cocoapods

Firebase/Core ↙ under use Frameworks
Firebase/Auth
Firebase/Firestore
Firebase/Storage
Gallery
InstantSearch Client
Empty DataSet-Swift
NVActivityIndicatorView/AppExtension
JGProgressHUD
PayPal-iOS-SDK

drag and drop the folder path to terminal

// Tab Bar Controller
change the title of buttons
drag the buttons to change order

// collection view controller → section inserts ⇒ 10
Relationship segue ⇒ View Controller
embeded in navigation controller
put image view in the cell
+ label → # set the lines to 0 & auto detect

// Free icons : icons8.com

// AppDelegate.swift
```
import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        FirebaseApp.configure()

        return true
    }
```

// Firebase CollectionReference.swift (helpers)
```
import Foundation
import FirebaseFirestore

enum FCollectionReference: String {
    case User
    case Category
    case Item
    case Basket
}

func FirebaseReference(_ collectionReference: FCollectionReference) -> CollectionReference {
    return Firestore.firestore().collection(collectionReference.rawValue)
}
```

// Constants.swift (Helpers)
```
import Foundation
// IDs and Keys
public let kFILEREFERENCE = "gs://.....appspot.com"
public let kALGOLIA_APP_ID = "...."
public let kALGOLIA_KEY = "....."
// Firebase Headers
public let kUSER_PATH = "User"
public let kCATEGORY_PATH = "Category"
public let kITEMS_PATH = "Items"
public let kBASKET_PATH = "Basket"
// Category
```

* For testing do not forget to change the rules
↗ link from firebase storage

// use capitalized letter to differenciate main folder and sub

```
public let kNAME = "name"
public let kIMAGENAME = "imageName"
public let kOBJECTID = "objectId"
//Item
public let kCATEGORYID = "categoryId"
public let kDESCRIPTION = "description"
public let kPRICE = "price"
public let kIMAGELINKS = "imageLinks"
//Basket
public let kOWNERID = "ownerId"
public let kITEMIDS = "itemIds"
//User
public let kEMAIL = "email"
public let kFIRSTNAME = "firstName"
public let kLASTNAME = "lastName"
public let kFULLNAME = "fullName"
public let kCURRENTUSER = "currentUser"
public let kFULLADDRESS = "fullAddress"
public let kONBOARD = "onBoard"
public let kPURCHASEDITEMIDS = "purchasedItemIds"
```

// Reachability.swift (Helpers) ※ ⇒ check if the device is connected to the Internet

import SystemConfiguration // will return true of false ※ copy from lecture 25

```swift
public class Reachabilty {

    class func HasConnection() -> Bool {

        var zeroAddress = sockaddr_in(sin_len: 0, sin_family: 0, sin_port: 0,
sin_addr: in_addr(s_addr: 0), sin_zero: (0, 0, 0, 0, 0, 0, 0, 0))

        zeroAddress.sin_len = UInt8(MemoryLayout.size(ofValue: zeroAddress))
        zeroAddress.sin_family = sa_family_t(AF_INET)

        let defaultRoutReachability = withUnsafePointer(to: &zeroAddress, {
            $0.withMemoryRebound(to: sockaddr.self, capacity: 1, {
                zeroSockAddress in

                SCNetworkReachabilityCreateWithAddress(nil, zeroSockAddress)
            })
        })

        var flags : SCNetworkReachabilityFlags = SCNetworkReachabilityFlags(rawValue:
0)

        if SCNetworkReachabilityGetFlags(defaultRoutReachability!, &flags) == false {
            return false
        }

        let isReachable = (flags.rawValue & UInt32(kSCNetworkFlagsReachable)) != 0

        let needConnection = (flags.rawValue &
UInt32(kSCNetworkFlagsConnectionRequired)) != 0

        return (isReachable && !needConnection)
    }
}
```

// DownLoader.swift (Helpers)

```
import Foundation
import FirebaseStorage

let storage = Storage.storage()

func uploadImages(images: [UIImage?], itemId: String, completion: @escaping(_ imageLinks: [String]) -> void){
    if Reachability.HasConnection(){
        var uploadedImagesCount = 0
        var imageLinkArray: [String] = []
        var nameSuffix = 0
        for image in Images{
            let fileName = "ItemImages/" + itemId + "/" + "\(nameSuffix)" + ".jpg"
            let imageData = image!.jpegData(compressionQuality: 0.5)
            saveImageInFirebase(imageData: imageData!, fileName: fileName){ (imageLink) in
                if imageLink != nil{
                    imageLinkArray.append(imageLink!)
                    uploadedImagesCount += 1
                    if uploadedImagesCount == images.count{
                        completion(imageLinkArray)
                    }
                }
                nameSuffix += 1
            }
        } else{
            print("No Internet connection")
        }
    }
func saveImageInFirebase(imageData: Data, fileName: String, completion: @escaping(_ imageLink: String?)->void){
    var task: StorageUploadTask!
    let storageRef = storage.reference(forURL: kFILEREFERENCE).child(fileName)

    task = storageRef.putData(imageData, metaData: nil, completion: { (metaData, error) in
            task.removeAllObservers()
            if error != nil{
                print("Error uploading image", error!.localizedDescription)
                completion(nil)
                return
            }
            storageRef.downloadURL{(url, error) in
                guard let downloadUrl = url else{
                    completion(nil)
                    return
                }
                completion(downloadUrl.absoluteString)
            }
        }
    }
}
```

```
func downloadImages(imageUrls:[String], completion: @escaping (_ images:[UIImage?]) -> void){
    var imageArray:[UIImage] = []
    var downloadCounter = 0

    for link in imageUrls{
        let url = NSURL(string: link)
        let downloadQueue = DispatchQueue(label: "imageDownloadQueue")
        downloadQueue.async{
            downloadCounter += 1
            let data = NSData(contentsOf: url! as URL)
            if data != nil{
                imageArray.append(UIImage(data: data! as Data)!)
                if downloadCounter == imageUrls.count
                    DispatchQueue.main.async{ -> ? imageArray, counter in lecture
                        completion(imageArray)
                    }
                }
            } else{
                print("could not download images")
                completion(imageArray)
            }
        }
    }
}
```

```
//HelperFunctions.swift (Helpers)
import Foundation

func convertToCurrency(_ number: Double) -> String{
    let currencyFormatter = NumberFormatter()
    currencyFormatter.usesGroupingSeparator = true
    currencyFormatter.numberStyle = .currency
    currencyFormatter.locale = Locale.current

    return currencyFormatter.string(from: NSNumber(value: number))!
}
```

```
//CategoryCollectionViewCell.swift (Custom Cells)
import UIKit
class CategoryCollectionViewCell: UICollectionViewCell{
    @IBOutlet weak var nameLabel: UILabel!
    @IBOutlet weak var imageView: UIImageView!

    func generateCell(_ category: Category){
        nameLabel.text = category.name
        imageView.image = category.image
    }
}
```

//option + click in main story board

//ctrl + drag

// ItemTableViewCell.swift (Custom Cells)

* make sure the class connected to the cell

```
import UIKit

class ItemTableViewCell: UITableViewCell{
    @IBOutlet weak var itemImageView : UIImageView!
    // similar ones

    override func awakeFromNib(){
        super.awakeFromNib()
    }

    override func setSelected(_ selected: Bool, animated: Bool){
        super.setSelected(selected, animated: animated)
    }

    func generateCell(_ item: Item){
        nameLabel.text = item.name
        descriptionLabel.text = item.description
        priceLabel.text = convertToCurrency(item.price)

        if item.imageLinks != nil && item.imageLinks.count > 0{
            downloadImages(imageUrls: [item.imageLinks.first!]){(images) in
                self.itemImageView.image = images.first as? UIImage
            }
        }
    }
```

