pagnation

command + R ⇒ run in simulator

Firebase → bundleID → down p-list * copy item if needed should be selected

/Auth

/Database

/Storag

Firebase security rules:

//Database

```
{
  "rules":{
    ".read":"true",
    ".write":"true"
  }
}
```

//Storage

// Utils

Extensions.swift

// Model

User.swift

// View

UserProfileHeader.swift

// Controller

- MainTabVC.swift
- Login
  - LoginVC.swift
  - SignUpVC.swift
- Feed
  - FeedVC.swift
- Search
  - SearchVC.swift
- UploadPost
  - UploadPostVC.swift
- Notifications
  - NotificationsVC.swift
- UserProfile
  - UserProfileVC.swift

App Delegate

```
//App Delegate

import UIKit
import Firebase

func didFinishLaunchingWithOptions() -> Bool {
    FirebaseApp.configure()

    window = UIWindow()
    window?.rootViewController = UINavigationController(rootViewController: LoginVC())

    return true
}
```

txt

```
//Extensions.swift (Utils)

import UIKit

extension UIView{
    func anchor(top: NSLayoutYAxisAnchor?, Left: NSLayoutXAxisAnchor?, bottom: NSLayoutYAxisAnchor?, right: NSLayoutXAxisAnchor?,
        paddingTop: CGFloat, paddingLeft: CGFloat, paddingBottom: CGFloat, paddingRight: CGFloat, width: CGFloat, height: CGFloat){
        translatesAutoresizingMaskIntoConstraints = false
        if let top = top{
            self.topAnchor.constraint(equalTo: top, constant: paddingTop).isActive = true
        }
        if let left = left{
            self.leftAnchor.constraint(equalTo: left, constant: paddingLeft).isActive = true
        }
        if let bottom = bottom{
            self.bottomAnchor.constraint(equalTo: bottom, constant: -paddingBottom).isActive = true
        }
        if let right = right{
            self.rightAnchor.constraint(equalTo: right, constant: -paddingRight).isActive = true
        }
        if width != 0{
            widthAnchor.constraint(equalToConstant: width).isActive = true
        }

        if height != 0{
            heightAnchor.constraint(equalToConstant: height).isActive = true
        }
    }
}


var imageCache = [String: UIImage]()
extension UIImageView{
    func loadImage(with urlString: String){
        if let cachedImage = imageCache[urlString]{
            self.image = cachedImage
            return
        }

        guard let url = URL(string: urlString) else {return}

        URLSession.shared.dataTask(with: url){(data, response, error) in
            if let error = error{
                print("Failed to load image with error", error.localizedDescription)
            }
            guard let imageData = data else {return}
            let photoImage = UIImage(data: imageData)
            imageCache[url.absoluteString] = photoImage
            DispatchQueue.main.async{
                self.image = photoImage
            }
        }.resume()
    }
}
```

Protocols

~~//Protocols.swift~~ (Utils)

```
protocol UserProfileHeaderDelegate {
    func handleEditFollowTapped(for header: UserProfileHeader)
```

~~// User.swift~~ (Model)

User

```
class User {
    var username: String!
    var name: String!
    var profileImageUrl: String!
    var uid: String!
    var isFollowed = false


    init(uid: String, dictionary: Dictionary<String, AnyObject>){
        self.uid = uid

        if let username = dictionary["username"] as? String {
            self.username = username
        }

        if let name = dictionary["name"] as? String {
            self.name = name
        }

        if let profileImageUrl = dictionary["profileImageUrl"] as? String {
            self.profileImageUrl = profileImageUrl
        }

        func follow(){
            guard let currentUid = Auth.auth().currentUser?.uid else {return}
            guard let uid = uid else {return}

            self.isFollowed = true

            USER_FOLLOWING_REF.child(currentUid).updateChildValues([uid:1])
            USER_FOLLOWER_REF.child(uid).updateChildValues([currentUid:1])

            uploadFollowingNotificationToServer()
            USER_POSTS_REF.child(uid).observe(.childAdded){(snapshot) in
                let postId = snapshot.key
                USER_FEED_REF.child(currentUid).updateChildValues([postId:1])
            }
        }

        func unfollow(){
            guard let currentUid = Auth.auth().currentUser?.uid else {return}
            guard let uid = uid else {return}

            self.isFollowed = false

            USER_FOLLOWING_REF.child(currentUid).child(uid).removeValue()
            USER_FOLLOWER_REF.child(uid).child(currentUid).removeValue()
```

```
        USER_POSTS_REF.child(uid).observe(.childAdded) { (snapshot) in
            let postId = snapshot.key
            USER_FEED_REF.child(currentUid).child(postId).removeValue()
        }
    }

    func checkIfUserIsFollowed(completion: @escaping (Bool) -> ()) {
        guard let currentUid = Auth.auth().currentUser?.uid else {return}

        User_FOLLOWING_REF.child(currentUid).observeSingleEvent(of: .value) { (snapshot) in
            if snapshot.hasChild(self.uid) {
                self.isFollowed = true
                completion(true)
            } else {
                self.isFollowed = false
                completion(false)
            }
        }
    }

}
```

