Project Report

Art Corner

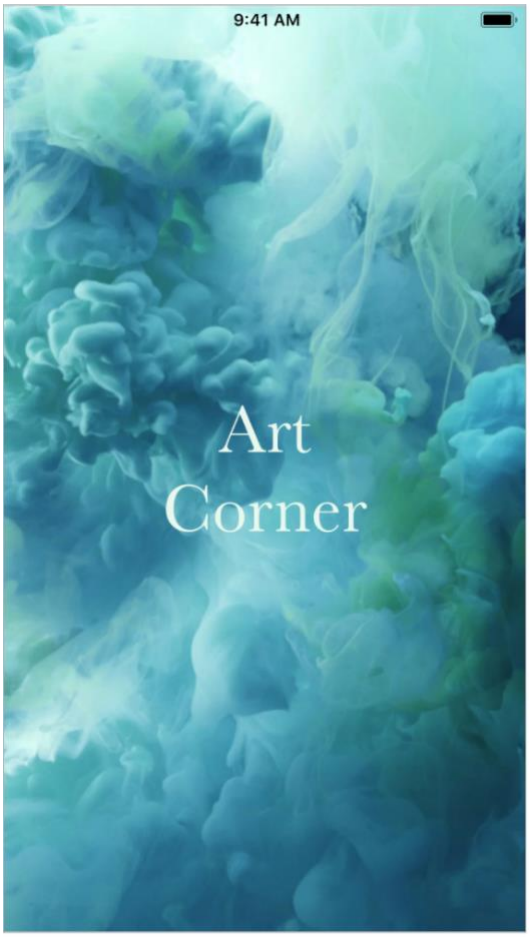
-Suruchi Singh
SUID: 755238759
May 6, 2018

Table of Contents

Summary

App made programmatically, i.e., without storyboard

- Social Media app for people who love art.
- The user can sign up to Register and after they have Logged in they can access the app.
- The users can post the art that they have seen or possess and the information related to it so others can see what they art they like or possess on their feed.
- Other users can like/favorite it and give their reviews on it as well.
- This can help art lovers find about their favorite art from their friends and also have an option to review it.
- The users can follow/unfollow other users.
- You can search the other users that you want to follow with a search bar.
- The user profile will show the location of the user, using a Map.
- The user can search for a painting they would like to know more about and they can see an image and information pulled up from Wikipedia
- Post, Review and User information stored in Firebase



List of Technical Features:

- Fetch photos from Photo Library framework
    - Save all the photos in the form of assets into a data model: `var assets = [PHAsset]()`
    - Use the data model to populate the collection view
    - By default, the first image will be shown on the header or the selected image from collection view

- UIMap Kit
    - Display user's current location by default
    - Search for a place to visit
- Wikipedia API
    - Search for any painting or artist or anything
    - Displays image if present and information about the searched element
- Search Functionality
    - Search for users using the app to see what they have posted by following them
    - Other users posts are available on your home feed once you follow them
- UICollectionView for the home feed, login guide, search and add post functionalities
- Firebase for sign up and login features
- Firebase to store post, review, like/unlike, follow/unfollow, user profile and painting images
- **Contains 4 Models**
    - Page
    - Paining
    - User
    - Review

List of Functional Features:

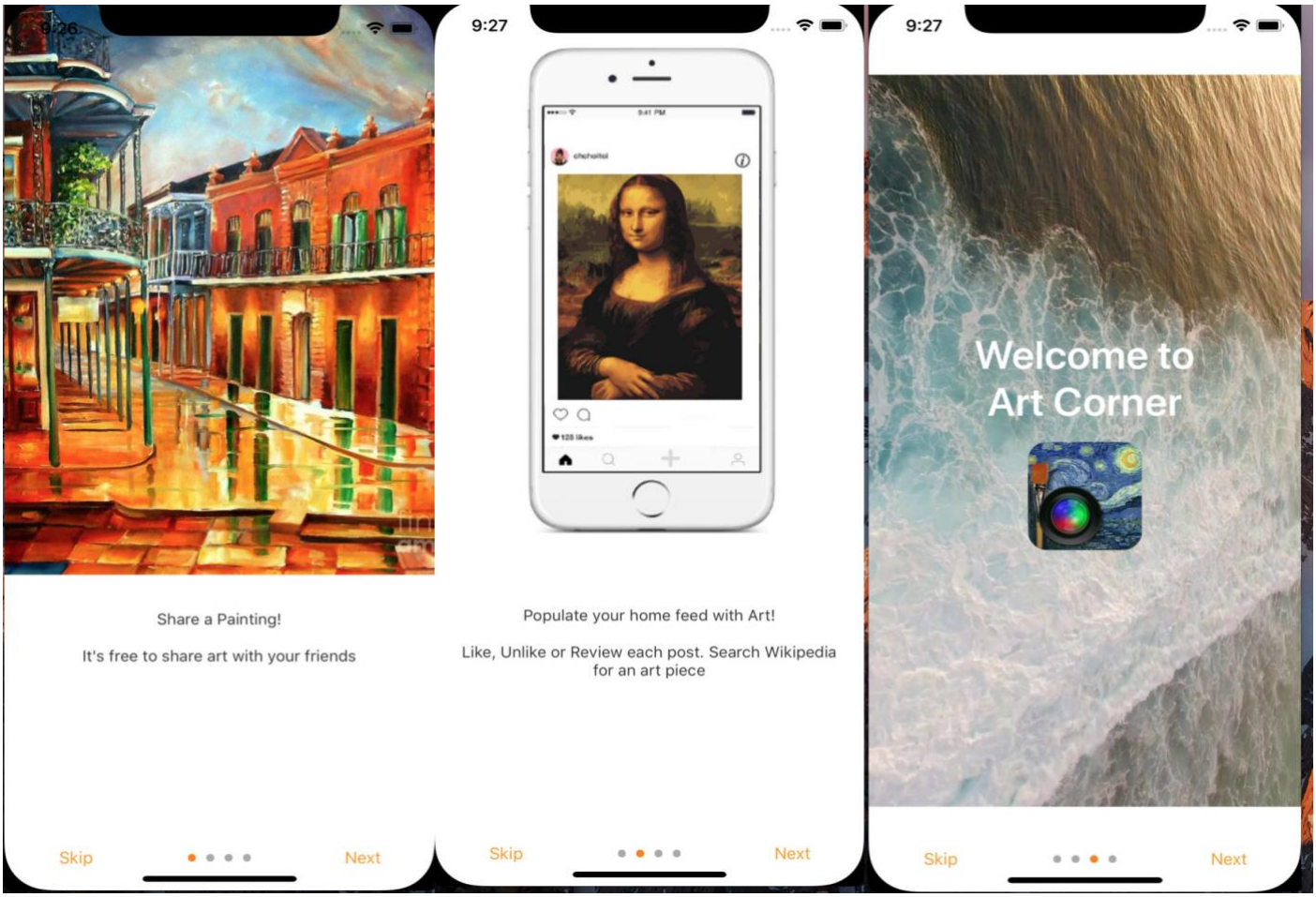
- Walk through or Login Guide
    - Contains background video
    - Animation to take the welcome label and image out of view
- User Sign up and Log in using Firebase
- Create a User profile – profile photo, user profile name and email id
    - The buttons enable and change color or become un-faded once all the text fields are filled
- Upload art pictures through photo library
- Give a title to the art
- Home Feed to see what your friends have shared
- Search for other users
- Review the shared art pieces on the home feed
- Favorite or Like the art pieces posted by others
- Follow or Un Follow users on the app
- Display information of art pieces from Wikipedia
- See your location on the map
- Search for other locations on the map
- See the others user's location on the map

