UIKit  
@UIApplicationMain  
class AppDelegate: UIResponder, UIApplicationDelegate {  
    var window: UIWindow?  
    let tab  
    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:  
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {  
        let tabBarController = UITabBarController()  
        let homeStoryboard = UIStoryboard(name: "Home", bundle: nil)  
        let searchStoryboard = UIStoryboard(name: "Search", bundle: nil)  
        ↓ other storyboards
```

let homeVC = homeStoryboard.instantiateViewController(withIdentifier: "Home") as! HomeViewController

let profileVC = ...

↓ other viewControllers

```
let vcData: [[UIViewController, UIImage, UIImage]] = [  
    (homeVC, UIImage(named: "home-tab-icon")!, UIImage(named: "home-selected-tab-icon")!),  
    ↓ for other VCs  
]
```

```
let vcs = vcData.map { (vc, defaultImage, selectedImage) -> UINavigationController in  
    let nav = UINavigationController(rootViewController: vc)  
    nav.tabBarItem.image = defaultImage  
    nav.tabBarItem.selectedImage = selectedImage  
    return nav
```

}

```
tabController.viewControllers = vcs  
tabController.tabBar.isTranslucent = false  
tabController.delegate = tabBarControllerDelegate  
if let items = tabController.tabBar.items {
```

for item in items {

if let image = item.image {

item.image = image.withRenderingMode(UIImage.RenderingMode.alwaysOriginal)

}

if let selectedImage = item.selectedImage {

item.selectedImage = selectedImage.withRenderingMode(UIImage.RenderingMode.alwaysOriginal)

3

item.imageInsets = UIEdgeInsets(top: 6, left: 0, bottom: -6, right: 0) // hide the title

3

UINavigationBar.appearance().backgroundColor = UIColor.white

window?.rootViewController = tabController

return true

3

11 Collection Layout .swift

```
import UIKit

class CollectionLayout : UICollectionViewLayout {
    fileprivate var numberOfColumns: Int = 3
    fileprivate var cellPadding: CGFloat = 3
    fileprivate var cache = [UICollectionViewLayoutAttributes]()
    fileprivate var contentWidth: CGFloat {
        guard let collectionView = collectionView else {
            return 0
        }
        let insets = collectionView.contentInset
        return collectionView.bounds.width - (insets.left + insets.right) - (cellPadding * (CGFloat(numberOfColumns) - 1))
    }
    fileprivate var contentHeight: CGFloat = 0
    override var collectionViewContentSize: CGSize {
        return CGSize(width: contentWidth, height: contentHeight)
    }
    override func prepare() {
        guard cache.isEmpty == true, let collectionView = collectionView else {
            return
        }
        let itemsPerRow: Int = 3
        let normalColumnWidth: CGFloat = contentWidth / CGFloat(itemsPerRow)
        let normalColumnHeight: CGFloat = normalColumnWidth
        let featuredColumnWidth: CGFloat = (normalColumnWidth * 2) + cellPadding
        let featuredColumnHeight: CGFloat = featuredColumnWidth
        var xOffsets: [CGFloat] = [CGFloat]()
        for item in 0..<6 { // also is items per row
            let multiplier = item % 3
            let xPos = CGFloat(multiplier) * (normalColumnWidth + cellPadding)
            xOffsets.append(xPos)
        }
        xOffsets.append(0.0)
        for _ in 0..<2 {
            xOffsets.append(featuredColumnWidth + cellPadding)
        }
        for item in 0...9 {
            var yPos = floor(Double(item / 3)) * (Double(normalColumnHeight) + Double(cellPadding))
            if item == 8 {
                yPos += (Double(normalColumnHeight) + Double(cellPadding))
            }
            yOffsets.append(CGFloat(-yPos))
        }
        let numberOfRowsInSection: Int = 9 // not the section in actual collection view, the part with repeatable pattern
        let heightOfSection: CGFloat = 4 * normalColumnHeight + (4 * cellPadding)
        var itemInSection: Int = 0
        for item in 0..
```