// UserProfileHeader.swift (View)

Profile Header

```
import UIKit
import Firebase
class UserProfileHeader: UICollectionViewCell {

    var delegate: UserProfileHeaderDelegate?

    var user: User? {
        didSet {
            configureEditProfileFollowButton()

            let fullName = user?.name
            nameLabel.text = fullName

            guard let profileImageUrl = user?.profileImageUrl else { return }
            profileImageView.loadImage(with: profileImageUrl)

        }
    }


    let profileImageView: UIImageView = {
        let iv = UIImageView()
        iv.contentMode = .scaleAspectFill
        iv.clipsToBounds = true
        iv.backgroundColor = .lightGray
        return iv
    }()

    let nameLabel: UILabel = {
        let label = UILabel()
        label.text = " "
        label.font = UIFont.boldSystemFont(ofSize: 12)
        return label
    }()

    let postsLabel: UILabel = {
        let label = UILabel()
        label.numberOfLines = 0
        label.textAlignment = .center
        let attributedText = NSMutableAttributedString(string: "5\n", attributes: [NSAttributedStringKey.font: UIFont.
            boldSystemFont(ofSize: 14)])
        attributedText.append(NSAttributedString(string: "posts", attributes: [NSAttributedStringKey.font: UIFont.
            systemFont(ofSize: 14), NSAttributedStringKey.foregroundColor: UIColor.lightGray]))
        return label
    }()
```

↓ similar for followersLabel followingLabel

```
let editProfileButton : UIButton = {
    let button = UIButton(type: .system)
    button.setTitle("Loading ", for: .normal)
    button.layer.cornerRadius = 3
    button.layer.borderColor = UIColor.lightGray.cgColor
    button.layer.borderWidth = 0.5
    button.titleLabel?.font = UIFont.boldSystemFont(ofSize: 14)
    button.setTitleColor(.black, for: .normal)
    button.addTarget(self, action: #selector(handleEditProfileFollow), for: .touchUpInside)

    return button
}()



let gridButton : UIButton = {
    let button = UIButton(type: .system)
    button.setImage(:::, for: .normal)
    button.tintColor = UIColor(white: 0, alpha: 0.2)
    return button
}()
```