// ImageCollectionViewCell.swift (Custom Cells)

```
import UIKit

class: ImageCollectionViewCell: UICollectionViewCell{
    @IBOutlet weak var imageView: UIImageView!
    func setupImageWith(itemImage: UIImage){
        imageView.image = itemImage
    }


}
```

```
//Category.swift (Model)
import Foundation
import UIKit

class Category{
    var id: String
    var name: String
    var image: UIImage?
    var imageName: String?

    init(_name: String, _imageName: String){
        id = ""
        name = _name
        imageName = _imageName
        image = UIImage(named: _imageName)
    }
    init(_dictionary: NSDictionary){
        id = _dictionary[kOBJECTID] as! String
        name = _dictionary[kNAME] as! String
        image = UIImage(named: _dictionary[kIMAGENAME] as? String ?? "")   // default value
    }
}
//MARK: Download category from firebase
func downloadCategoriesFromFirebase(completion: @escaping (_ categoryArray: [Category])->void){
    var categoryArray: [Category] = []
    FirebaseReference(.Category).getDocuments{(snapshot, error) in
        guard let snapshot = snapshot else{
            completion(categoryArray)
            return
        }
        if !snapshot.isEmpty{
            for categoryDict in snapshot.documents{
                categoryArray.append(Category(_dictionary: categoryDict.data() as NSDictionary))
            }
        }
        completion(categoryArray)
    }
}

//MARK: Save category function, only use once
func saveCategoryToFirebase(_ category: Category){
    let id = UUID().uuidString
    category.id = id
                    // user defined => check FirebaseCollectionReference.swift
    FirebaseReference(.Category).document(id).setData(categoryDictionaryFrom(category) as!
                                                      [String: Any])
}
```

```
//MARK: Helpers, convert to dictionary
func categoryDictionaryFrom(_ category: Category) -> NSDictionary {

    return NSDictionary(objects: [category.id, category.name, category.imageName],
        forKeys: [kOBJECTID as NSCopying, kNAME as NSCopying, kIMAGENAME as NSCopying])

}

//use only one time
func createCategorySet(){
    let womanClothing = Category(_name: "Woman's Clothing", _imageName: "womanCloth")
    //similar ones

    let arrayOfCategories = [womanClothing, ----]

    for category in arrayOfCategories{
        saveCategoryToFirebase(category)
    }
}
```

```
//Basket.swift (model)
import Foundation

class Basket{
    var id: String!
    var ownerId: String!
    var itemIds: [String]!

    init(){
    }
    init(_dictionary: NSDictionary){
        id = _dictionary[kOBJECTID] as? String
        ownerId = _dictionary[kOWNERID] as? String
        itemIds = _dictionary[kITEMIDS] as? [String]
    }
}
```

```
//MARK: Download Items
func downloadBasketFromFirestore(_ ownerId: String, completion: @escaping (_basket: Basket?) -> Void){
    FirebaseReference(.Basket).whereField(kOWNERID, isEqualTo: ownerId).getDocuments{(snapshot, error) in
        guard let snapshot = snapshot else{
            completion(nil)
            return
        }
    }
    if !snapshot.isEmpty && snapshot.documents.count > 0{
        let basket = Basket(_dictionary: snapshot.documents.first!.data() as NSDictionary)
        completion(basket)
    }else{
        completion(nil)
```

```
    }
  }
}

//MARK: - Save to Firebase
func saveBasketToFirestore(_ basket: Basket){
    FirebaseReference(.Basket).document(basket.id).setData(basketDictionaryFrom(basket) as! [String:Any])

}

//MARK: Helper functions
func basketDictionaryFrom(_ basket: Basket) -> NSDictionary{
    return NSDictionary(objects:[basket.id, basket.ownerId, basket.itemIds],
                        forKeys:[kOBJECTID as NSCopying, kOWNERID as NSCopying, kITEMIDS as NSCopying])
}

//MARK: update basket
func updateBasketInFireStore(_ basket: Basket, withValues:[String:Any], completion: @escaping (_
    error: Error?) -> Void){
        FirebaseReference(.Basket).document(basket.id).updateData(withValue){ (error) in
            completion(error)
        }
}

//Items.swift      (Model)
import Foundation
import UIKit
import InstantSearchClient
class Item{
    var id: String!
    var categoryId: String!
    var name: String!
    var description: String!
    var price: Double!
    var imageLinks: [String]!

    init(){
    }
    init(_ dictionary: NSDictionary){
        id = dictionary[kOBJECTID] as? String
        categoryId = dictionary[kCATEGORYID] as? String
        name = dictionary[kNAME] as? String
        description = dictionary[kDESCRIPTION] as? String
        price = dictionary[kPRICE] as? Double
        imageLinks = dictionary[kIMAGELINKS] as? [String]
    }
}
//MARK: Save Items func
func saveItemToFireStore(_ item: Item){
    FirebaseReference(.Items).document(item.id).setData(itemDictionaryFrom(item) as! [String:Any])
}
```

```
//MARK: Helper functions
func itemDictionaryFrom(_ item: Item) -> NSDictionary {
    return NSDictionary(objects: [item.id, item.categoryId, item.name, item.description, item.price, item.imageLinks],
                        forKeys: [kOBJECTID as NSCopying, kCATEGORYID as NSCopying, kNAME as NSCopying,
                                  kDESCRIPTION as NSCopying, kPRICE as NSCopying, kIMAGELINKS as NSCopying])
}

//MARK: Download Func
func downloadItemsFromFirebase(_ withCategoryId: String, completion: @escaping (_ itemArray: [Item]) -> Void){
    var itemArray: [Item] = []
    FirebaseReference(.Items).whereField(kCATEGORYID, isEqualTo: withCategoryId).getDocuments{ (snapshot, error) in
        guard let snapshot = snapshot else{
            completion(itemArray)
            return
        }
        if !snapshot.isEmpty{
            for itemDict in snapshot.documents{
                itemArray.append(Item(_dictionary: itemDict.data() as NSDictionary))
            }
        }
        completion(itemArray)
    }
}

func downloadItems(_ withIds: [String], completion: @escaping (_ itemArray: [Item]) -> Void){
    var count = 0
    var itemArray: [Item] = []
    if withIds.count > 0{
        for itemId in withIds{
            FirebaseReference(.Items).document(itemId).getDocument{ (snapshot, error) in
                guard let snapshot = snapshot else{
                    completion(itemArray)
                    return
                }
                if snapshot.exists{
                    itemArray.append(Item(_dictionary: snapshot.data()! as NSDictionary))
                    count += 1
                }else{
                    completion(itemArray)
                }
                if count == withIds.count{
                    completion(itemArray)
                }
            }
        }
    }else{
        completion(itemArray)
    }
}
```

```
// MARK: Algolia Funcs

func saveItemToAlgolia(item: Item){
    let index = AlgoliaService.shared.index
    let itemToSave = itemDictionaryFrom(item) as! [String: Any]
    index.addObject(itemToSave, withID: item.id, requestOptions: nil(){(content, error) in
        if error != nil{
            print("error saving to algolia", error!.localizedDescription)
        }else{
            print("added to algolia")
        }
    }
}

func searchAlgolia(searchString: String, completion: @escaping(_ itemArray: [String]) -> Void){
    let index = AlgoliaService.shared.index
    var resultIds: [String] = []
    let query = Query(query: searchString)
    query.attributesToRetrive = ["name", "description"]
    index.search(query){(content, error) in
        if error == nil{
            let cont = content!["hits"] as! [[String: Any]]
            resultIds = []
            for result in cont{
                resultIds.append(result["objectId"] as! String)
            }
            completion(resultIds)
        }else{
            print("error algolia search", error!.localizedDescription)
            completion(resultIds)
        }
    }
}
```

```
//MUser.swift (Model)
import Foundation
import FirebaseAuth

class MUser{
    let objectId: String
    var email: String
    var firstName: String
    var lastName: String
    var fullName: String
    var purchasedItemIds: [String]
    var fullAddress: String?
    var onBoard: Bool

    //MARK: Initializers
    init(_objectId: String, _email: String, _firstName: String, _lastName: String){
        objectId = _objectId
        email = _email
        firstName = _firstName
        lastName = _lastName
        fullName = _firstName + " " + _lastName
        fullAddress = ""
        onBoard = false
        purchasedItemIds = []
    }

    init(_dictionary: NSDictionary){
        objectId = _dictionary[kOBJECTID] as! String
        if let mail = _dictionary[kEMAIL]{
            email = mail as! String
        }else{
            email = ""
        }
        if let fname = _dictionary[kFIRSTNAME]{
            firstName = fname as! String
        }else{
            firstName = ""
        }
        if let lname = _dictionary[kLASTNAME]{
            lastName = lname as! String
        }else{
            lastName = ""
        }
        fullname = firstName + " " + lastName
        if let faddress = _dictionary[kFULLADDRESS]{
            fulladdress = faddress as! String
        }else{
            fulladdress = ""
        }
        ↓ similar ones for onBoard Bool => default false
```

```
        v purchaseIds => [String] => default []
    }

    //MARK: Return current user
    class func currentId() -> String{
        return Auth.auth().currentUser.uid
    }
    class func currentUser() -> MUser?{
        if Auth.auth().currentUser != nil{
            if let dictionary = UserDefaults.standard.object(forKey: kCURRENTUSER){
                return MUser.init(_dictionary: dictionary as! NSDictionary)
            }
        }
        return nil
    }

    //MARK: login func
    class func loginUserWith(email:String, password:String, completion: @escaping (_ error: Error?,
        _ isEmailVerified:Bool) -> Void){
        Auth.auth().signIn(withEmail: email, password: password){ (authDataResult, error) in
            if error == nil{
                if authDataResult!.user.isEmailVerified{
                    downloadUserFromFirestore(userId: authDataResult?.user.uid, email: email)
                    completion(error, true)
                } else{
                    print("email not verified")
                    completion(error, false)
                }
            }else{
                completion(error, false)
            }
        }
    }

    //MARK: Register user
    class func registerUserWith(email: String, password: String, completion: @escaping (_ error: Error?) -> Void){
        Auth.auth().createUser(withEmail: email, password: password){ (authDataResult, error) in
            completion(error)
            if error == nil{
                authDataResult!.user.sendEmailVerification{ (error) in
                    print("error:", error?.localizedDescription)
                }
            }
        }
    }
}

MARK: Resend link methods
class func resetPasswordFor(email: String, completion: @escaping (_ error: Error?) -> Void){
    Auth.auth().sendPasswordReset(withEmail: email){ (error) in
        completion(error)
    }
```

```
}

class func resendVerificationEmail(email: String, completion: @escaping(_ error: Error?) -> Void){
    Auth.auth().currentUser?.reload(completion: {(error) in
        Auth.auth().currentUser?.sendEmailVerification(completion: {(error) in
            print("resend email error: ", error?.localizedDescription)
            completion(error)
        })
    })
}

class func logOutCurrentUser(completion: @escaping(_ error: Error?) -> Void){
    do{
        try Auth.auth().signOut()
        UserDefaults.standard.removeObject(forKey: kCURRENTUSER)
        UserDefaults.standard.synchronize()
        completion(nil)
    }catch let error as NSError{
        completion(nil)
    }
}
} // end of class

//MARK: Download User
func downloadUserFromFirestore(userId: String, email: String){
    FirebaseReference(.User).document(userId).getDocument{ (snapshot, error) in
        guard let snapshot = snapshot else { return }
        if snapshot.exists {
            print("download current user from firebase")
            saveUserLocally(mUserDictionary: snapshot.data()! as NSDictionary)
        } else{
            let user = MUser(_objectId: userId, email: email, _firstName: "", _lastName: "")
            saveUserLocally(mUserDictionary: userDictionaryFrom(user: user))
            saveUserToFirestore(mUser: user)
        }
    }
}

//MARK: Save user to firebase
func saveUserToFirestore(mUser: MUser){
    FirebaseReference(.User).document(mUser.objectId).setData(userDictionaryFrom(user: mUser) as!
        [String: Any]){(error) in
        if error != nil{
            print("error saving user \(error!.localizedDescription)")
        }
    }
}
func saveUserLocally(mUserDictionary: NSDictionary){
    UserDefaults.standard.set(mUserDictionary, forKey: kCURRENTUSER)
    UserDefaults.standard.synchronize()
}
```

```
//MARK: Helper Function
func userDictionaryFrom(user: MUser) -> NSDictionary{
    return NSDictionary(objects:[user.objectId, user.email, user.firstName, user.firstName,
        user.lastName, user.fullName, user.fullAddress ?? "", user.onBoard, user.purchasedItemIds],
        forKeys:[kOBJECTID as NSCopying, kEMAIL as NSCopying, kFIRSTNAME as NSCopying, kLASTNAME as NSCopying,
        kFULLNAME as NSCopying .....])
}

//MARK: update user
func updateCurrentUserInFirestore(withValues:[String:Any], completion: @escaping (_ error: Error?) -> Void){
    if let dictionary = UserDefaults.standard.object(forKey: kCURRENTUSER){
        let userObject = (dictionary as! NSDictionary).mutableCopy() as! NSMutableDictionary
        userObject.setValuesForKeys(withValues)
        FirebaseReference(.User).document(MUser.currentId()).updateData(withValues){ (error) in
            completion(error)
            if error == nil{
                saveUserLocally(mUserDictionary: userObject)
            }
        }
    }
}

//AlgoliaService.swift (Model)

import Foundation
import InstantSearchClient

class AlgoliaService{
    static let shared = AlgoliaService()
    let client = Client(appID: kALGFOLIA_APP_ID, apiKey: kALGOLIA_ADMIN_KEY)
    let index = Client(appID: kALGFOLIA_APP_ID, apiKey: kALGOLIA_ADMIN_KEY).index(withName: "itemName")

    private init(){}
}
```

```
//new - cocoa touch class UICollectionViewController
//CategoryCollectionViewController (main Views)
import UIKit

class CategoryCollectionViewController: UICollectionViewController {
    //MARK: Vars
    var categoryArray: [Category] = []
    private let sectionInsets = UIEdgeInsets(top: 20.0, left: 10.0, bottom: 20.0, right: 10.0)
    private let itemsPerRow: CGFloat = 3

    //MARK: View Life Cycle
    override func viewDidLoad(){
        super.viewDidLoad()                    //createCategorySet() & run once
    }

    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        loadCategories()
    }

    //MARK: UICollectionViewDataSource
    override func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int)->
        Int{
        return categoryArray.count
    }

    override func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath)
        -> UICollectionViewCell{
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "Cell", for: indexPath)
            as! CategoryCollectionViewCell
        cell.generateCell(categoryArray[indexPath.row])

        return cell
    }

    //MARK: UICollectionView Delegate
    override func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath){

        performSegue(withIdentifier: "categoryToItemsSeg", sender: categoryArray[indexPath.row])
    }

    //MARK: Navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?){
        if segue.identifier == "categoryToItemsSeg"{
            let vc = segue.destination as! ItemsTableViewController
            vc.category = sender as! Category

        }
    }
```