```
let multiplier:Double = floor(Double(item) / Double(numberOfItemsPerSection))
let yPos = yOffsets[itemInSection] + (heightOfSection * CGFloat(multiplier))
var cellWidth = normalColumnWidth
var cellHeight = normalColumnHeight
if (itemInSection + 1) % 7 == 0 && itemInSection != 0 {
    cellWidth = featuredColumnWidth
    cellHeight = featuredColumnHeight
}
let frame = CGRect(x:xPos, y:yPos, width: cellWidth, height: cellHeight)
let attributes = UICollectionViewLayoutAttributes(forCellWith: indexPath)
attributes.frame = frame
cache.append(attributes)
contentHeight = max(contentHeight, frame.maxY)
itemInSection = itemInSection < (numberOfItemsPerSection - 1) ? (itemInSection + 1) : 0
}
```

```
override func layoutAttributesForElements(in rect: CGRect) -> [UICollectionViewLayoutAttributes]? {
    var visibleLayoutAttributes = [UICollectionViewLayoutAttributes]()
    for attributes in cache {
        if attributes.frame.intersects(rect) {
            visibleLayoutAttributes.append(attributes)
        }
    }
    return visibleLayoutAttributes
}
```

```
override func layoutAttributesForItem(at indexPath: IndexPath) -> UICollectionViewLayoutAttributes? {
    return cache[indexPath.item]
}
```

}

}

117. Tab Bar Delegate.swift

import UIKit

```
class TabBarDelegate : NSObject, UITabBarControllerDelegate {
    func tabBarController(_ tabBarController: UITabBarController, didSelect viewController: UIViewController) {
        let navigationController = viewController as? UINavigationController
        _ = navigationController?.popViewController(animated: false)
    }

    func tabBarController(_ tabBarController: UITabBarController, shouldSelect viewController: UIViewController) -> Bool {
        let selectedViewController = tabBarController.selectedViewController
        guard let _selectedViewController = selectedViewController else {
            return false
        }
        if viewController == _selectedViewController {
            return false
        }

        guard let controllerIndex = tabBarController.viewControllers?.index(of: viewController) else {
            return true
        }

        if controllerIndex == 2 {
            let newPostStoryboard = UIStoryboard(name: "New Post", bundle: nil)
            let newPostVC = newPostStoryboard.instantiateViewController(withIdentifier: "New Post") as! NewPostViewController
            let navController = UINavigationController(rootViewController: newPostVC)
            _selectedViewController.present(navController, animated: true, completion: nil)
            return false
        }

        let navigationController = viewController as? UINavigationController
        _ = navigationController?.popToRootViewController(animated: false)
        return true
    }
}
```

// CustomSegmentedControl.swift

import UIKit

protocol ActivityDelegate : class {

func scrollTo(index: Int)

}

class CustomSegmentedControl : UIView {

var buttons = [UIButton]()

var selector: UIView!

var selectedSegmentedIndex = 0

let buttonTitles: [String] = ["Following", "You"]

let textColor: UIColor = .lightGray

let selectorColor: UIColor = .black

let selectorTextColor: UIColor = .black

weak var delegate: ActivityDelegate?

override func awakeFromNib() {

super.awakeFromNib()

updateView()

}

func updateView() {

buttons.removeAll()

subviews.forEach { (view) in

view.removeFromSuperview()

}

for buttonTitle in buttonTitles {

let button = UIButton(type: .system)

button.setTitle(buttonTitle, for: .normal)

button.setTitleColor(textColor, for: .normal)

button.addTarget(self, action: #selector(buttonTapped(button:)), for: .touchUpInside)

buttons.append(button)

}

buttons[0].setTitleColor(selectorTextColor, for: .normal)

let selectorWidth = frame.width / CGFloat(buttonTitles.count)

let y = (self.frame.maxY - self.frame.minY) - 2.0

selector = UIView.init(frame: CGRect.init(x: 0, y: y, width: selectorWidth, height: 2.0))

selector.backgroundColor = selectorColor

addSubview(selector)

let stackView = UIStackView(arrangeSubviews: buttons)

stackView.axis = .horizontal

stackView.alignment = .fill

stackView.distribution = .fillEqually

stackView.spacing = 0.0

addSubview(stackView)

stackView.translatesAutoresizingMaskIntoConstraints = false

stackView.topAnchor.constraint(equalTo: self.topAnchor).isActive = true

↓ similar for bottom, (left, right)

3