↓ similar for listbutton and bookmarkButton

```
override init(frame: CGRect){
    super.init(frame: frame)
    addSubview(profileImageView)
    profileImageView.anchor(top: self.topAnchor, left: self.leftAnchor, bottom: nil, right: nil, paddingTop: 16, paddingLeft:
        12, paddingBottom: 0, paddingRight: 0, width: 80, height: 80)
    profileImageView.layer.cornerRadius = 80/2

    addSubView(nameLabel)
    nameLabel.anchor(top: profileImageView.bottomAnchor, left: self.leftAnchor, bottom: nil, right: nil,
        paddingTop: 12, paddingLeft: 12, paddingBottom: 0, paddingRight: 0, width: 0, height: 0)

    configureUserStatus()

    addSubview(editProfileButton)
    editProfileButton.anchor(top: postsLabel.bottomAnchor, left: postsLabel.leftAnchor, bottom: nil, right: self.
        rightAnchor, paddingTop: 12, paddingLeft: 8, paddingBottom: 0, paddingRight: 4, width: 0,
        height: 30)

    configureBottomToolBar()
}
```

```
@objc func handleEditProfileFollow(){
    //delegate?.handleEditFollowTapped(for: self)

    guard let user = self.user else {return}

    if editProfileFollowButton.titleLabel?.text == "Edit Profile"{
        print("Handle edit profile")
    }else{
        if editProfileFollowButton.titleLabel?.text == "Follow"{
            editProfileFollowButton.setTitle("Following", for: .normal)
            user.follow()
        }else{
            editProfileFollowButton.setTitle("Follow", for: .normal)
            user.unfollow()
        }
    }
}
```

```
func configureBottomToolBar(){

    let topDividerView = UIView()
    topDividerView.backgroundColor = .lightGray
    let bottomDividerView = UIView()
    bottomDividerView.backgroud = .lightGray

    let stackView = UIStackView(arrangedSubviews: [gridButton, listButton, bookmarkButton])
    stackView.axis = .horizontal
    stackView.distribution = .fillEqually

    addSubview(stackView)
    addSubview(topDividerView)
    addSubview(bottomDividerView)

    stackView.anchor(top: nil, left: leftAnchor, bottom: self.bottomAnchor, right: rightAnchor,
        paddingTop: 0, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 50)

    topDividerView.(top: stackView.topAnchor, left: leftAnchor bottom: nil, right: rightAnchor,
        paddingTop: 0, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 0.5)

    bottomDividerView.(top: stackView.bottomAnchor, left: leftAnchor, bottom: nil, right: rightAnchor,
        paddingTop: 0, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 0.5)
}
```

```
func configureUserStatus(){
    let stackView = UIStackView(arrangedSubviews:[postsLabel, followersLabel, followingLabel])
    stackView.axis = .horizontal
    stackView.distribution = .fillEqually
    addSubview(stackView)
    stackView.anchor(top: topAnchor, left: profileImageView.rightAnchor, bottom: nil, right: rightAnchor,
        paddingTop: 12, paddingLeft: 12, paddingBottom: 0, paddingRight: 12, width: 0, height: 50)
}

func setUserStats(uid: String){
    var numberOfFollowers: Int!
    var numberOfFollowing: Int!



func configureEditProfileFollowButton(){
    guard let currentUid = Auth.auth().currentUser?.uid else {return}
    guard let user = self.user else {return}

    if currentUid == user.uid{
        editProfileFollowButton.setTitle("Edit Profile", for: .normal)
    } else{
        editProfileFollowButton.setTitleColor(.white, for: .normal)
        editProfileFollowButton.backgroundColor = UIColor(red: 17/255, green: 154/255, blue: 237/255, alpha: 1)

        user.checkIfUserIsFollowed(completion: {(followed) in
            if followed{
                self.editProfileFollowButton.setTitle("Following", for: .normal)
            } else{
                self.editProfileFollowButton.setTitle("Follow", for: .normal)

        })
    }
}

required init?(coder aDecoder: NSCoder){
    fatalError("init(coder:) has not been implemented")
}
}
```

~~//SearchUserCell : UITableViewCell~~ (View)