select the cell ⇒ # ⇒ identifier "Cell"
⇒ class: CategoryCollectionViewCell

```
//MARK: Download categories
private func loadCategories(){                    → From Model
    downloadCategoriesFromFirebase{ (allCategories) in
        self.categoryArray = allCategories
        self.collectionView.reloadData()
    }
}

}
```

storyboard → collection view
* bug fix ⇒ size inspector ⇒ Estimate size ⇓ None

```
extension CategoryCollectionViewController: UICollectionViewDelegateFlowLayout {

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
        sizeForItemAt indexPath: indexPath) -> CGSize{
        let paddingSpace = sectionInsets.left * (itemsPerRow+1)
        let availableWidth = view.frame.width - paddingSpace
        let widthPerItem = availableWidth / itemsPerRow

        return CGSize(width: widthPerItem, height: widthPerItem)
    }

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
        insetForSectionAt section: Int) -> UIEdgeInsets{

        return sectionInsets
    }

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

        return sectionInsets.left
    }
```

```
//ItemsTableViewController (main Views)
import UIKit
import EmptyDataSet_Swift
class ItemsTableViewController: UITableViewController{
    //MARK: Vars
    var category: Category?
    var itemArray: [Item] = []

    //MARK: View LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.tableFooterView = UIView()          → // remove the empty cells
        self.title = category?.name
        tableView.emptyDataSetSource = self
        tableView.emptyDataSetDelegate = self
```