Login Guide

- Collection View Controller - 2 types of cells, Page Cell for image and text field. Last Page cell to handle animation, shown on the next page
- Page Control, Text Field, Image, Skip and Next Button

Page Model:

```swift
//
//  Page.swift
//  Project
//
//  Created by Suruchi Singh on 4/1/18.
//  Copyright © 2018 Suruchi Singh. All rights reserved.
//

import Foundation

struct Page{

    let title : String
    let message : String
    let imageName : String
}

```



Page Control and transitioning of collection View cells:

```
let cellId = "cellId"
let lastPageCellId = "signUpCellViewId"
let loginCellId = "loginCellId"

let pages: [Page] = {

    let firstPage = Page(title: "Share a Painting!", message: "It's free to share art with your friends", imageName: "FirstScreen")
    let secondPage = Page(title: "Populate your home feed with Art!", message: "Like, Unlike or Review each post. Search Wikipedia for an art piece", imageName:
        "ThirdScreen")

    let thirdPage = Page(title: "Send from your Library", message: "Tap the Add menu to share a painting from your Photo Library", imageName: "SecondScreen1")

    return [firstPage, secondPage, thirdPage]

}()

lazy var pageControl: UIPageControl = {

    let pc = UIPageControl()
    pc.pageIndicatorTintColor = .lightGray
    pc.numberOfPages = self.pages.count + 1
    pc.currentPageIndicatorTintColor = .orange
    return pc
}()


override func willTransition(to newCollection: UITraitCollection, with coordinator: UIViewControllerTransitionCoordinator) {

    print(UIDevice.current.orientation.isLandscape)
    collectionView.collectionViewLayout.invalidateLayout()
    let indexPath = IndexPath(item: pageControl.currentPage, section: 0)

    DispatchQueue.main.async {
        self.collectionView.scrollToItem(at: indexPath, at: .centeredHorizontally, animated: true)
        self.collectionView.reloadData()
    }

}

//fileprivate func pagingControls() {
fileprivate func pagingControls(){
    let bottomStackView = UIStackView(arrangedSubviews: [skipButton, pageControl, nextButton])
```

Registration of login cell and the last page cell that contains Animation:

```
fileprivate func registerCells(){

    collectionView.register(PageCell.self, forCellWithReuseIdentifier: cellId)
    collectionView.register(LasPageCell.self, forCellWithReuseIdentifier: lastPageCellId)
}

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    return pages.count + 1
}

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

    if indexPath.item == pages.count{

        let signUpCellView = collectionView.dequeueReusableCell(withReuseIdentifier: lastPageCellId, for: indexPath) as! LasPageCell
        signUpCellView.delegate = self
        return signUpCellView

    }
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "cellId", for: indexPath) as! PageCell
    //cell.backgroundColor = .white
    let page = pages[indexPath.item]
    cell.page = page

    return cell
}
```

Video Background Feature

- Last Page Cell – Contains Video Background and an animation feature that shifts the label and image to the left and then in an upward animation out of the view

Code Snippet for Video Background:

Import AVFoundation –

Video Background on the last page cell and animation on Tap gesture

```
    override init(frame: CGRect) {
        super.init(frame: frame)

        // Start with a generic UIView and add it to the ViewController view
        let myPlayerView = UIView(frame: self.bounds)

        myPlayerView.backgroundColor = UIColor.white
        addSubview(myPlayerView)

        // Use a local or remote URL
        let filePath = Bundle.main.path(forResource: "video1", ofType: "mov")
        let url = URL(fileURLWithPath: filePath!)

        // Make a player
        let myPlayer = AVPlayer(url: url)
        myPlayer.actionAtItemEnd = .none
        myPlayer.isMuted = true
        myPlayer.play()

        // Make the AVPlayerLayer and add it to myPlayerView's layer
        let avLayer = AVPlayerLayer(player: myPlayer)
        avLayer.frame = myPlayerView.frame

        //avLayer.frame = uiView.bounds

        //Loop the video
        NotificationCenter.default.addObserver(forName: .AVPlayerItemDidPlayToEndTime, object: myPlayer.currentItem, queue: nil) { (_) in
            myPlayer.seek(to: kCMTimeZero)
            myPlayer.play()
        }
        myPlayerView.layer.addSublayer(avLayer)

        let stackView = UIStackView(arrangedSubviews: [titleLabel,iconAppImage])

        myPlayerView.addSubview(stackView)

        stackView.axis = .vertical
        stackView.spacing = 15
        addSubview(stackView)

        stackView.translatesAutoresizingMaskIntoConstraints = false

        stackView.centerXAnchor.constraint(equalTo: centerXAnchor).isActive = true
        stackView.centerYAnchor.constraint(equalTo: centerYAnchor).isActive = true
        //stackView.widthAnchor.constraint(equalTo: widthAnchor,  constant: -100).isActive = true
        stackView.heightAnchor.constraint(equalToConstant: 200).isActive = true
        stackView.widthAnchor.constraint(equalToConstant: 250).isActive = true

        myPlayerView.addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(handleTapAnimation)))
        addGestureRecognizer(UITapGestureRecognizer(target: self, action: #selector(handleTapAnimation)))
    }
```

Animation Feature

First Animation to shift the label and image to the left and the second animation to shift it up and out of the view Both done one after another on Tap Gesture