Search Cell View

```
import UIKit

class SearchUserCell: UITableViewCell{

    var user: User?{
        didSet{
            guard let profileImageUrl = user?.profileImageUrl else {return}
            guard let username = user?.username else {return}
            guard let fullname = user?.name else {return}

            profileImageView.loadImage(with: profileImageUrl)
            self.textLabel?.text = username
            self.detailTextLabel?.text = fullName
        }
    }

    let profileImageView: UIImageView = {
        let iv = UIImageView()
        iv.contentMode = .scaleAspectFill
        iv.clipsToBounds = true
        iv.backgroundColor = .lightGray
        return iv
    }()

    override init(style: UITableViewCellStyle, reuseIdentifier: String?){
        super.init(style: .subtitle, reuseIdentifier: reuseIdentifier)

        addSubview(profileImageView)
        profileImageView.anchor(top: nil, left: self.leftAnchor, bottom: nil, right: nil, paddingTop: 0, paddingLeft: 0
            , paddingBottom: 0, paddingRight: 0, width: 48, height: 48)
        profileImageView.centerYAnchor.constraint(equalTo: self.centerYAnchor).isActive = true
        profileImageView.layer.cornerRadius = 48/2
        profileImageView.clipsToBounds = true

        self.textLabel?.text = "Username"
        self.detailTextLabel?.text = "Full name"

    }
```

```
override func layoutSubviews() {
    super.layoutSubviews()

    textLabel?.frame = CGRect(x: 68, y: textLabel!.frame.origin.y - 2, width: (textLabel?.frame.width)!,
        height: (textLabel?.frame.height)!)
    textLabel?.font = UIFont.boldSystemFont(ofSize: 12)

    detailTextLabel?.frame = CGRect(x: 68, y: detailTextLabel!.frame.origin.y, width: self.frame
        .width - 108, height: detailTextLabel!.frame.height)
    detailTextLabel?.textColor = .lightGray
    detailTextLabel?.font = UIFont.systemFont(ofSize: 12)
}

required init?(coder aDecoder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
}
```

// MainTabVC.swift (Controller)

Main Tab

```
import UIKit
import Firebase
class MainTabVC : UITabBarController, UITabBarControllerDelegate{

    override func viewDidLoad(){
        super.viewDidLoad()
        self.delegate = self
        configureViewControllers()
        checkIfUserIsLoggedIn()
    }

    func configureViewControllers(){
        let feedVC = constructNavController(unselectedImage: ..., selectedImage: ..., rootViewController: FeedVC(collectionViewLayout:
            UICollectionViewFlowLayout()))

        let searchVC = constructNavController(unselectedImage: ..., selectedImage: ..., rootViewController: searchVC())

        ↓ similar for other controllers

        viewControllers = [FeedVC, searchVC, uploadPostVC, notificationVC, userProfileVC]
        tabBar.tintColor = .black

    }

    func constructNavController(unselectedImage: UIImage, selectedImage: UIImage, rootViewController: UIViewController = UIViewController())
        -> UINavigationController{
        let navController = UINavigationController(rootViewController: rootViewController)
        navController.tabBarItem.image = unselectedImage
        navController.tabBarItem.selectedImage = selectedImage
        navController.navigationBar.tintColor = .black

        return navController

    }

    func checkIfUserIsLoggedIn(){
        if Auth.auth().currentUser == nil{
            DispatchQueue.main.async{
                let loginVC = LoginVC()
                let navController = UINavigationController(rootViewController: loginVC)
                self.present(navController, animated: true, completion: nil)
            }
            return
        }
    }
```

//SearchVC.swift (Controller)

SearchVC

```
import UIKit

private let reuseIdentifier = "SearchUserCell"
class SearchVC: UITableViewController {

    //Properties
    var users = [User]()


    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.register(SearchUserCell.self, forCellReuseIdentifier: reuseIdentifier)
        tableView.separatorInset = UIEdgeInsets(top:0, left: 64, bottom:0, right:0)

        configureNavController()
        fetchUsers()

    }


    override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        return 60
    }


    override func numberOfSections() -> Int {
        return 1

    }

    override func ... numberOfRowsInSection ..) -> Int {

        return users.count
    }

    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let user = users[indexPath.row]

        let userProfileVC = UserProfileVC(collectionViewLayout: UICollectionViewFlowLayout())
        userProfileVC.userToLoadFromSearchVC = user

        navigationController?.pushViewController(userProfileVC, animated: true)

    }
```

```
override func ..... cellForRowAt --- ) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: reuseIdentifier, for: indexPath) as!
        SearchUserCell
    cell.user = users[indexPath.row]


    return cell
}

//MARK: - Handlers
func configureNavController() {
    navigationItem.title = "Explore"
}


//MARK: - API
func fetchUsers() {
    Database.database().reference().child("Users").observe(.childAdded) { (snapshot) in
        let uid = snapshot.key
        guard let dictionary = snapshot.value as? Dictionary<String, AnyObject> else {return}

        let user = User(uid: uid, dictionary: dictionary)
        self.users.append(user)
        self.tableView.reloadData()
    }
}
```