* it bug ⇒ chang segue the change back to add the nav bar button

// select cell ⇒ # ⇒ Accesory ⇒ Disclosure Indicator
ctrl + drag ⇒ equal width

```
}

override func viewDidAppear(_ animated: Bool){
    super.viewDidAppear(animated)
    if category != nil{
        loadItems()
    }
}
MARK: table view data source
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
    return itemArray.count
}

override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! ItemTableViewCell
    cell.generateCell(itemArray[indexPath.row])
    return cell
}

//MARK: TableView Delegate
override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){
    tableView.deselectRow(at: indexPath, animated: true)
    showItemView(itemArray[indexPath.row])
}

//MARK: Navigation
override func prepare(for segue: UIStoryboardSegue, sender: Any?){
    if segue.identifier == "itemToAddItemSeg"{
        let vc = segue.destination as!
        vc.category = category!
    }
}
private func showItemView(_ item: Item){                    * give the view controller a storyboard id
    let itemVC = UIStoryboard.init(name: "Main", bundle: nil).instantiateViewController(identifier: "itemView") as!
        ItemViewController
    itemVC.item = item
    self.navigationController?.pushViewController(itemVC, animated: true)
}

//MARK: Load Items
private func loadItems(){
    downloadItemsFromFirebase(category!.id,){ (allItems) in
        self.itemArray = allItems
        self.tableView.reloadData()
    }
}
```

```
extension ItemsTableViewController: EmptyDataSetSource, EmptyDataSetDelegate{
    func title(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "No items to display!")
    }

    func image(forEmptyDataSet scrollView: UIScrollView) -> UIImage?{
        return UIImage(named: "emptyData")
    }

    func description(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "Please check back later")
    }
}
```

// AddItemViewController.swift (Main Views)

⌥ option + click ⇒ opens assistant editor

¥8-2

```
import UIKit
import Gallery
import JGProgressHUD
import NVActivityIndicatorView

class AddItemViewController: UIViewController{
    //MARK: IBOutlets
    @IBOutlet weak var titleTextField: UITextField!
    @IBOutlet weak var priceTextField: UITextField!
    @IBOutlet weak var descriptionTextView: UITextView!

    //MARK: Vars
    var category: Category!
    var gallery: GalleryController!
    let hud = JGProgressHUD(style: .dark)
    var activityIndicator: NVActivityIndicatorView?

    var itemImages: [UIImage?] = []

    //MARK: View LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
    }

    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        activityIndicator = NVActivityIndicatorView(frame: CGRect(x: self.view.frame.width/2 - 30,
            y: self.view.frame.height/2 - 30, width: 60, height: 60), type: .ballPulse, color: #colorLiteral(red: 0.4998469949,
            green: 0.4941213727, blue: 0.4784867811, alpha: 1), padding: nil)
    }

    //MARK: IBActions
    @IBAction func doneBarButtonItemPressed(_ sender: Any){
        dissmissKeyboard()
        if fieldsAreCompleted(){
            saveToFirebase()
        } else {
            self.hud.textLabel.text = "All Fields are required!"
            self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
            self.hud.view(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)

        }
    }

    @IBAction func cameraButtonPressed(_ sender: Any){
        itemImages = []
        showImageGallery()
    }
```

```
                                                    → Tap recognizer
@IBAction func backgroundTapped(_ sender: Any){
    dismissKeyboard()
}

//MARK: Helper functions
private func fieldsAreCompleted() -> Bool{
    return (titleTextField.text != "" && priceTextField.text != "" && descriptionTextView.text != "")
}

private func dismissKeyboard(){
    self.view.endEditing(false)
}

private func popTheView(){
    self.navigationController?.popViewController(animated: true)
}
//MARK: Save Item
private func saveToFirebase(){
    showLoadingIndicator()
    let item = item()
    item.id = UUID().uuidString
    item.name = titleTextField.text!
    item.categoryId = category.id
    item.description = descriptionTextView.text
    item.price = Double(priceTextField.text!)
    if itemImages.count > 0{
        uploadImages(images: itemImages, itemId: item.id){ (imageLinkArray) in
            item.imageLinks = imageLinkArray
            saveItemToFireStore(item)
            saveItemToAlgolia(item: item)
            self.hideLoadingIndicator()
            self.popTheView()
        }
    }else{
        saveItemToFirestore(item)
        saveItemToAlgolia(item: item)
        popTheView()
    }
}
//MARK: Activity Indicator
private func showLoadingIndicator(){
    if activityIndicator != nil{
        self.view.addSubview(activityIndicator!)
        activityIndicator!.startAnimating()
    }
}
private func hideLoadingIndicator(){
    if activityIndicator != nil{
        activityIndicator!.removeFromSuperview()
        activityIndicator!.stopAnimating()
```

```
        }
    }

    //MARK: Show Gallery
    private func showImageGallery(){
        self.gallery = GalleryController()
        self.gallery.delegate = self

        Config.tabsToShow = [.imageTab, .cameraTab]
        Config.Camera.imageLimit = 6

        self.present(self.gallery, animated: true, completion: nil)
    }
}

extension AddItemViewController: GalleryControllerDelegate{
    func galleryController(_ controller: GalleryController, didSelectImages images: [Image]){
        if images.count > 0{
            Image.resolve(images: images){ resolvedImages in
                self.itemImages = resolvedImages

            }
        }
        controller.dismiss(animated: true, completion: nil)
    }

    func galleryController(_ controller: GalleryController, didSelectVideo video: Video){
        controller.dismiss(animated: true, completion: nil)
    }

    func galleryController(_ controller: GalleryController, requestLightbox images: [Image]){
        controller.dismiss(animated: true, completion: nil)
    }

    func galleryControllerDidCancel(_ controller: GalleryController){
        controller.dismiss(animated: true, completion: nil)
    }

}
```

⌘ p-list +⇒ Privacy Camera Usage Description
Privacy Photo Library Usage Description