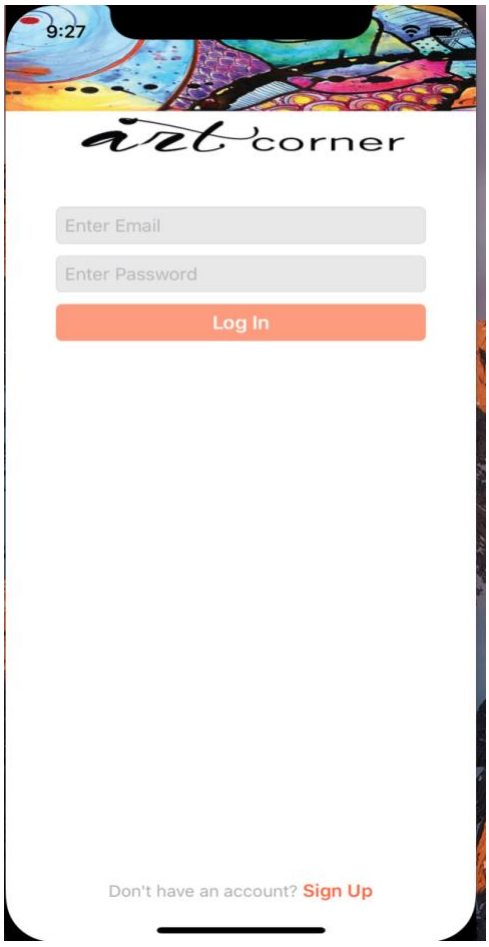
Tap Animation Code Snippet:

```
@objc func handleTapAnimation(){

    UIView.animate(withDuration: 0.5, delay: 0.5, usingSpringWithDamping: 0.5, initialSpringVelocity: 0.5, options: .curveEaseOut, animations: {

        self.titleLabel.transform = CGAffineTransform(translationX: -30, y: 0)

    }) { (_) in

        UIView.animate(withDuration: 0.5, delay: 0.5, usingSpringWithDamping: 1, initialSpringVelocity: 1, options: .curveEaseOut, animations: {

            self.titleLabel.transform = self.titleLabel.transform.translatedBy(x: 0, y: -400)
        })//, completion: ((Bool) -> Void)?)
    }

    UIView.animate(withDuration: 0.5, delay: 0.5, usingSpringWithDamping: 0.5, initialSpringVelocity: 0.5, options: .curveEaseOut, animations: {

        self.iconAppImage.transform = CGAffineTransform(translationX: -30, y: 0)

    }) { (_) in

        UIView.animate(withDuration: 1, delay: 0.5, usingSpringWithDamping: 1, initialSpringVelocity: 1, options: .curveEaseOut, animations: {

            self.iconAppImage.transform = self.iconAppImage.transform.translatedBy(x: 0, y: -800)
        }){ (ok) in
            print("Ended \(ok)")
             self.startSignUp()

        }

    }

}
```

Login Cell

- Login in using Firebase - email and password

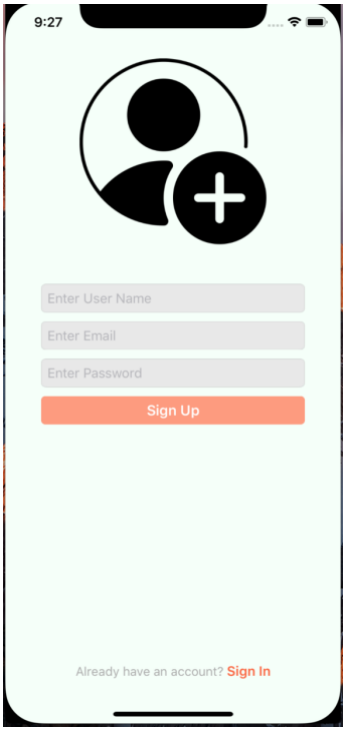
```
@objc func handleUserLogIn(){

    print("logging in")
    guard let email = emailTextField.text else {return}
    guard let password = passwordTextField.text else {return}

    Auth.auth().signIn(withEmail: email, password: password) { (user, error) in

      if let error = error{
          print("Log in failed",error)
        }

      print("Successfully signed in!", user?.uid ?? "")

      guard let mainTabBarController = UIApplication.shared.keyWindow?.rootViewController as? TabBarController else {return}
      mainTabBarController.setUpViewControllers()

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


 @objc func presentSignOut(){

     let signUpController = SignUpViewController()
     //print(navigationController)
     navigationController?.pushViewController(signUpController, animated: true)

}
```