// LoginVC.swift (login)

Login

```
import UIKit
import Firebase


let logoContainerView: UIView = {
    let view = UIView()
    let logoImageView = UIImageView(image: ....)
    logoImageView.contentMode = .scaleAspectFill
    view.addSubview(logoImageView)
    logoImageView.anchor(top: nil, left: nil, bottom: nil, right: nil, paddingTop: 0, paddingLeft: 0, paddingBottom: 0,
        paddingRight: 0, width: 200, height: 50)
    logoImageView.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
    logoImageView.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    view.backgroundColor = UIColor(red: 0/255, green: 120/255, blue: 175/255, alpha: 1)
    return view
}()
let emailTextField: UITextField = {
    let tf = UITextField()
    tf.placeholder = "Email"
    tf.backgroundColor = UIColor(white: 0, alpha: 0.03)
    tf.borderStyle = .roundedRect
    tf.font = UIFont.systemFont(ofSize: 14)
    tf.addTarget(self, action: #selector(formValidation), for: .editingChanged)
    return tf
}()

let passwordTextField: UITextField = {
    let tf = UITextField()
    tf.placeholder = "Password"
    tf.backgroundColor = UIColor(white: 0, alpha: 0.03)
    tf.borderStyle = .roundedRect
    tf.font = UIFont.systemFont(ofSize: 14)
    tf.isSecureTextEntry = true
    tf.addTarget(self, action: #selector(formValidation), for: .editingChanged)
    return tf
}()
```

```
let loginButton: UIButton = {
    let button = UIButton(type: .system)
    button.setTitle("Login", for: .normal)
    button.setTitleColor(.white, for: .normal)
    button.backgroundColor = UIColor(red: 149/255, green: 204/255, blue: 244/255, alpha: 1)
    button.addTarget(self, action: #selector(handleLogin), for: .touchUpInside)
    button.isEnabled = false
    button.layer.cornerRadius = 5
    return button
}()

let dontHaveAccountButton: UIButton = {
    let button = UIButton(type: .system)
    let attributedTitle = NSMutableAttributedString(string: "Dont have an account?  ", attributes: [NSAttributedStringKey.font:
        UIFont.boldSystemFont(ofSize: 14), NSAttributedStringKey.foregroundColor: UIColor(red: 17/255, green: 154/255, blue:
        237/255, alpha: 1)])
    attributedTitle.append(NSAttributedString(string: "Sign Up", attributes: [NSAttributedStringKey.font: UIFont.boldSystemFont(ofSize: 14),
        NSAttributedStringKey.foregroundColor: UIColor(red: 17/255, green: 154/255, blue: 237/255, alpha: 1)]))
    button.addTarget(self, action: #selector(handleShowSignUp), for: .touchUpInside)
    button.setAttributedTitle(attributedTitle, for: .normal)
    return button
}()


override func viewDidLoad(){
    super.viewDidLoad()
    view.backgroundColor = .white
    navigationController?.navigationBar.isHidden = true

    view.addSubview(logoContainerView)
    logoContainerView.anchor(top: view.topAnchor, left: view.leftAnchor, bottom: nil, right: view.rightAnchor, paddingTop: 0,
        paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 150)
    configureViewComponents()
    view.addSubview(dontHaveAccountButton)
    dontHaveAccountButton.anchor(top: nil, left: view.leftAnchor, bottom: view.bottomAnchor, right: view.rightAnchor, paddingTop:
        0, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 50)

}

@objc func handleShowSignUp(){
    let signUpVC = SignUpVC
    navigationController?.pushViewController(signUpVC, animated: true)

}


@objc func handleLogin(){
    guard
        let email = emailTextField.text,
        let password = passwordTextField.text else {return}
```

```
Auth.auth().signIn(withEmail: email, password: password) { (user, error) in
    if let error = error {
        print("Unable to sign user in with error", error.localizedDescription)
        return
    }

    guard let mainTabVC = UIApplication.shared.keyWindow?.rootViewController as? MainTabVC else {return}
    mainTabVC.configureViewControllers()
    self.dismiss(animated: true, completion: nil)

  }
}



@objc func formValidation() {
    guard
        emailTextField.hasText,
        passwordTextField.hasText else {
            loginButton.isEnabled = false
            loginButton.backgroundColor = UIColor(red: 149/255, green: 204/255, blue: 244/255, alpha: 1)
            return
    }

    loginButton.isEnabled = false
    loginButton.backgroundColor = UIColor(red: 17/255, green: 154/255, blue: 237/255, alpha: 1)


func configureViewComponents() {
    let stackView = UIStackView(arrangedSubviews: [emailTextField, passwordTextField, loginButton])
    stackView.axis = .vertical
    stackView.spacing = 10
    stackView.distribution = .fillEqually
    view.addSubview(stackView)
    stackView.anchor(top: logoContainerView.bottomAnchor, left: view.leftAnchor, bottom: nil, right: view.rightAnchor, paddingTop
        40, paddingLeft: 40, paddingBottom: 0, paddingRight: 40, width: 0, height: 140)
}
```