// ItemViewController.swift (Main Views)

```
import UIKit
import JGProgressHUD
class ItemViewController: UIViewController{
    //MARK: IBOutlets
    @IBOutlet weak var imageCollectionView: UICollectionView!
    ↓ similar ones

    //MARK: Vars
    var item : Item!
    var itemImages : [UIImages] = []
    let hud = JGProgressHUD(style.dark)

    private let sectionInsets = UIEdgeInsets(top: 0.0, left: 0.0, bottom: 0.0, right: 0.0)
    private let itemsPerRow: CGFloat = 1
    private let cellHeight: CGFloat = 196.0

    //MARK: ViewLifecycle
    override func viewDidLoad(){
        super.viewDidLoad()
        setupUI()
        downloadPictures()
        self.navigationItem.leftBarButtonItems = [UIBarButtonItem(image: UIImage(named: "back"), style:.plain,
                target: self, action: #selector(self.backAction))]
        self.navigationItem.rightBarButtonItems = [UIBarButtonItem(image: UIImage(named: "addToBasket"), style:.plain,
                target: self, action: #selector(self.addToBasketButtonPressed))]

    }

    //MARK: Download Pictures
    private func downloadPictures(){
        if item != nil && item.imageLinks != nil{
            downloadImages(imageUrls: item.imageLinks){(allImages) in
                if allImages.count > 0{
                    self.itemImages = allImages as! [UIImage]
                    self.imageCollectionView.reloadData()
                }
            }
        }
    }

    //MARK: Setup UI
    private func setupUI(){
        if item != nil{
            self.title = item.name
            nameLabel.text = item.name
            priceLabel.text = convertToCurrency(item.price)
            descriptionTextView.text = item.description
        }
    }
```

// command+N ⇒ new file
// change the cell size to the collection view cell
// give id to the cell as "Cell"
* main storyboard ⇒ select Image Collection View
↓ ⇒ dataSource → ItemViewController → equivileant to use code
delegate → ItemViewController
(control and drag)
⇒ → scrolling from vertical to horizontal

* Collection View ⇒ paging enabled

```
MARK: - IBActions
@objc func backAction(){
    self.navigationController?.popViewController(animated: true)
}

@objc func addToBasketButtonPressed(){
    if MUser.currentUser() != nil{
        downloadBasketFromFirestore(MUser.currentId()){ (basket) in
            if basket == nil{
                self.createNewBasket()
            } else {
                basket!.itemIds.append(self.item.id)
                self.updateBasket(basket: basket!, withValues: [kITEMIDS: basket!.itemIds])
            }
        }
    } else{
        showLoginView()
    }
}

//MARK: Add to basket
private func createNewBasket(){
    let newBasket = Basket()
    newBasket.id = UUID().uuidString
    newBasket.ownerId = MUser.currentId()
    newBasket.itemIds = [self.item.id]
    saveBasketToFirestore(newBasket)

    self.hud.textLabel.text = "Added to basket!"
    self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
    self.hud.show(in: self.view)
    self.hud.dismiss(afterDelay: 2.0)
}

Private func updateBasket(basket: Basket, withValues: [String: Any]){
    updateBasketInFirestore(basket, withValues: withValues){ (error) in
        if error != nil{
            self.hud.textLabel.text = "Error: (error!.localizedDescription)"
            self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
            self.hud.show(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)
            print("error updating basket", error!.localizedDescription)
        }else{
            self.hud.textLabel.text = "Added to basket!"
            self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
            self.hud.show(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)
        }
    }
}
```

```
    //MARK: show login view
    private func showLoginView(){
        let loginView = UIStoryboard.init(name: "Main", bundle: nil).instantiateViewController(identifier: "loginView") as!


        self.present(loginView, animated: true, completion: nil)
    }
}

extension ItemViewController: UICollectionViewDataSource, UICollectionViewDelegate{

    //MARK: UICollectionViewDataSource
    override func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int)->
        Int{
        return itemImages.count == 0 ? 1 : itemImages.count
    }

    override func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath)
        -> UICollectionViewCell{
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "Cell", for: indexPath) as!
                ImageCollectlayViewCell
        if itemImages.count > 0{
            cell.setupImageWith(itemImage: itemImages[indexPath.row])
        }

        return cell
    }
}

                                                    storyboard -> collection view  <- also Image viewer
// bug fix => size inspector => Estimate size
                                                                  None
extension ItemViewController: UICollectionViewDelegateFlowLayout {

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
        sizeForItemAt indexPath: indexPath) -> CGSize{
        let availableWidth = collectionView.frame.width - sectionInsets.left
        return CGSize(width: availableWidth, height: cellHeight)
    }

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
        insetForSectionAt section: Int) -> UIEdgeInsets{

        return sectionInsets
    }

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

        return sectionInsets.left
    }
```

// Basket View Controller.swift (Main Views)

```
import UIKit

class BasketViewController: UIViewController{

    //MARK: IBOutlets
    @IBOutlet weak var footerView: UIView!
```

↓ footerView (UIView), totalItemsLabel, basketTotalPriceLabel (UILabel) checkOutButtonOutlet (UIButton)

```
    //MARK: Vars
    var basket: Basket?
    var allItems: [Item] = []
    var purchasedItemIds: [String] = []
    let hud = JGProgressHUD(style: .dark)

    //MARK: View Lifecycle
    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.tableViewFooterView = footerView
    }

    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)

        if MUser.currentUser() != nil{
            loadBasketFromFirestore()
        }else{
            self.updateTotalLabels(true)
        }
    }

    //MARK: IBActions
    @IBAction func checkoutButtonPressed(_ sender: Any){

        if MUser.currentUser()!.onBoard{
            //temptunction()
            payButtonPressed()

            // addItemsToPurchaseHistory(self.purchasedItemIds)
            // emptyTheBasket()
        }else{
            self.hud.textLabel.text = "Please complete your profile!"
            self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
            self.hud.show(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)
        }
    }
```

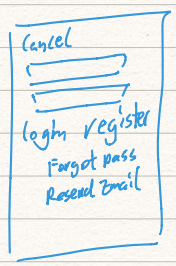
```
//MARK: Download basket
private func loadBasketFromFirestore(){
    downloadBasketFromFirestore(MUser.currentId()) { (basket) in
        self.basket = basket
        self.getBasketItems()
    }
}

private func getBasketItems(){
    if basket != nil{
        downloadItems(basket!.itemIds) { (allItems) in
            self.allItems = allItems
            self.updateTotalLabels(false)
            self.tableView.reloadData()
        }
    }
}

//MARK: Helper functions

                    -> only for testing
// func tempFunction(){
    for item in allItems{
        purchasedItemIds.append(item.id)
    }
}

private func updateTotalLabels(_ isEmpty: Bool){
    if isEmpty{
        totalItemsLabel.text = "0"
        basketTotalPriceLabel.text = returnBasketTotalPrice()
    }else{
        totalItemsLabel.text = "\(allItems.count)"
        basketTotalPriceLabel.text = returnBasketTotalPrice()
    }
    checkoutButtonStatusUpdate()
}

private func returnBasketTotalPrice() -> String{
    var totalPrice = 0.0
    for item in allItems{
        totalPrice += item.price
    }
    return "Total price: " + convertToCurrency(totalPrice)
}

private func emptyTheBasket(){
    purchasedItemIds.removeAll()
    allItems.removeAll()
    tableView.reloadData()
    basket!.itemIds = []
```

```
            updateBasketInFirestore(basket!, withValues: [kITEMIDS: basket!.itemIds]) { (error) in
                if error != nil{
                    print("error updating basket", error!.localizedDescription)
                }
                self.getBasketItems()

            }
        }

        private func addItemsToPurchaseHistory(_ itemIds: [String]){
            if MUser.currentUser() != nil{
                let newItemIds = MUser.currentUser()!.purchasedItemIds + itemIds
                updateCurrentUserInFirestore(withValues: [kPURCHASEDITEMIDS: newItemIds]) { (error) in
                    if error != nil{
                        print("error adding purchased items", error!.localizedDescription)
                    }
                }

            }
        }

        //MARK: Navigation
        private func showItemView(withItem: Item){                    // give the view controller a storyboard id
            let itemVC = UIStoryboard.init(name: "Main", bundle: nil).instantiateViewController(identifier: "itemView") as!
                ItemViewController
            itemVC.item = withItem
            self.navigationController?.pushViewController(itemVC, animated: true)
        }

        //MARK: Control checkOutButton
        private func checkOutButtonStatusUpdate(){
            checkOutButtonOutlet.isEnabled = allItems.count > 0
            if checkOutButtonOutlet.isEnabled{
                checkOutButtonOutlet.backgroundColor = [] //type in color literal
            }else{
                disableCheckoutButton()
            }
        }

        private func disableCheckoutButton(){
            checkOutButtonOutlet.isEnabled = false
            checkOutButtonOutlet.backgroundColor = []
        }

        private func removeItemFromBasket(itemId: String){
            for i in 0..<basket!.itemIds.count{
                if itemId == basket!.itemIds[i]{
                    basket!.itemIds.remove(at: i)
                    return
                }
            }
        }
    }
```

```
extension BasketViewController: UITableViewDataSource, UITableViewDelegate{

    MARK: table view data source
    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        return allItems.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! ItemTableViewCell
        cell.generateCell(AllItems[indexPath.row])
        return cell
    }

    //MARK: TableView Delegate
    func tableView(_ tableView: UITableView, canEditRowAt indexPath: IndexPath) -> Boo{
        return true
    }

    func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt
        indexPath: IndexPath){
        if editingStyle == .delete{
            let itemToDelete = allItems[indexPath.row]
            allItems.remove(at: indexPath.row)
            tableView.reloadData()
            removeItemFromBasket(itemId: itemToDelete.id)
            updateBasketInFirestore(basket!, withValues: [kITEMIDS: basket!.itemIds]){ (error) in
                if error != nil{
                    print("error updating the basket", error!.localizedDescription)
                }
                self.getBasketItems()
            }
        }
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){
        tableView.deselectRow(at: indexPath, animated: true)
        showItemView(withItem: allItems[indexPath.row])
    }

}
```

