SDK ⇒ software development kits

install the firebase pod $sudo gem install cocoapods hit tab will auto complete

cd ls

pod init

// open the podfile pod '.....' // multi pods

Analytics
Auth
Core
Firestore
Storage
SDWebImage

save and close the Podfile

pod install

open the workplace file .xcworkspace

// 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
    }
```

// SceneDelegate.swift

import UIKit
import Firebase

chang the Tab Bar Controller 🎞 storyBoard ID to tabBar

```
class SceneDelegate: UIResponder, UIWindowSceneDelegate{
    var window: UIWindow?
    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectingOptions){
        let currentUser = Auth.auth().currentUser
        if currentUser != nil{
            let board = UIStoryboard(name: "Main", bundle: nil)
            let tabBar = board.instantiateViewController(identifier: "tabBar") as! UITabBarController
            window?.rootViewController = tabBar
        }
        guard let _ = (scene as? UIWindowScene) else {return}
    }
```

↗ can be custom classes

// Main.storyboard

Tab bar Controller (up to 5 tab bars)

segway ⇒ ⬆ Present Modally Full Screen

add a new viewController → creat a segway and the Tab bar controller and select relationship segway

⇒ // interface failed

it got a bug ⇒ open main.storyboard as source code

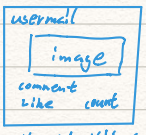
the ⓒ of ~~tab bar controller~~ class should be empty

// new File Cocoa Touch Class -> Feed Cell
subclass of UITableViewCell

// Feed Cell.swift

```
import UIKit
import Firebase

class FeedCell : UITableViewCell{

    // labels
    @IBOutlet weak var documentIdLabel : UILabel! //hidden label

    override func awakeFromNib(){
        super.awakeFromNib()

    }

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


    }

    @IBAction func likeButtonClicked(_ sender : Any){
        let fireStoreDatabase = Firestore.firestore()
        if let likeCount = Int(likeLabel.text!){
            let likeStore = ["likes": likeCount + 1] as [String : Any]
            fireStoreDatabase.collection("Posts").document(documentIdLabel.text!).setData(likeStore, merge : true)
        }
        let userEmail = userEmailLabel.text!
        fireStoreDatabase.collection("PlayerId").whereField("email", isEqualTo : userEmail).getDocuments{(snapshot, error) in
            if error == nil{
                if snapshot?.isEmpty == false && snapshot != nil{
                    for document in snapshot!.documents{
                        if let playId = document.get("player_id") as? String{
                            OneSignal.postNotification(["contents": ["en" : "\(Auth.auth().currentUser!.email!) liked"],
                                "include_player_ids": ["\(playerId)"]])

                        }
                    }
                }
            }
        }
    }
}
```

// create a new cell
+ -> prototype cell -> change from 0 to 1
select the cell and add contents in it



the cell's identifier => cell
select Table View Cell

// assign the new cell class to

Create a hidden label

// security rule

```
rule_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write : if request != null;
    }
  }
}
```

// push notifications

// One signal.com create Account
developer account .p12 file

keychain Access
click system roots
certificate Assistant ⇒ Request a Certificate From a certificate authority....

// apple.com ⇒
identifiers Certificates, Identifiers & Profiles

⊕ apple ID copy the bundle ID

do not check anything yet ⇒ create
check Notification but not configure

create a new certificate

// key chains
export the downloaded file to .p12

xcode ⇒ File ⇒ new ⇒ target ⇒ Notification service extension

exact Name: OneSignalNotificationServiceExtension ⇒ make sure exact name

⇒ suggest to click cancel for Activate "OneSignalNotificationServiceExtension" scheme?

* suggest not below iOS 10 → iOS 13 is okay

replace everything in NotificationService.swift with the code provided

copy the podfile and install the pods

+ capability → background modes and push notifications * make sure selected the correct target

add the required code to the appdelegate.swift

test with a real iphone device

go to one signal dashboard
if not working → update pods → pod repo update
pod install

↓ delete from iphone and reinstall

// ios native documation
FAQ How do I send a notification to a single user

* simulators can not receive push notifications
but: can send push notifications

// ViewController.swift

```
import UIKit
import Firebase

class ViewController : UIViewController{
    @IBOutlet weak var emailText : UITextField!
    @IBOutlet weak var passwordText : UITextField!

    override func viewDidLoad(){
        super.viewDidLoad()