~~//SignUpVC.swift~~ (Login)

Sign Up

```
import UIKit
import Firebase

class SignUpVC : UIViewController, UIImagePicherControllerDelegate, UINavigationControllerDelegate{
    let imageSelected = false


    let plusPhotoBtn : UIButton = {
        let button = UIButton(type: .system)
        button.setImage(... .withRenderingMode(.alwaysOriginal), for: .normal)
        button.addTarget(self, action: #selector(handleSelectProfilePhoto), for: .touchUpInside)
        return button
    }()

    let emailTextField: UITextField = {
        let tf = UITextField()
        tf.placeholder = "Email"
        tf.backgroundColor = UIColor(white:0, alpha:0.03)
        tf.borderStyle = .roundedRect
        tf.font = UIFont.systemFont(ofSize:14)
        tf.addTarget(self, action: #selector(formValidation), for: .editingChanged)
        return tf
    }()

    other similar ones: password, fullName, username
                        tf.isSecureTextEntry = true
                        tf.addTarget(self, action: #selector(formValidation), for: .editingChanged)

    let signUpButton: UIButton = {
        let button = UIButton(type: .system)
        button.setTitle("Sign Up", for .normal)
        button.setTitleColor(.white, for .normal)
        button.backgroundColor = UIColor(red: 149/255, green: 204/255, blue: 244/255, alpha: 1)
        button.layer.cornerRadius = 5
        button.isEnabled = false
        button.addTarget(self, action: #selector(handleSignUp), for: .touchUpInside)
        return button
    }()

    let alreadyHaveAccountButton: UIButton = {
        let button = UIButton(type: .system)
        let attributedTitle = NSMutableAttributedString(string: "Already have an account?  ", attributes: [NSAttributedStringKey.font:
            UIFont.boldSystemFont(ofSize:14), NSAttributedStringKey.foregroundColor: UIColor(red: 17/255, green: 154/255, blue:
            237/255, alpha: 1)]))

        attributedTitle.append(NSAttributedString(string: "Sign In", attributes:[NSAttributedStringKey.font: UIFont.boldSystemFont(ofSize:14),
            NSAttributedStringKey.foregroundColor: UIColor(red: 17/255, green: 154/255, blue: 237/255, alpha: 1)]))
        button.addTarget(self, action: #selector(handleShowLoginIn), for: .touchUpInside)
        button.setAttributedTitle(attributedTitle, for: .normal)
        return button
    }()
```

```
override func viewDidLoad(){
    super.viewDidLoad()
    view.backgroundColor = .white
    view.addSubview(plusPhotoBtn)
    plusPhotoBtn.anchor( top: view.TopAnchor left: nil, bottom: nil, right: nil, paddingTop
        40, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 140, height: 140)
    plusPhotoBtn.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
    configureViewComponents()

    view.addSubview(alreadyHaveAccountButton)
    alreadyHaveAccountButton.anchor( top: nil, left: view.leftAnchor, bottom: view.bottomAnchor, right: view.rightAnchor, paddingTop:
        0, paddingLeft: 0, paddingBottom: 0, paddingRight: 0, width: 0, height: 50)
}


func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [String: Any]){
    guard let profileImage = info[UIImagePickerControllerEditedImage] as? UIImage else {
        imageSelected = false
        return
    }
    imageSelected = true
    plusPhotoBtn.layer.cornerRadius = plusPhotoBtn.frame.width/2
    plusPhotoBtn.layer.masksToBounds = true
    plusPhotoBtn.layer.borderColor = UIColor.black.cgColor
    plusPhotoBtn.layer.borderWidth = 2