// WelcomeViewController.swift (Main Views) (login View)

what after in the mainstory board will be on top
// storyboard Id




```
import UIKit
import JGProgressHUD
import NVActivityIndicatorView

class WelcomeViewController: UIViewController {
    //MARK: IBOutlets
    @IBOutlets weak var emailTextField: UITextField!
    ↓ similar ones ⇒ passwordTextField, resendButtonOutlet

    //MARK: Vars
    let hud = JGProgressHUD(style: .dark)
    var activityIndicator: NVActivityIndicatorView?

    //MARK: view LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()

    }

    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        activityIndicator = NVActivityIndicatorView(frame: CGRect(x: self.view.frame.width/2 -30
            , y: self.view.frame.width/2 - 30, y: self.view.frame.height/2 -30, width: 60.0, height: 60.0
            ,type: .ballPulse, color: #colorLiteral(red: 0.9998..., green: ..., blue: ..., alpha: 1.0)
            , padding: nil)
    }
```

test buttons with print statements

```
    //MARK: IBActions
    @IBAction func cancelButtonPressed(_ sender: Any) {
            dissmissView()
    }

    @IBAction func loginButtonPressed(_ sender: Any) {
        if textFieldsHaveText(){
            loginUser()
        }else{
            hud.textLabel.text = "All fields are required"
            hud.indicatorView = JGProgressHUDErrorIndicatorView()
            hud.show(in: self.view)
            hud.dismiss(afterDelay: 2.0)
        }
    }
```

```
@IBAction func registerButtonPressed(_ sender:Any){
    if textFieldsHaveText(){
        registerUser()
    }else{
        hud.textLabel.text = "All fields are required"
        hud.indicatorView = JGProgressHUDErrorIndicatorView()
        hud.show(in: self.view)
        hud.dismiss(afterDelay: 2.0)
    }
}


@IBAction func forgetPasswordButtonPressed(_ sender :Any){
    if emailTextField.text != ""{
        resetThePassword()
    }else{
        hud.textLabel.text = "please enter email"
        hud.indicatorView = JGProgressHUDErrorIndicatorView()
        hud.show(in: self.view)
        hud.dismiss(afterDelay: 2.0)
    }


}

@IBAction func resendEmailButtonPressed(_ sender:Any){
    MUser.resendVerificationEmail(email: emailTextField.text!){ (error) in
        print("error resending email", error.localizedDescription)
    }
}

// MARK: Login User
private func loginUser(){
    showLoadingIndicator()
    MUser.loginUserWith(email: emailTextField.text!, password: passwordTextField.text!){
        (error, isEmailVerified) in
        if error == nil{
            if isEmailVerified{
                self.dissmissView()
            }else{
                self.hud.textLabel.text = "Please verify your email"
                self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
                self.resendButtonOutlet.isHidden = false
            }
        }else{
            self.hud.textLabel.text = (error!.localizedDescription)
```

```
                self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
            }
            self.hideLoadingIndicator()
        }
    }
```

```
    //MARK: Register User
    private func registerUser(){
        showLoadingIndicator()
        MUser.registerUserWith(email: emailTextField.text!, password: passwordTextField.text!) { (error) in
            if error == nil {
                self.hud.textLabel.text = "Verification Email Sent"
                self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
            } else{
                self.hud.textLabel.text = (error!.localizedDescription)
                self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
            }
            self.hideLoadingIndicator()
        }
    }

    //MARK: helpers

    private func resetThePassword(){
        MUser.resetPasswordFor(email: emailTextField.text!) { (error) in
            if error == nil
                self.hud.textLabel.text = "Reset password email sent"
                self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
            } else{
                self.hud.textLabel.text = error!.localizedDescription
                self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
                self.hud.show(in: self.view)
                self.hud.dismiss(afterDelay: 2.0)
            }
        }
    }
```

```
private func textFieldsHaveText() -> Bool{
    return (emailTextField.text != "" && passwordTextField.text != "")
}
private func dismissView(){
    self.dismiss(animated: true, completion: nil)
}
private func dismissView(){
    self.dismiss(animated: true, completion: nil)
}

//MARK: Activity Indicator
private func showLoadingIndicator(){
    if activityIndicator != nil{
        self.view.addSubview(activityIndicator!)
        activityIndicator!.startAnimating()
    }
}
private func hideLoadingIndicator(){
    if activityIndicator != nil{
        activityIndicator!.removeFromSuperView()
        activityIndicator!.stopAnimating()
    }
}
```

// ProfileTableViewController.swift (Main Views)