    }

                                performSegue(withIdentifier: "toFeedVC", sender: nil)

    @IBAction func signInClicked(_ sender:Any){
        if emailText.text! != "" && passwordText.text! != ""{
            Auth.auth().signIn(withEmail: emailText.text!, password: passwordText.text!){(authdata, error) in
                if error != nil{                                    // set a default value
                    makeAlert(titleInput: "Error!", messageInput: error?.localizedDescription ?? "Error")
                }else{
                    self.performSegue(withIdentifier: "toFeedVC", sender: nil)
                }
            }
        }else{
            makeAlert(titleInput: "Error!", messageInput: "Username/Password?")
        }

    }
```

```
@IBAction func signUpClicked(_ sender: Any){
    if emailText.text != "" && passwordText.text != ""{
        Auth.auth().createUser(withEmail: emailText.text!, password: passwordText.text!){ (authdata, error) in
            if error != nil{
                makeAlert(titleInput: "Error!", messageInput: error?.localizedDescription ?? "Error")  // set a default value
            }else{
                self.performSegue(withIdentifier: "toFeedVC", sender: nil)
            }
        }
    }else{
        makeAlert(titleInput: "Error!", messageInput: "Username/Password?")
    }
}

func makeAlert(titleInput: String, messageInput: String){
    let alert = UIAlertController(title: titleInput, message: messageInput, preferredStyle:
        UIAlertController.Style.alert)
    let okButton = UIAlertAction(title: "OK", style: UIAlertAction.Style.default, handler: nil)
    alert.addAction(okButton)
    self.present(alert, animated: true, completion: nil)
}
```

// SettingsViewController.swift

```
import UIKit
import Firebase

class SettingsViewController: UIViewController{

    override func viewDidLoad(){
        super.viewDidLoad()

    }
    @IBAction func logoutClicked(_ sender: Any){
        do{
            try Auth.auth().signOut()
            self.performSegue(withIdentifier: "toViewController", sender: nil)

        }catch{
            print("error")
        }
    }
}
```

```
// UploadViewController.swift
import UIKit
import Firebase


class UploadViewController : UIViewController, UIImagePickerControllerDelegate , UINavigationControllerDelegate {
    @IBOutlet weak var imageView : UIImageView!
    ↓ similar ones

    override func viewDidLoad(){
        super.viewDidLoad()

        imageView.isUserInteractionEnabled = true
        let imageTapRecognizer = UITapGestureRecognizer(target: self, action: #selector(selectImage))
        imageView.addGestureRecognizer(imageTapRecognizer)
}

    @objc func selectImage(){
        let picker = UIImagePickerController()
        picker.delegate = self
        picker.sourceType = .photoLibrary
        picker.allowEditing = true
        present(picker, animated: true, completion: nil)
}

func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info:
    [UIImagePickerController.InfoKey : Any]){
        imageView.image = info[.originalImage] as? UIImage

        saveButton.isEnabled = true

        self.dismiss(animated: true, completion: nil)
}
//before IOS12 => info.plist => plus button => Privacy - Photo Library Usage Description => Value: to access photos
                                                                                  ← explaination

@IBAction func saveButtonCliched(_ sender: Any){

    let storage = Storage.storage()
    let storageReference = storage.reference()

    let mediaFolder = storageReference.child("media")

    if let data = imageView.image?.jpegData(compressionQuality: 0.5){
        let uuid = UUID().uuidString
        let imageReference = mediaFolder.child("\(uuid).jpg")
        imageReference.putData(data, metadata: nil){ (metaData, error) in
            if error != nil{
                self.makeAlert(titleInput: "Error!", messageInput: error?.localizedDescription ?? "error")
            }else{
                imageReference.downloadURL{ (url, error) in
                    if error == nil{
                        let imageUrl = url?.absoluteString
```