```
@objc func buttonTapped (button: UIButton) {  
    for (buttonIndex, btn) in buttons.enumerated () {  
        btn.setTitleColor (textColor, for: .normal)  
        if btn == button {  
            selectedSegmentIndex = buttonIndex  
            delegate?.scrollTo (index: selectedSegmentIndex)  
        }  
    }  
}
```

```
func updateSegmentedControlSegs (index: Int) {  
    for btn in buttons {  
        btn.setTitleColor (textColor, for: .normal)  
    }  
    let selectorStartPosition = frame.width / CGFloat (buttons.count) * CGFloat (index)  
    UIView.animate (withDuration: 0.3) {  
        self.selector.frame.origin.y = selectorStartPosition  
    }  
    buttons [index].setTitleColor (selectorTextColor, for: .normal)  
}
```

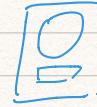
// StoryCollectionViewCell.swift (Views) also create XIB file

```
import UIKit
```

class StoryCollectionViewCell: UICollectionViewCell {
 @IBOutlet weak var storyImage: UIImageView!
 @IBOutlet weak var userNameLabel: UILabel!

override func awakeFromNib() {
 super.awakeFromNib()
 storyImage.layer.cornerRadius = storyImage.frame.width / 2
 storyImage.layer.masksToBounds = true
 storyImage.layer.borderColor = UIColor.white.cgColor
 storyImage.layer.borderWidth = CGFloat(2.0)
}

* change the reuseIdentifier to "StoryCollectionViewCell"
width = 90 height = 110
background color → white
* image 8-9-8 ratio → 1:1
* add a label
background color → F7F7F7



// StoriesTableViewCell.swift (Views) * also create XIB file * height 125 F7F7F7

```
import UIKit
```

class StoriesTableViewCell: UITableViewCell, UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout {
 lazy var stories: [Story] = {
 let model = Model()
 return model.storyList
 }()
}

@IBOutlet weak var collectionView: UICollectionView!

override func awakeFromNib() {
 super.awakeFromNib()
 selectionStyle = UITableViewCellSelectionStyle.none
 collectionView.register(UINib(nibName: "StoryCollectionViewCell", bundle: nil), forCellWithReuseIdentifier: "StoryCollectionViewCell")
 collectionView.delegate = self
 collectionView.dataSource = self
}

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
 return stories.count
}

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
 let cell = collectionView.dequeueReusableCell(withIdentifier: "StoryCollectionViewCell", for: indexPath) as! StoryCollectionViewCell
 cell.storyImage.image = stories[indexPath.row].post.postImage
 cell.userNameLabel.text = stories[indexPath.row].post.user.name
 return cell
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
 return CGSize(width: 90, height: 110)
}

// FeedTableViewCell.swift (Views) *also create XIB

import UIKit

class FeedTableViewCell: UITableViewCell {

@IBOutlet weak var profileImage: UIImageView!

↓ other buttons, labels, image views

override func awakeFromNib() {

super.awakeFromNib()

profileImage.layer.cornerRadius = profileImage.frame.width / 2

profileImage.layer.masksToBounds = true

selectionStyle = UITableViewCellSelectionStyle.None

}

* change the XIB reuse identifier

width 599, height 685

* profile image, constrain to margins not selected
control drag no less than 10px, button → more-icon



// ExploreCollectionViewCell.swift (Views)

import UIKit

class ExploreCollectionViewCell: UICollectionViewCell {

@IBOutlet weak var exploreImage: UIImageView!

3

// SearchBarContainerView.swift (Views)

import UIKit

class SearchBarContainerView: UIView {

let searchBar: UISearchBar

init(customSearchBar: UISearchBar) {

searchBar = customSearchBar

super.init(frame: CGRect.zero)

addSubview(searchBar)

3

convenience override init(frame: CGRect) {

self.init(customSearchBar: UISearchBar())

self.frame = frame

3

required init?(coder aDecoder: NSCoder) {

fatalError("init(coder:) has not been implemented")

3

override func layoutSubviews() {

super.layoutSubviews()

searchBar.frame = bounds

3

3

// Photo Collection View Cell.swift (views)

```
import UIKit
```

```
class PhotoCollectionViewCell: UICollectionViewCell {  
    @IBOutlet weak var photoImageView: UIImageView!  
}
```

// Activity Table View Cell.swift (Views)