import UIKit

* tableViewController ← embed in nav controller
//change the table view cell from
↳ dynamic to static
select the cell ⇒ disclosure Indicator

```
class ProfileTableViewController: UITableViewController {
    //MARK: IBOutlets
    @IBOutlet weak var finishRegisterationButtonOutlet: UIButton!
    @IBOutlet weak var purchaseHistoryButtonOutlet: UIButton!

    //MARK: Vars
    var editBarButtonOutlet: UIBarButtonItem!


    //MARK: View LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.tableFooterView = UIView()  // remove the cell lines

    }
    override func viewDidAppear(_ animated: Boo(){
        super.viewDidAppear(animated)
        checkLoginStatus()
        checkOnboardlyStatus()
    }

//MARK: table view data source
    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        return 3
    }
        * make sure change this based on the number of static cells
//MARK: TableView Delegate
    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){
        tableView.deselectRow(at: indexPath, animated: true)

    }



//MARK: Helpers
    private func checkOnboardlyStatus(){
        if MUser.currentUser() != nil{
            if MUser.currentUser()!.onBoard{
                finishRegisterationButtonOutlet.setTitle("Account is Active", for: .normal)
                finishRegisterationButtonOutlet.isEnabled = false
            }else{
                finishRegisterationButtonOutlet.setTitle("Finish registration", for: .normal)
                finishRegisterationButtonOutlet.isEnabled = true
                purchaseHistoryButtonOutlet.tintColor = .red
            }
```

```
            purchaseHistoryButtonOutlet.isEnabled = true
        } else {
            finishRegisterationButtonOutlet.setTitle("Logged out", for: .normal)
            finishRegisterationButtonOutlet.isEnabled = false
            purchaseHistoryButtonOutlet.isEnabled = false
        }
    }
    private func checkLoginStatus(){
        if MUser.currentUser() == nil{
            createRightBarButton(title: "Login")
        } else{
            createRightBarButton(title:"Edit")
        }
    }
    private func createRightBarButton(title:String){
        editBarButtonOutlet = UIBarButtonItem(title:title, style:.plain, target: self,
            action: #selector(rightBarButtonItemPressed))
        self.navigationItem.rightBarButtonItem = editBarButtonOutlet

    }
    //MARK: IBActions
    @objc func rightBarButtonItemPressed(){
        if editBarButtonOutlet.title == "Login"{
            showLoginView()
        }else{
            goToEditProfile()
        }
    }

    private func showLoginView(){
        let loginView = UIStoryboard.init(name:"Main", bundle:nil).instantiateViewController(identifier: "LoginView")
        self.present(loginView, animated:true, completion:nil)
    }

    private func goToEditProfile(){
        performSegue(withIdentifier: "profileToEditSeg", sender: self)
    }
```

```
// Finish Registration View Controller : UIViewController   (main views) (login views)

import UIKit
import JGProgressHUD

class FinishRegistrationViewController : UIViewController{
    //MARK: IBOutlets
    @IBOutlet weak var nameTextField: UITextField!
    surnameTextField! addressTextField! doneButtonOutlet

    //MARK: Vars
     let hud = JGProgressHUD(style: .dark)


    //MARK: View LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
        nameTextField.addTarget(self, action: #selector(self.textFieldDidChange(_:)), for: UIControl.Event.editingChanged)
        surnameTextField.addTarget(self, action: #selector(self.textFieldDidChange(_:)), for: UIControl.Event.editingChanged)
        addressTextField.addTarget(self, action: #selector(self.textFieldDidChange(_:)), for: UIControl.Event.editingChanged)

    }

    //MARK: IBActions
    @IBAction func doneButtonPressed(_ sender: Any){
        finishOnboarding()
    }


    @IBAction func cancelButtonPressed(_ sender: Any){
        self.dismiss(animated: true, completion: nil)
    }

    @objc func textFieldDidChange(_ textField: UITextField){
        updateDoneButtonStatus()

    }

    //MARK: Helper
    private func updateDoneButtonStatus(){
        if nameTextField.text != "" && surnameTextField.text != "" && addressTextField.text != ""{
            doneButtonOutlet.backgroundColor = 
            doneButtonOutlet.isEnabled = true
        }else{
            doneButtonOutlet.backgroundColor = 
            doneButtonOutlet.isEnabled = false
        }
}
```

```
private func finishOnboarding(){
    let withValues = [kFIRSTNAME: nameTextField.text!, kLASTNAME: surnameTextField.text!, kONBOARD: true,
        kFULLADDRESS: addressTextField.text!, kFULLNAME: (nameTextField.text! + " " + surnameTextField.text!)] as [String: Any]
    updateCurrentUserInFirestore(withValues: withValues) { (error) in
        if error == nil {
            self.hud.textLabel.text = "Updated"
            self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
            self.hud.show(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)
            self.dismiss(animated: true, completion: nil)
        } else {
            print("error updating user \(error!.localizedDescription)")
            self.hud.textLabel.text = error!.localizedDescription
            self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
            self.hud.show(in: self.view)
            self.hud.dismiss(afterDelay: 2.0)
        }
    }
}
```

```
//EditProfileViewController : UIViewController    (main Views)

import UIKit
import JGProgressHUD

class EditProfileViewController : UIViewController{
    //MARK: IBOutlets
    @IBOutlet weak var nameTextField: UITextField!
    ↓ surnameTextField , addressTextField

    //MARK: Vars
     let hud = JGProgressHUD(style: .dark)

    //MARK: view lifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
        loadUserInfo()



    }

    //MARK: IBActions
    @IBAction func saveBarButtonPressed(_ sender: Any) {
            dismissKeyboard()
            if textFieldsHaveText(){
                let withValues = [kFIRSTNAME: nameTextField.text!, kLASTNAME: surnameTextField.text!, kFULLNAME:(
                    nameTextField.text! + " " + surnameTextField.text!), kFULLADDRESS: addressTextField.text!]
                updateCurrentUserInFirestore(withValues: withValues){ (error) in
                    if error == nil{
                        self.hud.textLabel.text = "Updated!"
                        self.hud.indicatorView = JGProgressHUDSuccessIndicatorView()
                        self.hud.show(in: self.view)
                        self.hud.dismiss(afterDelay: 2.0)

                    }else{
                        self.hud.textLabel.text = error!.localizedDescription
                        self.hud.indicatorView = JGProgressHUDErrorIndicatorView()
                        self.hud.show(in: self.view)
                        self.hud.dismiss(afterDelay: 2.0)
                    }
                }
            }else{
                hud.textLabel.text = "All fields are required!"
                hud.indicatorView = JGProgressHUDErrorIndicatorView()
                hud.show(in: self.view)
                hud.dismiss(afterDelay: 2.0)
            }
    }
```

```
@IBAction func logOutButtonPressed(_ sender: Any){
    logOutUser()

}

//MARK: Update UI
private func loadUserInfo(){
    if MUser.currentUser() != nil{
        let currentUser = MUser.currentUser()!
        nameTextField.text = currentUser.firstName
        surnameTextField.text = currentUser.lastName
        addressTextField.text = currentUser.fullAddress
    }
}
//MARK: Helper functions
private func dismissKeyboard(){
    self.view.endEditing(false)
}
private func textFieldsHaveText() -> Bool{
    return nameTextField.text != "" && surnameTextField.text != "" && addressTextField.text != ""
}

private func logOutUser(){

    MUser.logOutCurrentUser{ (error) in
        if error == nil{
            print("logged out")
            self.navigationController?.popViewController(animated: true)
        }else{
            print("error login out", error!.localizedDescription)
        }
    }
}

}
```

```
// Purchased History Table View Controller (Main Views)
import UIKit
import EmptyDataSet_Swift

class PurchasedHistoryTableViewController: UITableViewController {          // cell identifier
    //MARK: Vars
    var itemArray: [Item] = []

    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.tableFooterView = UIView()
        tableView.emptyDataSetDelegate = self
        tableView.emptyDataSetSource = self
    }
    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        loadItems()

    }
    MARK: table view data source
    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        return itemArray.count
    }
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell", for: indexPath) as! ItemTableViewCell
        cell.generateCell(itemArray[indexPath.row])

        return cell
    }
    // MARK: Load Items
    private func loadItems(){
        downloadItems(withIds: mUser.currentUser()!.purchasedItemIds){ (allItems) in
            self.itemArray = allItems
            self.tableView.reloadData()
        }
    }
}
extension PurchasedHistoryTableViewController: EmptyDataSetSource, EmptyDataSetDelegate{
    func title(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "No items to display!")
    }

    func image(forEmptyDataSet scrollView: UIScrollView) -> UIImage?{
        return UIImage(named: "emptyData")
    }

    func description(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "Please check back later")
    }
}
```

// Integrate paypal

// AppDelegate.swift

→ call this inside didFinishLaunchingWithOptions

```
MARK: PayPal Init
func initializePayPal(){
    PayPalMobile.initializeWithClientIds(forEnvironments:
        [payPalEnvironmentProduction: "....",
         payPalEnvironmentSandbox: "...."])
}
```

* Create a Objective-C File
file type → extension → anything name
↳ create bridging Header
↳
// AppName-Bridging-Header.h

```
#import <PayPalMobile.h>
```