```
// database        collection ⇒ document ⇒ field or collections
let firestoreDatabase = Firestore.firestore()
var firestoreReference: DocumentReference? = nil
let firestorePost = ["imageUrl": imageUrl!, "postBy": Auth.auth().currentUser!.email!, "postComment":
        self.commentText.text!, "date": FieldValue.serverTimestamp(), "likes": 0] as [String: Any]
firestoreReference = firestoreDatabase.collection("Posts").addDocument(data: firestorePost, completion:{(error) in
    if error != nil{
        self.makeAlert(titleInput: "Error!", messageInput: error?.localizedDescription ?? "error")
    } else{
        self.imageView.image = UIImage(named: "select.png")
        self.commentText.text = ""
        self.tabBarController?.selectedIndex = 0
    }
})
}
}
}
}
}
}

func makeAlert(titleInput: String, messageInput: String){
    let alert = UIAlertController(title: titleInput, message: messageInput, preferredStyle:
        UIAlertControllerStyle.alert)
    let okButton = UIAlertAction(title: "Ok", style: UIAlertActionStyle.default, handler: nil)
    alert.addAction(okButton)
    self.present(alert, animated: true, completion: nil)
}
```

// FeedViewController.swift

```
import UIKit
import Firebase
import SDWebImage
import OneSignal
```

→ if not showing up → Build Phases → link Binary with Libraries → + SDWebImage

// SDWebImage

pod 'SDWebImage' & close xcode first

```
class FeedViewController : UIViewController, UITableViewDelegate, UITableViewDataSource{
    @IBOutlet weak var tableView : UITableView!

    var userEmailArray = [String]()
    var userCommentArray = [String]()
    var likeArray = [Int]()
    var userImageArray = [String]()
    var documentIdArray = [String]()
    let fireStoreDatabase = Firestore.firestore()

    override func viewDidLoad(){
        super.viewDidLoad()
        tableView.delegate = self
        tableView.dataSource = self
        getDataFromFirestore()
        // OneSignal.postNotification(["contents":["en": "Test Message"], "include_player_ids":["  .....  "]])
        // playerIDs
        let status : OSPermissionSubscriptionState = OneSignal.getPermissionSubscriptionState()
        let playerId = status.subscriptionStatus.userId
        if let playerNewId = playerId{
            fireStoreDatabase.collection("PlayerId").whereField("email", isEqualTo: Auth.auth().currentUser!.email!)
                .getDocuments{ (snapshot, error) in
                    if error == nil{
                        if snapshot?.isEmpty == false && snapshot != nil{
                            for document in snapshot!.documents{
                                if let playerIDFromFirebase = document.get("player_id") as? String{
                                    let documentId = document.documentID
                                    if playerNewId != playerIDFromFirebase{
                                        let playerIdDictionary = ["email": Auth.auth().currentUser!.email!, "player_id" : playerNewId] as! [String:Any]
                                        fireStoreDatabase.collection("PlayerId").addDocument(data: playerIdDictionary){ (error) in
                                            if error != nil{
                                                print(error?.localizedDescription)
                                            }
                                        }
                                    }
                                }
                            }
                        } // // if the snapshot is not empty
                        else{
                            let playerIdDictionary = ["email": Auth.auth().currentUser!.email!, "player_id" : playerNewId] as! [String:Any]
                            fireStoreDatabase.collection("PlayerId").addDocument(data: playerIdDictionary){ (error) in
                                if error != nil{
                                    print(error?.localizedDescription)
                                }
                            }
                        }
                    }
                }
        }
    }
```

playerID

```
func getDataFromFirestore(){

    /* let settings = fireStoreDatabase.settings
    settings.areTimestampsInSnapshotsEnabled = true  // may not need this because the default setting is true
    fireStoreDatabase.settings = settings */
    fireStoreDatabase.collection("Posts").order(by: "date", descending: true).addSnapshotListener { (snapshot, error) in
        if error != nil {
            print(error?.localizedDescription)
        } else {
            if snapshot?.isEmpty != true && snapshot != nil {
                self.userImageArray.removeAll(keepingCapacity: false)
                // similar ones, documentIdArray
                for document in snapshot!.documents {
                    let documentID = document.documentID
                    self.documentIdArray.append(documentID)

                    if let postedBy = document.get("postedBy") as? String {   // can be nested if statements
                        self.userEmailArray.append(postedBy)
                    }
                    // similar ones

                }
                self.tableView.reloadData()
            }
        }
    }
}

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

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! FeedCell
    cell.userEmailLabel.text = userEmailArray[indexPath.row]
    cell.likeLabel.text = String(likeArray[indexPath.row])
    cell.commentLabel.text = userCommentArray[indexPath.row]
    cell.userImageView.sd_setImage(with: URL(string: self.userImageArray[indexPath.row]))
    cell.documentIdLabel.text = documentIdArray[indexPath.row]

    return cell
}
```