```
import UIKit
```

```
class ActivityTableViewCell: UITableViewCell {  
    @IBOutlet weak var profileImage: UIImageView!  
    @IBOutlet weak var detailsLabel: UILabel!
```

```
override func awakeFromNib(){
```

```
super.awakeFromNib()
```

```
profileImage.layer.masksToBounds = true
```

```
selectionStyle = .none
```

```
}
```

```
override func layoutSubviews(){
```

```
super.layoutSubviews()
```

```
profileImage.layer.cornerRadius = profileImage.frame.width/2
```

```
}
```

// Activity View.swift (views)

```
import UIKit
```

```
class ActivityView : UIView, UITableViewDelegate, UITableViewDataSource {
```

```
    @IBOutlet weak var tableView: UITableView!
```

```
    var activityData: [Activity] = [Activity]() {
```

```
        didSet {
```

```
            tableView.reloadData()  
        }
```

```
}
```

```
override func awakeFromNib() {
```

```
    super.awakeFromNib()  
}
```

```
tableView.register(UINib(nibName: "ActivityTableViewCell", bundle: nil), forCellReuseIdentifier: "ActivityTableViewCell")
```

```
tableView.delegate = self
```

```
tableView.dataSource = self
```

```
tableView.tableFooterView = UIView()
```

```
}
```

```
func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {  
    return CGFloat(80.0)
```

```
}
```

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {  
    return activityData.count  
}
```

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "Activity TableView Cell") as! ActivityTableViewCell  
    cell.profileImage.image = activityData[indexPath.row].userImage  
    cell.detailsLabel.text = activityData[indexPath.row].details  
    return cell
```

}

}

// HomeView Controller.swift (ViewControllers)

```
import UIKit
```

```
class HomeViewController : UIViewController, UITableViewDelegate, UITableViewDataSource {
```

```
    @IBOutlet weak var tableView : UITableView!
```

```
    lazy var posts : [Post] = {
```

```
        let model = Model()
```

```
        return model.postList
```

```
}()
```

```
override func viewDidLoad() {
```

```
    super.viewDidLoad()
```

```
    tableView.estimatedRowHeight = CGFloat(88.0)
```

```
    tableView.rowHeight = UITableView.automaticDimension
```

} auto adjust row height

```
    tableView.register(UINib(nibName: "FeedTableViewCell", bundle: nil), forCellReuseIdentifier: "FeedTableViewCell")
```

```
    tableView.register(UINib(nibName: "StoriesTableViewCell", bundle: nil), forCellReuseIdentifier: "StoriesTableViewCell")
```

```
    tableView.dataSource = self
```

```
    tableView.delegate = self
```

```
    tableView.tableFooterView = UIView()
```

```
    var rightBarItemImage = UIImage(named: "send-nav-icon")
```

```
    rightBarItemImage = rightBarItemImage?.withRenderingMode(.alwaysOriginal)
```

```
    self.navigationItem.rightBarButtonItem = UIBarButtonItem(image: rightBarItemImage, style: .plain, target: nil, action: nil)
```

```
    var leftBarItemImage = UIImage(named: "camera-nav-icon")
```

```
    leftBarItemImage = leftBarItemImage?.withRenderingMode(.alwaysOriginal)
```

```
    self.navigationItem.rightBarButtonItem = UIBarButtonItem(image: leftBarItemImage, style: .plain, target: nil, action: nil)
```

```
    let profileImageView = UIImageView(image: UIImage(named: "logo-nav-icon"))
```

```
    self.navigationItem.titleView = profileImageView
```

```
}
```

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
```

```
    return posts.count + 1
```

```
}
```

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
```

```
    if indexPath.row == 0 {
```

```
        let cell = tableView.dequeueReusableCell(withIdentifier: "StoriesTableViewCell") as! StoriesTableViewCell
```

```
        return cell
```

```
}
```

```
    let cell = tableView.dequeueReusableCell(withIdentifier: "FeedTableViewCell") as! FeedTableViewCell
```