Sign Up Controller

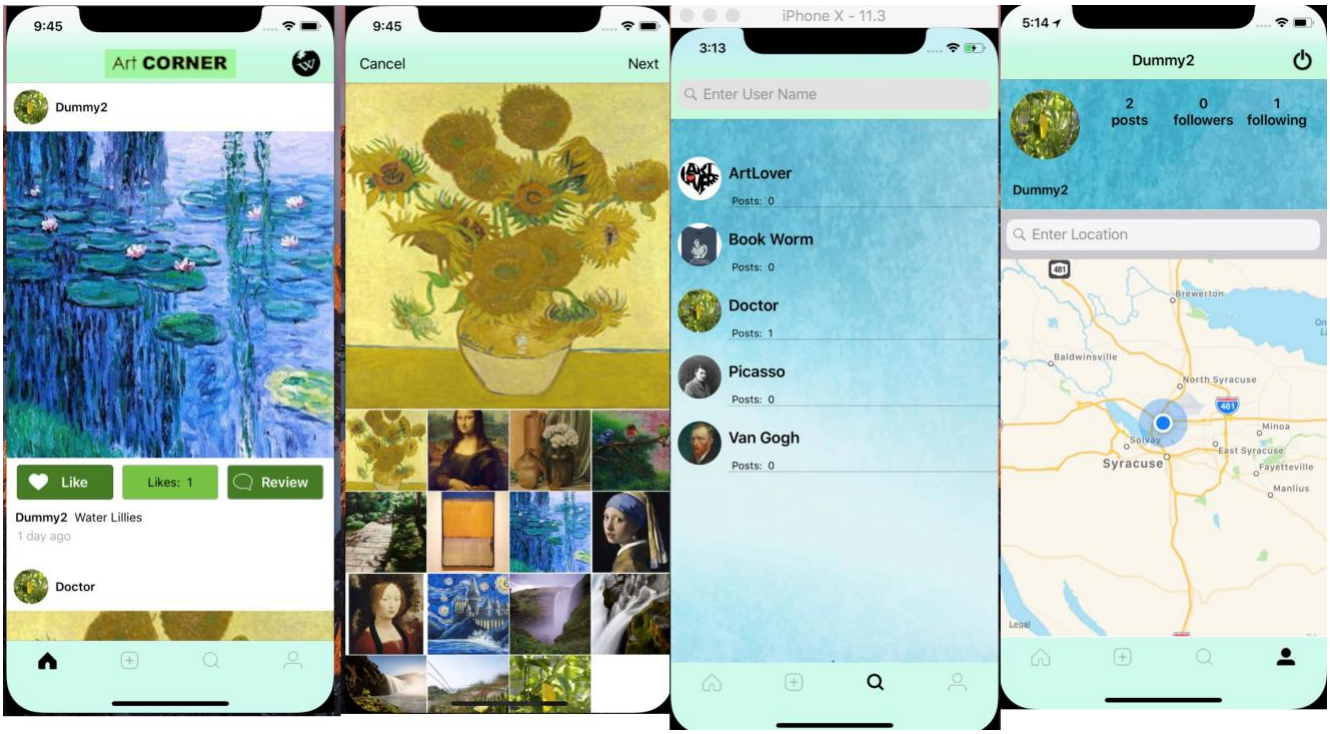
- Sign Up using Firebase using email, user name, password and profile image

```
@objc func handleRegistration(){

    guard let email = emailTextField.text, email.count > 0 else {return}
    guard let password = passwordTextField.text, password.count > 0 else {return}
    guard let userName = userNameTextField.text, userName.count > 0 else {return}
    Auth.auth().createUser(withEmail: email, password: password) { (user, error) in
        if let error = error{
            print("Failed to create user", error)
            return
        }
        print("Successfully created user", user?.uid ?? "")
        guard let image = self.addPhotoButton.imageView?.image else {return}
        guard let uploadData = UIImageJPEGRepresentation(image, 0.1) else {return}
        let imageFileName = NSUUID().uuidString
        Storage.storage().reference().child("profile-images").child(imageFileName).putData(uploadData, metadata: nil, completion: { (metadata, error) in

            if let error = error{
                print("Couldn't upload profile image to Firebase", error)
                return
            }
            //guard let profileImageURL = metadata?.downloadURL()?.absoluteString  else {return}

            Storage.storage().reference().child("profile-images").child(imageFileName).downloadURL(completion: { (url, error) in
                if error != nil {
                    print(error as Any)
                } else {
                    guard let imageUrl = url?.absoluteString else { return }
                    self.profilePicUrl = imageUrl
                    print("Successful saving profile image to Firebase: - ", self.profilePicUrl!)
                    //completion(imageUrl)
                }

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

                let userNameValues = ["username": userName, "profileImageURL": self.profilePicUrl]
                let values = [uid: userNameValues]
                Database.database().reference().child("users").updateChildValues(values, withCompletionBlock: { (error, reference) in

                    if let error = error{
                        print("Couldn't save user sign up details to Firebase", error)
                    }
                    print("Successful in saving name and profile pic to Firebase")
                    guard let mainTabBarController = UIApplication.shared.keyWindow?.rootViewController as? TabBarController else {return}
                    mainTabBarController.setUpViewControllers()

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



Tab Bar Controller

- Home Feed
- Add Painting
- Search Users
- User Profile

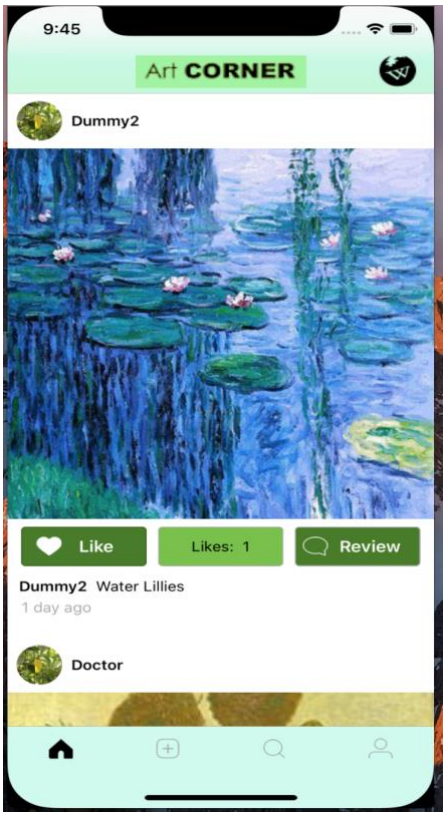



```
    func setUpViewControllers(){

        //Home Feed
        let homeController = templateNavController(unselectedImage: , selectedImage: , rootViewController: HomeController(collectionViewLayout: UICollectionViewFlowLayout()))

        //Search

        let searchNavController = templateNavController(unselectedImage: , selectedImage: , rootViewController: UserSearchController(collectionViewLayout: UICollectionViewFlowLayout()))


        //Add
        let addPostController = templateNavController(unselectedImage: , selectedImage: )

        let userProfileNavController = templateNavController(unselectedImage: , selectedImage: , rootViewController: UserProfileController())
        tabBar.tintColor = .black

        viewControllers = [homeController,addPostController,searchNavController,userProfileNavController]

        guard let items = tabBar.items else {return}
        for item in items{

            item.imageInsets = UIEdgeInsets(top: 4, left: 0, bottom: -4, right: 0)
        }
    }

    fileprivate func templateNavController(unselectedImage: UIImage, selectedImage: UIImage, rootViewController: UIViewController = UIViewController())-> UINavigationController{

        let viewController = rootViewController
        let navController = UINavigationController(rootViewController: viewController)

        navController.tabBarItem.image =  unselectedImage
        navController.tabBarItem.selectedImage =  selectedImage
        return navController
    }
}
```

Home Feed Controller

- UICollectionVIew
- Each cell has
    - Painting image
    - Painting Title
    - User Name and User Profile Image
    - Information Button
    - Like Button
    - Review Button
    - Date of posting the painting

Painting/Post Model:




```swift
import UIKit

struct Painting {

    var id: String?
    //var likeCount: Int
    let user: User
    let paintingUrl: String
    let paintingTitle: String
    let creationDate: Date

    var hasLiked: Bool = false

    init(user: User, dictionary: [String: Any]) {

        self.user = user
        self.paintingUrl = dictionary["paintingUrl"] as? String ?? ""
        self.paintingTitle = dictionary["paintingTitle"] as? String ?? ""
        //self.likeCount = dictionary["likeCount"] as? Int ?? 0

        let seconds = dictionary["creationDate"] as? Double ?? 0
        self.creationDate = Date(timeIntervalSince1970: seconds)
    }
}
```

Notification Observer to handle a new Post

- Handle Notification for a new post using NotificaitonCenter.default.addObserver and Refreshing the collectionView using UI RefreshControl
- Refresh control gives a scroll view animation to reload collection view so new posts can be fetched from firebase