    plusPhotoBtn.setImage(profileImage.withRenderingMode(.alwaysOriginal), for: .normal)

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


@objc func handleSelectProfilePhoto(){
    // configure image picker
    let imagePicker = UIImagePickerController()
    imagePicker.delegate = self
    imagePicker.allowsEditing = true

    // present image picker
    self.present(imagePicker, animated: true, completion: nil)
}
```

```
@objc func handleShowLogin(){
    _ = navigationController?.popViewController(animated: true)
}

@objc func handleSignUp(){
    guard let email = emailTextField.text else {return}
    guard let password = passwordTextField.text else {return}
    guard let fullName = fullNameTextField.text else {return}
    guard let userName = usernameTextField.text?.lowercased else {return}

    Auth.auth().createUser(withEmail: email, password: password) { (user, error) in
        //handle error
        if let error = error{
            print("Failed to create user with error: ", error.localizedDescription)
            return
        }

        guard let profileImg = self.plusPhotoBtn.imageView?.image else {return}

        guard let uploadData = UIImageJPEGRepresentation(profileImg, 0.3) else {return}
        let fileName = NSUUID().uuidString
        Storage.storage().reference().child("profile_images").child(fileName).putData(uploadData, metaData:
            nil, completion: {(metaData, error) in
                if let error = error{
                    print("Failed to upload Image", error.localizedDescription)
                }
                guard let profileImageURL = metadata?.downloadURL()?.absoluteString else {return}
                guard let uid = user?.uid else {return} // uid from firebase
                let dictionaryValues = ["name": fullName,
                                        "username": username,
                                        "profileImageUrl": profileImageURL]
                let values = [user?.uid: dictionaryValues]

                Database.database().reference().child("users").updateChildValues(values, withCompletionBlock:{
                    (error, ref) in
                    guard let mainTabVC = UIApplication.shared.keyWindow?.rootViewController as? MainTabVC else {return}
                    mainTabVC.configureViewControllers()
                    self.dismiss(animated: true, completion: nil)

                })

            })
    }
}
```

```
@objc func formValidation(){
    guard
        emailTextField.hasText,
        passwordTextField.hasText
        fullNameTextField.hasText
        userNameTextField.hasText
        imageSelected == true else{
         signUpButton.isEnabled = false
         signUpButton.backgroundColor = UIColor(red: 149/255, green: 204/255, blue: 244/255, alpha: 1)
         return
    }
    signUpButton.isEnabled = true
    signUpButton.backgroundColor = UIColor(red: 17/255, green: 154/255, blue: 237/255, alpha: 1)
}

func configureViewComponents(){
    let stackView = UIStackView(arrangeSubviews: [emailTextField, → fullName, userName, password ])
    stackView.axis = .vertical
    stackView.spacing = 10
    stackView.distribution = .fillEqually
    view.addSubview(stackView)
    stackView.anchor(top: plusPhotoBtn.bottomAnchor, left: view.leftAnchor, bottom: nil, right: view.rightAnchor, paddingTop:
        20, paddingLeft: 40, paddingBottom: 0, paddingRight: 40, width: 0, height: 240)
}
```

~~//FeedVC.swift~~ (Feed)