// BasketViewController.swift

```
//MARK: Vars
↓ below hud
var environment: String = PayPalEnvironmentNoNetwork{
    willSet(newEnvironment){
        if (newEnvironment != environment){
            PayPalMobile.preconnect(withEnvironment: newEnvironment)
        }
    }
}
var payPalConfig = payPalConfiguration()

//MARK: View Lifecycle
override func viewDidLoad(){
    super.viewDidLoad()
    tableView.tableViewFooterView = footerView

    setupPaypal()
}
//MARK: Paypal
private func setupPayPal(){
    payPalConfig.acceptCreditCards = false
    payPalConfig.merchantName = "iOS Dev Market"        -> From paypal site
    payPalConfig.merchantPrivacyPolicyURL = URL(string: "....")
    payPalConfig.merchantUserAgreementURL = URL(string: "....")

    payPalConfig.languageOrLocale = Locale.preferredLanguages[0]
    payPalConfig.payPalShippingAddressOption = .both

}
```

```
private func payButtonPressed(){
    var itemsToBuy: [PayPalItem] = []
    for item in allItems{
        let tempItem = PayPalItem(name: item.name, withQuantity: 1, withPrice: NSDecimalNumber(value: item.price),
            withCurrency: "USD", withSku: nil)
        purchasedItemIds.append(item.id)
        itemsToBuy.append(tempItem)
    }
    let subTotal = PayPalItem.totalPrice(forItems: itemsToBuy)

    // optional
    let shippingCost = NSDecimalNumber(string: "50.0")
    let tax = NSDecimalNumber(string: "5.00")

    let paymentDetails = PayPalPaymentDetails(subtotal: subTotal, withShipping: shippingCost, withTax: tax)

    let total = subTotal.adding(shippingCost).adding(tax)
    let payment = PayPalPayment(amount: total, currencyCode: "USD", shortDescription: "Payment to ...", intent: .sale)
    payment.items = itemsToBuy
    payment.paymentDetails = paymentDetails
    if payment.processable{                                  // Do not forget conform to
        let paymentViewController = PayPalPaymentViewController(payment: payment, configuration: payPalConfig, delegate: self)
        present(paymentViewController!, animated: true, completion: nil)

    } else{
        print("Payment not processable")
    }

}

extension BasketViewController: PayPalPaymentDelegate{
    func payPalPaymentDidCancel(_ paymentViewController: PayPalPaymentViewController){
        print("...")
        paymentViewController.dismiss(animated: true, completion: nil)
    }
    func payPalPaymentViewController(_ paymentViewController: PayPalPaymentViewController, didComplete
        completedPayment: PayPalPayment){

        paymentViewController.dismiss(animated: true){
            self.addItemsToPurchaseHistory(self.purchasedItemIds)
            self.emptyTheBasket()

        }
    }
}
```

// SearchViewController.swift (main views)

```
import UIKit
import NVActivityIndicatorView
import EmptyDataSet_Swift
class SearchViewController: UIViewController{

    //MARK: - IBOutlets
    @IBOutlet weak var searchOptionsView: UIView!
    @IBOutlet weak var tableView: UITableView!
    @IBOutlet weak var searchTextField: UITextField!
    @IBOutlet weak var searchButtonOutlet: UIButton!


    //MARK: Vars
    var searchResults: [Item] = []
    var activityIndicator: NVActivityIndicatorView?

    //MARK: view LifeCycle
    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.tableFooterView = UIView()
        searchTextField.addTarget(self, action: #selector(self.textFieldDidChange(_:)), for: UIControl.Event.editingChanged)
        tableView.emptyDataSetDelegate = self
        tableView.emptyDataSetSource = self
    }
    override func viewDidAppear(_ animated: Bool){
        super.viewDidAppear(animated)
        activityIndicator = NVActivityIndicatorView(frame: CGRect(x: self.view.frame.width/2 - 30,
            y: self.view.frame.height/2 - 30, width: 60, height: 60), type: .ballPulse, color: #colorLiteral(red: 0.9998869949,
            green: 0.4941213727, blue: 0.4784867811, alpha: 1), padding: nil)
    }

    //MARK: IBActions
    @IBAction func showSearchBarButtonPressed(_ sender: Any){
        dismissKeyboard()
        showSearchField()
    }

    @IBAction func searchButtonPressed(_ sender: Any){
        if searchTextField.text != ""{
            searchInFirebase(forName: searchTextField.text!)
            emptyTextField()
            animateSearchOptionsIn()
            dismissKeyboard()
        }
    }
    //MARK: Search database
    private func searchInFirebase(forName: String){
        showLoadingIndicator()
        searchAlgolia(searchString: forName){(itemIds) in
            downloadItems(itemIds){(allItems) in
                self.searchResults = allItems
```

* embed search in the navigation controller

change the stock view's spacing

drag the data source and delegate to the view

x
y

* algolia

* make sure have both the name and description as searchable attributes

```
                self.tableView.reloadData()
                self.hideLoadingIndicator()
            }
        }
    }
    //MARK: Helpers
    private func emptyTextField(){
        searchTextField.text = ""
    }
    private func dismissKeyboard(){
        self.view.endEditing(false)
    }
    @objc func textFieldDidChange(_ textField: UITextField){
        searchButtonOutlet.isEnabled = textField.text != ""
        if searchButtonOutlet.isEnabled{
            searchButtonOutlet.backgroundColor = 
        }else{

        }
    }
    private func disableSearchButton(){
        searchButtonOutlet.isEnabled = false
        searchButtonOutlet.backgroundColor = 
    }
    private func showSearchField(){
        disableSearchButton()
        emptyTextField()
        animateSearchOptionsIn()
    }

    //MARK: Animations
    private func animateSearchOptionsIn(){
        UIView.animate(withDuration: 0.5){
            self.searchOptionsView.isHidden = !self.searchOptionsView.isHidden
        }
    }
    //MARK: Activity indicator
    private func showLoadingIndicator(){
        if activityIndicator != nil{
            self.view.addSubview(activityIndicator!)
            activityIndicator!.startAnimating()
        }
    }
    private func hideLoadingIndicator(){
        if activityIndicator != nil{
            activityIndicator!.removeFromSuperview()
            activityIndicator!.stopAnimating()
        }
    }

    private func showItemView(withItem: Item){                    * give the view controller a storyboard id
```

```
        let itemVC = UIStoryboard.init(name: "Main", bundle: nil).instantiateViewController(identifier: "ItemView") as!
            ItemViewController
        itemVC.item = withItem
        self.navigationController?.pushViewController(itemVC, animated: true)
    }
}


extension SearchViewController: UITableViewDataSource, UITableViewDelegate{

    MARK: table view data source
    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        return searchResults.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! ItemTableViewCell
        cell.generateCell(searchResults[indexPath.row])

        return cell
    }

    //MARK: TableView Delegate
    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath){
        tableView.deselectRow(at: indexPath, animated: true)
        showItemView(withItem: allItems[indexPath.row])
    }
}

extension SearchViewController: EmptyDataSetSource, EmptyDataSetDelegate{
    func title(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "No items to display!")
    }

    func image(forEmptyDataSet scrollView: UIScrollView) -> UIImage?{
        return UIImage(named: "emptyData")
    }

    func description(forEmptyDataSet scrollView: UIScrollView) -> NSAttributedString?{
        return NSAttributedString(string: "Please check back later")
    }
    func buttonImage(forEmptyDataSet scrollView: UIScrollView, for state: UIControl.State) -> UIImage?{
        return UIImage(named: "search")
    }
    func buttonTitle(forEmptyDataSet scrollView: UIScrollView, for state: UIControl.State) -> NSAttributedString?{
        return NSAttributedString(string: "Start Searching ..")
    }
    func emptyDataSet(_ scrollView: UIScrollView, didTapButton button: UIButton){
        showSearchField()
    }
}
```