```
class HomeController: UICollectionViewController, UICollectionViewDelegateFlowLayout, HomePostCellDelegate {
    let cellId = "cellId"
    var posts = [Painting]()
    let refreshControl = UIRefreshControl()
    let name = NSNotification.Name(rawValue: "UpdateHome")

    override func viewDidLoad() {
        super.viewDidLoad()
        tabBarController?.tabBar.isHidden = false
        self.tabBarController?.tabBar.layer.zPosition = 0

        NotificationCenter.default.addObserver(self, selector: #selector(handleNewPost), name: name, object: nil)

        collectionView?.backgroundColor = .white
        navigationController?.navigationBar.backgroundColor =
        collectionView?.register(HomePostCell.self, forCellWithReuseIdentifier: cellId)

        refreshControl.addTarget(self, action: #selector(handleRefresh), for: .valueChanged)
        collectionView?.refreshControl = refreshControl

        setUpNavigationItems()

        fetchAllPosts()

    }

    @objc func handleNewPost(){

        handleRefresh()
    }

    @objc func handleRefresh(){

        posts.removeAll()
        fetchAllPosts()
    }

    fileprivate func fetchAllPosts(){

        fetchPaintings()
        fetchFollowingUsers()
    }
```

- fetchFollowingUsers() fetches the posts by all the users that you are following from Firebase
- fetchPaintings() fetches all paintings and their corresponding users and stores them in descending order of time

Sort the incoming posts:

```
self.posts.append(post)

self.posts.sort(by: { (post1, post2) -> Bool in

    return post1.creationDate.compare(post2.creationDate) == .orderedDescending
})

self.collectionView?.reloadData()
```

Like/Unlike Post

Like Feature:

```
func didLike(for cell: HomePostCell){

    print("Home controller did like")
    guard let indexPath  = collectionView?.indexPath(for: cell) else {return}

    var post = self.posts[indexPath.item]

    guard let postId = post.id else {return}
    //let userId = post.user.uid

    guard let uid = Auth.auth().currentUser?.uid else {  return }
    let values = [uid: post.hasLiked == true ? 0 : 1]
    Database.database().reference().child("likes").child(postId).updateChildValues(values) { (error, _) in

        if let error = error{

            print("Couldnt save Likes into Db", error )
            return
        }
        print("Successfully liked post")
        post.hasLiked = !post.hasLiked
        self.posts[indexPath.item] = post
        //self.collectionView?.reloadItems(at: [indexPath])
        cell.setLikeCount()
    }
}
```

UnLike Feature:

```
    func didUnLike(for cell: HomePostCell){

        print("Home controller did Un like")
        guard let indexPath  = collectionView?.indexPath(for: cell) else {return}

        let post = self.posts[indexPath.item]

        guard let postId = post.id else {return}

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

        Database.database().reference().child("likes").child(postId).child(uid).removeValue() { (error1, ref) in
            if error1 != nil{
                print("Failed to delete movie", error1!)
                return

            }
            print("Successfully Un liked post")

            cell.setLikeCount()
        }
    }
}
```

Update label according to weather user like or disliked the post and fetch the number from firebase.

Get Time and Date of Posting each Post

```
//Date Modification
extension Date {
    func timeAgoDisplay() -> String {
        let secondsAgo = Int(Date().timeIntervalSince(self))

        let minute = 60
        let hour = 60 * minute
        let day = 24 * hour
        let week = 7 * day
        let month = 4 * week

        let quotient: Int
        let unit: String
        if secondsAgo < minute {
            quotient = secondsAgo
            unit = "second"
        } else if secondsAgo < hour {
            quotient = secondsAgo / minute
            unit = "min"
        } else if secondsAgo < day {
            quotient = secondsAgo / hour
            unit = "hour"
        } else if secondsAgo < week {
            quotient = secondsAgo / day
            unit = "day"
        } else if secondsAgo < month {
            quotient = secondsAgo / week
            unit = "week"
        } else {
            quotient = secondsAgo / month
            unit = "month"
        }

        return "\(quotient) \(unit)\(quotient == 1 ? "" : "s") ago"

    }
}
```

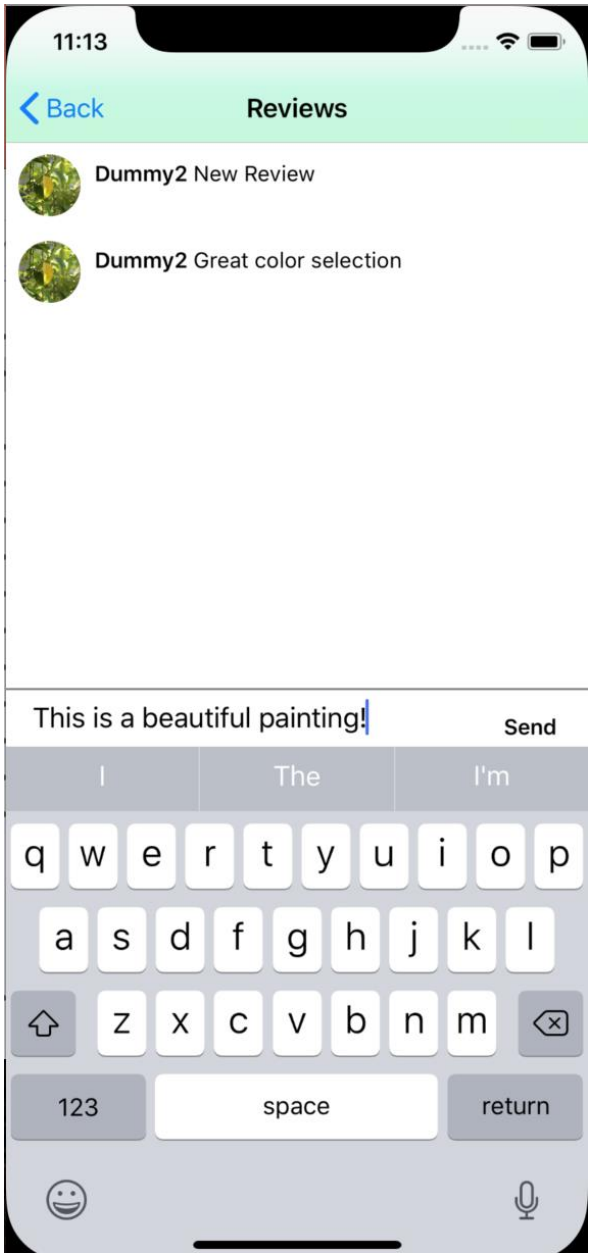
```
fileprivate func setPaintingName(){

    guard let post = post else { return }

    let attributedText = NSMutableAttributedString(string: post.user.username, attributes: [NSAttributedStringKey.font : UIFont.b
    attributedText.append(NSAttributedString(string: "  \(post.paintingTitle)", attributes: [NSAttributedStringKey.font: UIFont.s
    attributedText.append(NSAttributedString(string: "\n\n  ", attributes: [NSAttributedStringKey.font: UIFont.systemFont(ofSize:
    let timeDisplay = post.creationDate.timeAgoDisplay()
    attributedText.append(NSAttributedString(string: timeDisplay, attributes: [NSAttributedStringKey.font: UIFont.systemFont(ofSi
    paintingTitle.attributedText = attributedText
}
```

Fetch Painting/Post