Feed

```
import UIKit
import Firebase

private let reuseIdentifier = "Cell"

class FeedVC: UICollectionViewController {
    //MARK: -Properties

    override func viewDidLoad() {
        super.viewDidLoad()

        self.collectionView!.register(UICollectionViewCell.self, forCellWithReuseIdent
            ifier: reuseIdentifier)
        configureLogoutButton()
    }

    override func numberOfSections {
        return 0
    }

    override func ---- numberOfItemsInSection {
        return 0
    }

    override func ----- cellForItemAt {


    }


    func configureLogoutButton() {
        self.navigationItem.leftBarButtonItem = UIBarButtonItem(title: "Logout", style: .plain, target: self, action: #selector(
            handleLogOut))
    }
```

```
@objc func handleLogout(){
    let alertController = UIAlertController(title: nil, message: nil, preferredStyle: .actionSheet)
    alertController.addAction(UIAlertAction(title: "Log Out", style: .destructive, handler: {(_) in
        do{
            try Auth.auth().signOut()

            let loginVC = LoginVC()
            let navController = UINavigationController(rootViewController: loginVC)
            self.present(navController, animated: true, completion: nil)
            print("Success logout")
        }catch{
            print("failed to logout")
        }
    }))
    alertController.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: nil))
    present(alertController, animated: true, completion: nil)
}
```

Profile VC

```
// UserProfileVC.swift
import UIKit
import Firebase
private let reuseIdentifier = "Cell"
private let headerIdentifier = "UserProfileHeader"
class UserProfileVC: UICollectionViewController, UICollectionViewDelegateFlowLayout, UserProfileHeaderDelegate {

    var user: User?
    // var userToLoadFromSearchVC: User?
    var posts = [Post]()
    var currentKey: String?

    override func viewDidLoad(){
        super.viewDidLoad()

        self.collectionView!.register(UICollectionViewCell.self, forCellWithReuseIdentifier: reuseIdentifier)
        self.collectionView!.register(UserProfileHeader.self, forSupplementaryViewOfKind: UICollectionElementKindSectionHeader,
            withReuseIdentifier: headerIdentifier)
        self.collectionView?.backgroundColor = .white

        // if userToLoadFromSearchVC == nil {
            if self.user == nil{
                fetchCurrentUserData()
                //navigationItem.title = userToLoadFromSearchVC.username
            }

    }

    //MARK: - UICollectionView
    override func numberOfSections() -> Int {
        return 1
    }


    override func numberOfItemsInSection -> Int {


    }

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, referenceSizeForHeaderInSection
        section: Int) -> CGSize{
        return CGSize(width: view.frame.width, height: 200)
    }

    override func collectionView(_ collectionView: UICollectionView, viewForSupplementaryElementOfKind kind: String,
        at indexPath: IndexPath) -> UICollectionReusableView{
        let header = collectionView.dequeueReusableSupplementaryView(ofKind: kind, withReuseIdentifier:
            headerIdentifier, for: indexPath) as! UserProfileHeader
        header.delegate = self
```

```
// if let user = self.user{
//     header.user = user
// } else if let userToLoadFromSearchVC = self.userToLoadFromSearchVC{
//     header.user = userToLoadFromSearchVC
// }
    header.user = self.user
    navigationItem.title = user?.username

    return header
}


override func cellforItemAt () -> UICollectionViewCell {
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: reuseIdentifier, for: indexPath)
    return cell
}

//MARK:- API

func fetchCurrentUserData(){
    guard let currentUid = Auth.auth().currentUser?.uid else {return}
    print("current user id is \(currentUid)")

    Database.database().reference().child("users").child(currentUid).observeSingleEvent(of: .value){
        (snapshot) in
        guard let dictionary = snapshot.value as? Dictionary<String, AnyObject> else {return}
        let uid = snapshot.key
        let user = User(uid: uid, dictionary: dictionary)
        self.user = user
        self.navigationItem.title = user.username
        self.collectionView?.reloadData()

    }
}


//MARK:- UserProfileHeader
func handleEditFollowTapped(for header: UserProfileHeader){
    guard let user = header.user else {return}
    if header.editProfileFollowButton.titleLabel?.text == "Edit Profile"{
        let editProfileController = EditProfileController()
        editProfileController.user = user
        editProfileController.userProfileController = self
        let navigationController = UINavigationController(rootViewController: editProfileController)
        present(navigationController, animated: true, completion: nil)

    } else {
        if header.editProfileFollowButton.titleLabel?.text == "Follow"{
```

```
            header.editProfileFollowButton.setTitle("Following", for:.normal)
            user.follow()
        }else{
            header.editProfileFollowButton.setTitle("Follow", for:.normal)
            user.unfollow()
        }
    }
}
```