```
    let currentIndex = indexPath.row - 1
```

```
    let postData = posts[currentIndex]
```

```
    cell.profileImage.image = postData.user.profileImage
```

```
    ↴ similar ones
```

```
    cell.usernameTitleButton.setTitle(postData.user.name, for: .normal)
```

```
    return cell
```

```
}
```

// SearchViewController.swift (view Controllers) * set the identifier and class for cells

import UIKit

class SearchViewController: UIViewController, UICollectionViewDelegate, UICollectionViewDataSource {

@IBOutlet weak var collectionView: UICollectionView!

var searchController: UISearchController!

lazy var posts: [Post] = {

let model = Model()

return model.postsList

}

* change to custom for the layout option

* Collection View ⇒ Custom ⇒ class: CollectionLayout

override func viewDidLoad() {

super.viewDidLoad()

collectionView.delegate = self

collectionView.dataSource = self

searchController = UISearchController(searchResultsController: nil)

searchController.obscuresBackgroundDuringPresentation = true

searchController.searchBar.showsCancelButton = false

for subView in searchController.searchBar.subviews {

for subView1 in subView.subviews {

if let textField = subView1 as? UITextField {

textField.backgroundColor = UIColor(red: 0.93, green: 0.93, blue: 0.93, alpha: 1.0)

textField.textAlignment = NSTextAlignment.center

}

}

searchController.dimsBackgroundDuringPresentation = false

searchController.definesPresentationContext = true

searchController.hidesNavigationBarDuringPresentation = false

let searchBarContainer = SearchBarContainerView(customSearchBar: searchController.searchBar)

searchBarContainer.frame = CGRect(x: 0, y: 0, width: view.frame.width, height: 44)

navigationItem.titleView = searchBarContainer

}

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {

return posts.count

}

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {

let cell = collectionView.dequeueReusableCell(withIdentifier: "ExploreCollectionViewCell", for: indexPath) as!

ExploreCollectionViewCell

cell.exploreImage.image = posts[indexPath.row].postImage

return cell

3

11 NewPostViewController.swift (View Controller)

```
import UIKit
```

```
enum NewPostPagesToShow : Int {
```

```
    case library, camera
```

```
    var identifier : String {
```

```
        switch self {
```

```
        case .library:
```

```
            return "Photo Library VC"
```

```
        case .camera:
```

```
            return "Camera VC"
```

```
}
```

```
}
```

```
static func pagesToShow() -> [NewPostPagesToShow] {
```

```
    return [.library, .camera]
```

```
}
```

```
}
```

```
class NewPostViewController : UIViewController {
```

```
override func viewDidLoad() {
```

```
    super.viewDidLoad()
```

```
    navigationItem.leftBarButtonItem = UIBarButtonItem(title: "cancel", style: .plain, target: self, action: #selector(cancelButtonTapped))
```

```
    navigationController?.navigationBar.titleTextAttributes =
```

```
        [NSAttributedString.Key.foregroundColor: UIColor.black]
```

```
}
```

```
override var prefersStatusBarHidden : Bool {
```

```
    return true
```

```
}
```

```
@objc func cancelButtonPressed(_ sender : Any) {
```

```
    dismiss(animated: true, completion: nil)
```

```
}
```

```
@IBAction func libraryButtonDidTouch(_ sender : Any) {
```

```
    NotificationCenter.default.post(name: NSNotification.Name("newPage"), object: NewPostPagesToShow.library)
```

```
}
```

```
@IBAction func photoButtonDidTouch(_ sender : Any) {
```

```
    NotificationCenter.default.post(name: NSNotification.Name("newPage"), object: NewPostPagesToShow.camera)
```