```
fileprivate func fetchPaintings(){

    //print("fetching Paitings!")
    guard let uid = Auth.auth().currentUser?.uid else { return }


    Database.fetchUserUID(uid: uid, completion: { (user) in

        //print("completion block")
        self.fetchPostWithUser(user: user)
    })

}

func fetchPostWithUser(user: User){

    let uid = user.uid

    let ref = Database.database().reference().child("painting").child(uid)
    ref.observeSingleEvent(of: .value, with: { (snapshot) in

        self.collectionView?.refreshControl?.endRefreshing()
        guard let dicionaries = snapshot.value as? [String: Any] else { return }
        dicionaries.forEach({ (key, value) in

            guard let dictionary = value as? [String: Any] else { return }
            //print(dictionary)
            var post = Painting(user: user, dictionary: dictionary)
            post.id = key

            guard let currentUserId = Auth.auth().currentUser?.uid else {return}

            let ref = Database.database().reference().child("likes").child(key)
            ref.child(currentUserId ).observeSingleEvent(of: .value, with: { (snapshot) in


                if let value = snapshot.value as? Int, value == 1{
                    post.hasLiked = true

                } else{

                    post.hasLiked = false

                }

                self.posts.append(post)
```

Review each painting/post

- UiCollectionView
  - User Name
  - User Profile Picture
  - Review
- Accessory Input View
  - Text View
  - Send Button



Review Model:

```
import Foundation
import UIKit

struct Review {

    let user: User?

    let text: String?
    let uid: String?

    init(user: User, dictionary: [String: Any]){

        self.user = user
        self.text = dictionary["text"] as? String ?? ""
        self.uid = dictionary["userId"] as? String ?? ""

    }
}
```

Adjust Size for item in Review Cell according to the size of review text

```
//Custom size of each review cell
func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath:

    let frame = CGRect(x: 0, y: 0, width: view.frame.width, height: 50)
    let dummyCell = ReviewCell(frame: frame)
    dummyCell.review = reviewsArray[indexPath.item]
    dummyCell.layoutIfNeeded()
    let targetSize = CGSize(width: view.frame.width, height: 100)
    let estimatedSize = dummyCell.systemLayoutSizeFitting(targetSize)
    let height = max(40 + 8 + 8, estimatedSize.height)
    return CGSize(width: view.frame.width, height: height)
}
```

Input Accessory View:

```
    lazy var containerView: ReviewInputAccessoryView = {

        let frame = CGRect(x: 0, y: 0, width: view.frame.width, height: 50)
        let reviewView = ReviewInputAccessoryView(frame: frame)
        reviewView.delegate = self
        return reviewView

    }()

    override var inputAccessoryView: UIView?{
        get{  return containerView  }
    }

    override var canBecomeFirstResponder: Bool{
        return true
    }
```

Send Reviews to Firebase:

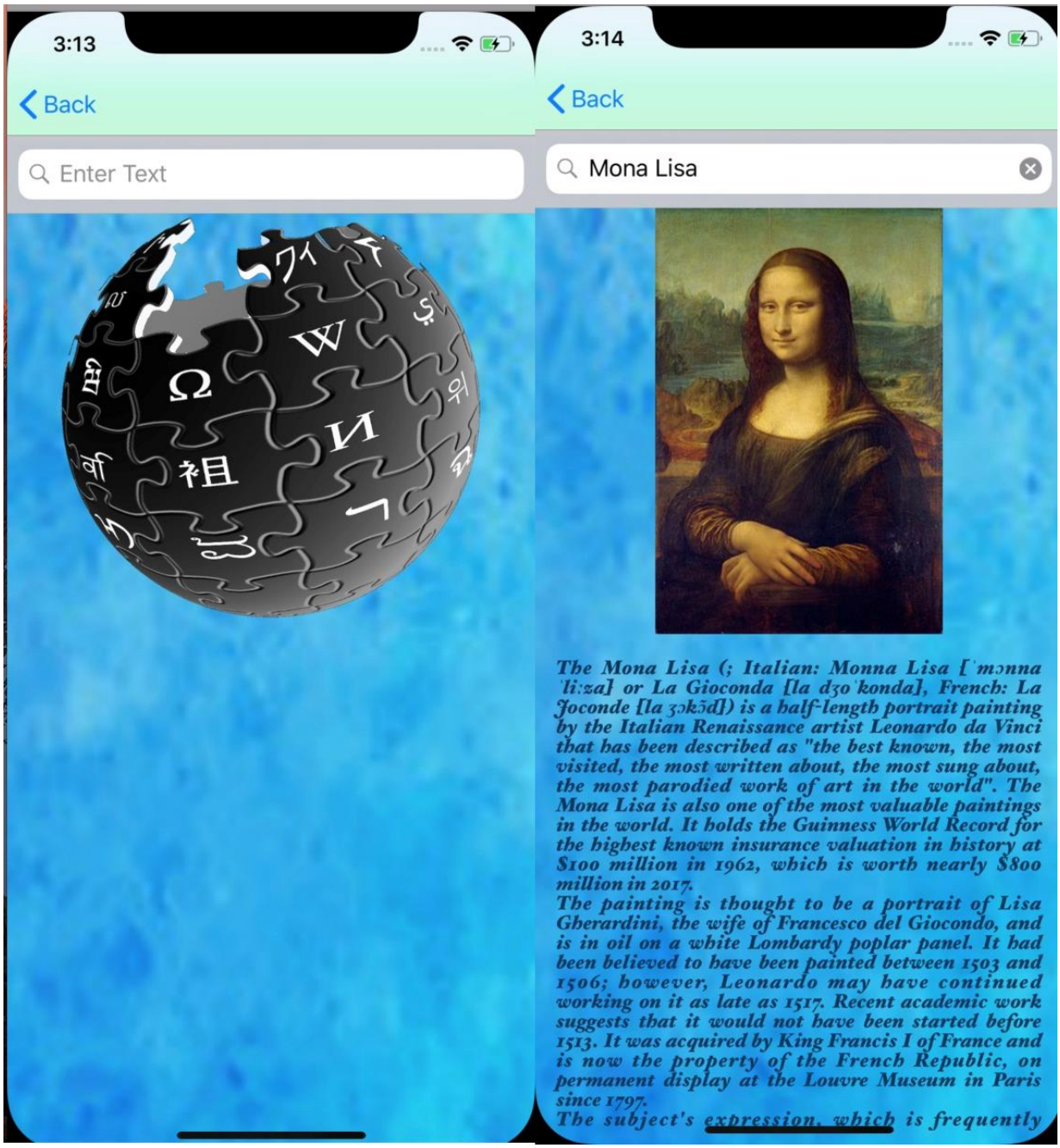
```
//Send review to Firebase
func didSendReview(for review: String){