```
}
```

// NewPostPageViewController.swift (ViewControllers)

import UIKit

```
class NewPostPageViewController : UIPageViewController, UIPageViewControllerDelegate {
    var orderedViewControllers : [UIViewController] = [UIViewController]()
    var pagesToShow : [NewPostPagesToShow] = NewPostPagesToShow.pagesToShow()
    var currentIndex : Int = 0
    override func viewDidLoad() {
        super.viewDidLoad()
        delegate = self
        dataSource = self
        for pageToShow in pagesToShow {
            let page = newViewController(pageToShow: pageToShow)
            orderedViewControllers.append(page)
        }
    }
    if let firstViewController = orderedViewControllers.first {
        setViewControllers([firstViewController], direction: UIPageViewController.NavigationDirection.forward, animated: true, completion: nil)
    }
    NotificationCenter.default.addObserver(self, selector: #selector(NewPostPageViewController.newPage(notification:)), name: NSNotification.Name(rawValue: "newPage"), object: nil)
}
@objc func newPage(notification: NSNotification) {
    if let receivedObject = notification.object as? NewPostPagesToShow {
        showViewController(index: receivedObject.rawValue)
    }
}
private func newViewController(pageToShow: NewPostPagesToShow) -> UIViewController {
    var viewController : UIViewController!
    let newPostStoryboard = UIStoryboard(name: "NewPost", bundle: nil)
    switch pageToShow {
    case .camera:
        viewController = newPostStoryboard.instantiateViewController(withIdentifier: pageToShow.identifier) as! CameraViewController
    case .library:
        viewController = newPostStoryboard.instantiateViewController(withIdentifier: pageToShow.identifier) as! PhotoLibraryViewController
    }
    return viewController
}
func showViewController(index: Int) {
    if currentIndex > index {
        setViewControllers([orderedViewControllers[index]], direction: UIPageViewController.NavigationDirection.reverse, animated: true, completion: nil)
    } else if currentIndex < index {
        setViewControllers([orderedViewControllers[index]], direction: UIPageViewController.NavigationDirection.forward, animated: true, completion: nil)
    }
    currentIndex = index
}
```

deinit {

 NotificationCenter.default.removeObserver(self)

 NotificationCenter.default.removeObserver(self, name: NSNotification.Name(rawValue: "newPage", objectin: nil))

}

3

extension NewPostPageViewController : UIPageViewControllerDataSource {

 // return the view controller after the given view controller

func pageViewController(_ pageViewController: UIPageViewController, viewControllerAfter viewController: UIViewController) -> UIViewController?

 guard let viewControllerIndex = orderedViewControllers.index(of: viewController) else {
 return nil

 3

 let nextIndex = viewController + 1

 let orderedViewControllerCount = orderedViewControllers.count

 guard orderedViewControllerCount != nextIndex else {
 return nil

 3

 guard orderedViewControllerCount > nextIndex else {

 return nil

 3

 return orderedViewControllers[nextIndex]

 3

func pageViewController(_ pageViewController: UIPageViewController, viewControllerBefore viewController: UIViewController) -> UIViewController?

 guard let viewControllerIndex = orderedViewControllers.index(of: viewController) else {
 return nil

 3

 let previousIndex = viewController - 1

 guard previousIndex >= 0 else {

 return nil

 3

 return orderedViewControllers[previousIndex]

 3

// CameraViewController.swift (ViewControllers)

import UIKit

class CameraViewController : UIViewController {

@IBOutlet weak var simpleCameraView: SimpleCameraView!

var simpleCamera: SimpleCamera!

override func viewDidLoad() {

super.viewDidLoad()

simpleCamera = SimpleCamera(cameraView: simpleCameraView)

}

override func viewDidAppear(_ animated: Bool) {

super.viewDidAppear(animated)

simpleCamera.startSession()

}

override func viewWillDisappear(_ animated: Bool) {

super.viewWillDisappear(animated)

simpleCamera.stopSession()

}

@IBAction func startCapture(_ sender: Any) {

if simpleCamera.currentCaptureMode == .photo {

simpleCamera.takePhoto { [image, success] in

if success {

print("image success")

}

}

}

}

plist ⇒ Camera usage

Microphone

Photo Library

II Photo Library View Controller.swift

(ViewControllers)

```
import UIKit
import Photos

class PhotoLibraryViewController: UIViewController, UICollectionViewDataSource, UICollectionViewDelegate, UICollectionViewDelegateFlowLayout {

    @IBOutlet weak var photosCollectionView: UICollectionView!
    var images = [PHAsset]()
    Alt + click → A representation of an image, video, or live photo in the photos library
    * change reuse identifier

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.

        photosCollectionView.delegate = self
        photosCollectionView.dataSource = self

        if PHPPhotoLibrary.authorizationStatus() == .authorized {
            getImages()
        } else {
            PHPPhotoLibrary.requestAuthorization { (status) in
                switch status {
                    case .authorized:
                        DispatchQueue.main.async {
                            self.getImages()
                        }
                    case .denied, .notDetermined, .restricted:
                        return
                }
            }
        }
    }

    func getImages() {
        let assets = PHAsset.fetchAssets(with: PHAssetMediaType.image, options: nil)
        assets.enumerateObjects({ (object, count, stop) in
            // self.cameraAssets.add(object)
            self.images.append(object)
        })
        // In order to get latest image first, we just reverse the array
        self.images.reverse()
        // To show photos, I have taken a UICollectionView
        self.photosCollectionView.reloadData()
    }

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return images.count
    }
}
```

```
func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "PhotoCell", for: indexPath) as! PhotoCollectionViewCell
    let asset = images[indexPath.row]
    let manager = PHImageManager.default()
    if cell.tag != 0 {
        manager.cancelImageRequest(PHImageRequestID(cell.tag))
    }
    cell.tag = Int(manager.requestImage(for: asset,
                                         targetSize: CGSize(width: 120.0, height: 120.0),
                                         contentMode: .aspectFill,
                                         options: nil) { (result, _) in
        cell.photoImageView?.image = result
    })
    return cell
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
    let width = self.view.frame.width * 0.32
    let height = self.view.frame.height * 0.179910045
    return CGSize(width: width, height: height)
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {
    return 2.5
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, insetForSectionAt section: Int) -> UIEdgeInsets {
    return UIEdgeInsets(top: 5, left: 5, bottom: 5, right: 5)
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {
    return 0
}
```

⇒ check storyboard make sure no conflict

11 ActivityViewController.swift (View Controllers)

import UIKit

```
class ActivityViewController : UIViewController, UIScrollViewDelegate {  
    @IBOutlet weak var segmentedControl : CustomSegmentedControl! {  
        didSet {  
            segmentedControl.delegate = self  
        }  
    }  
}
```

```
@IBOutlet weak var scrollView : UIScrollView! {  
    didSet {  
        scrollView.delegate = self  
    }  
}
```

```
var currentIndex : Int = 0
```

```
lazy var slides : [ActivityView] = {
```

```
    let followingActivityData = FollowingActivityModel()  
    let followingView = Bundle.main.loadNibNamed("Activity", owner: nil, options: nil())?.first as! ActivityView
```

```
    followingView.activityData = followingActivityData.followingActivity  
    let youActivityData = YouActivityModel()  
    let youView = Bundle.main.loadNibNamed("ActivityView", owner: nil, options: nil())?.first as! ActivityView
```

```
    return [followingView, youView]  
}
```

```
}
```

```
override func viewDidLoad() {
```

```
    super.viewDidLoad()  
    setupSlideScrollView(slides: slides)
```

```
}
```

```
override func viewDidAppear(_ animated: Bool) {
```

```
    super.viewDidAppear(animated)  
    self.navigationController?.setNavigationBarHidden(true, animated: false)
```

```
}
```

```
override func viewWillDisappear(_ animated: Bool) {
```

```
    super.viewWillDisappear(animated)  
    self.navigationController?.setNavigationBarHidden(false, animated: false)
```

```
}
```

```
func setupSlideScrollView(slides : [ActivityView]) {
```

```
    scrollView.frame = CGRect(x: 0, y: 0, width: view.frame.width, height: view.frame.height)
```

```
    scrollView.contentSize = CGSize(width: view.frame.width * CGFloat(slides.count), height: view.frame.height)
```

```
    scrollView.isPagingEnabled = true
```

```
    for i in 0...  
        < slides.count {
```

```
        slides[i].frame = CGRect(x: view.frame.width * CGFloat(i), y: 0, width: view.frame.width, height:  
            view.frame.height)
```

```
        scrollView.addSubview(slides[i])  
    }
```

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {  
    let pageIndex = round(scrollView.contentOffset.x / view.frame.width)  
    segmentedControl.updateSegmentedControlSegs(index: Int(pageIndex))  
    currentIndex = Int(pageIndex)}
```

}

3

```
extension ActivityViewController : ActivityDelegate {
```

```
func scrollTo(index: Int) {  
    if currentIndex == index {  
        return  
    }  
  
    let pageWidth: CGFloat = self.scrollView.frame.width  
    let slideToX = pageWidth * CGFloat(index)  
    self.scrollView.scrollRectToVisible(CGRect(x: slideToX, y: 0, width: pageWidth, height: self.scrollView.frame.height), animated: true)}
```

?

3