    print("insert review into firebase")

    print("Send a Review!", review)

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

    let postId = self.post?.id ?? ""
    let values = ["text": review, "creationDate": Date().timeIntervalSince1970, "userId": uid] as [String : Any]

    Database.database().reference().child("reviews").child(postId).childByAutoId().updateChildValues(values) { (error, reference) in
        if let error = error{
            print("Failure to update Firebase",error)
            return

        }
        print("Succesfully sent child ids to firebase")
        self.containerView.clearReviewTextField()
    }
}
```

Fetch Reviews from Firebase:

```
var reviewsArray = [Review]()

//fetch all reviews from Firebase
fileprivate func fetchReviews(){

    guard let postId = self.post?.id else { return }

    let ref = Database.database().reference().child("reviews").child(postId)
    ref.observe(.childAdded, with: { (snapshot) in

        guard let dictionary = snapshot.value as? [String: Any] else { return}

        guard let uid = dictionary["userId"] as? String else { return}

        Database.fetchUserUID(uid: uid, completion: { (user) in

            let review = Review(user: user, dictionary: dictionary)
            //review.user = user
            print(review.text as Any, review.uid as Any)
            self.reviewsArray.append(review)
            self.collectionView?.reloadData()

        })


    }) { (error) in

        print("Failed to observe reviews", error)
    }
}
```

Wikipedia Search – External API

- UIView
  - Search Bar
  - Image View
  - Text Field
- Wikipedia url
- Import SwiftyJSON , SDWebImage and Alamofire
- SDWebImage will convert the web image into a UIImage
- SwiftyJSON and Alamofire will help parse the information from the Wikipedia link



Code Snippet:

```
        let pName = searchText
        requestInfo(pName: pName)
    }

}

func requestInfo(pName: String){

    let parameters : [String:String] = [

        "format" : "json",
        "action" : "query",
        "prop"   : "extracts|pageimages",
        "exintro": "",
        "explaintext" : "",
        "titles" : pName,
        //"titles" : "Mona Lisa",
        "indexpageids" : "",
        "redirects" : "1",
        "pithumbsize" : "400"
    ]
    Alamofire.request(wikipediaURL, method: .get, parameters: parameters).responseJSON { (response) in
        if response.result.isSuccess{
            print("Got the Wikipedia")
            print(response)

            let infoJSON: JSON = JSON(response.result.value!)
            let pageID = infoJSON["query"]["pageids"][0].stringValue
            let artDescription = infoJSON["query"]["pages"][pageID]["extract"].stringValue

            let artImageURL = infoJSON["query"]["pages"][pageID]["thumbnail"]["source"].stringValue
            self.imageView.sd_setImage(with: URL(string: artImageURL))
            //self.paintingImage.sd_setImage(with: URL(string: artImageURL))
            print(artDescription)
            self.textView.text = artDescription

        }
    }
}
```

Add Painting/Post Controller

- UICollectionView
- PHAssetCollection – Getting photos from Photos.framework
- Collection View Header will contain the selected image and by default the first image fetched from the photo library
- Cells below the header will be populated by fetching all the photos from photo library in the iPhone

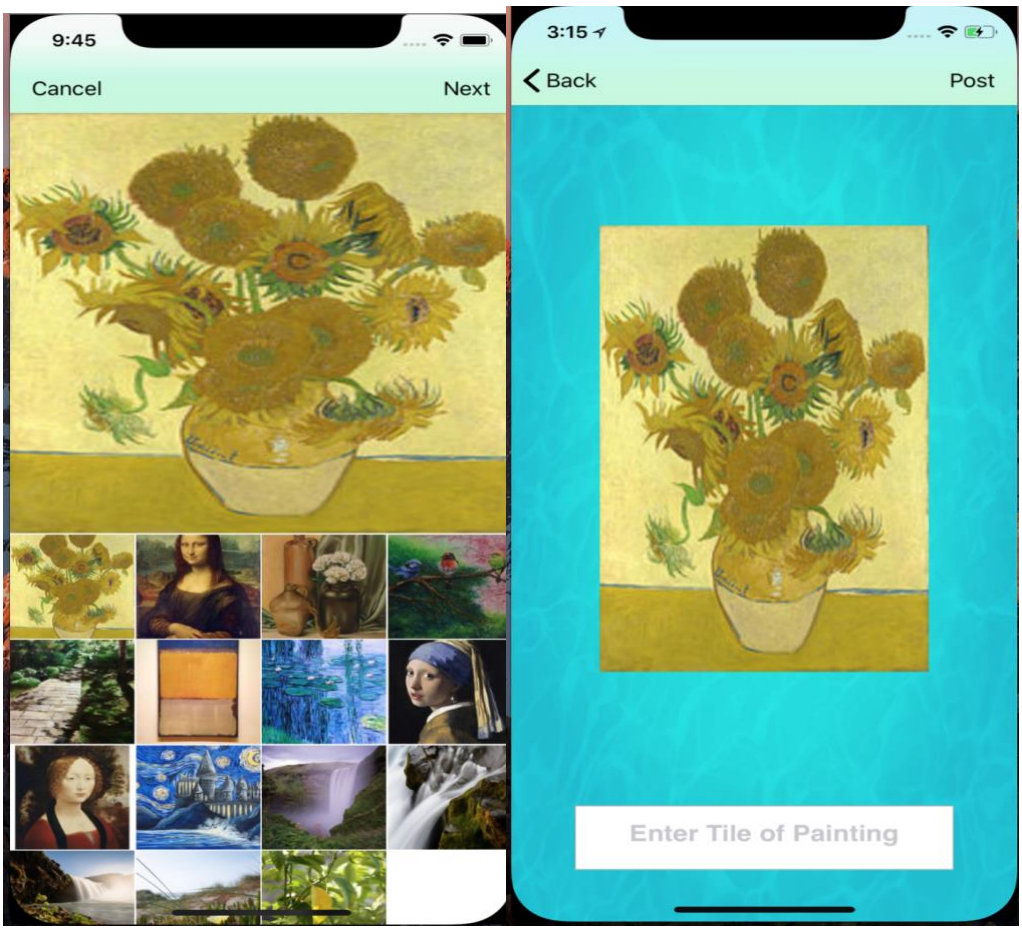
```
    var images = [UIImage]()
var selectedImage: UIImage?
var assets = [PHAsset]()
```

Fetch Photos from Photo library framework:

```
//Fetch Photos from Library
fileprivate func fetchPhotos(){

    print("Fetching Photos from Framework")
    let fetchOptions = PHFetchOptions()
    fetchOptions.fetchLimit = 15 //number of photos to fetch

    let sortDescriptors = NSSortDescriptor(key: "creationDate", ascending: false)
    fetchOptions.sortDescriptors = [sortDescriptors]


    let allPhotos = PHAsset.fetchAssets(with: .image, options: fetchOptions)

    DispatchQueue.global(qos: .background).async {

        allPhotos.enumerateObjects { (asset, count, stop) in

            //print(asset)

            let imageManager = PHImageManager.default()
            let targetSize = CGSize(width: 200, height: 200)
            let options = PHImageRequestOptions()
            options.isSynchronous = true

            imageManager.requestImage(for: asset, targetSize: targetSize, contentMode: .aspectFit, options: options, resultHandler: { (image, info) in

                if let image = image{
                    self.images.append(image)
                    self.assets.append(asset)
                    //print(image)

                    if self.selectedImage == nil{
                        self.selectedImage = image //shows the first image
                    }
                }

                if count == allPhotos.count - 1 {

                    DispatchQueue.main.async {
                        self.collectionView?.reloadData()
                    }
                }
            })
        }
    }
}
```

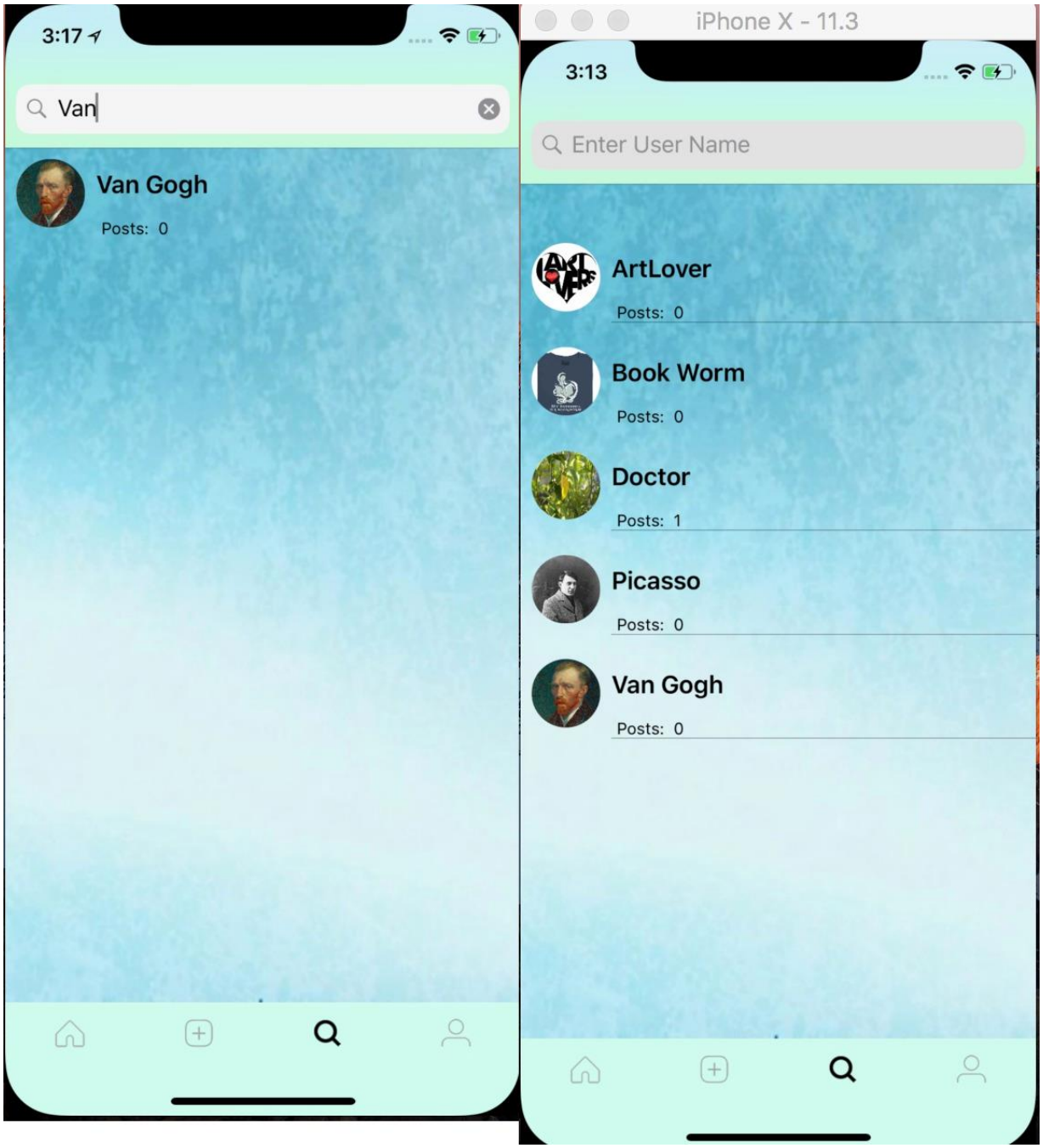
Set first image from photo library or the image selected from the collection view into the header cell

```
//Creating Header Cell
var header: PhotoSelectorHeaderCell?

override func collectionView(_ collectionView: UICollectionView, viewForSupplementaryElementOfKind kind: String, at indexPath: IndexPath) ->
    UICollectionReusableView {

    let header = collectionView.dequeueReusableSupplementaryView(ofKind: kind, withReuseIdentifier: headerId, for: indexPath) as! PhotoSelectorHeaderCell
    //header.backgroundColor = .red
    self.header = header
    header.photoImageView.image = selectedImage

    if let selectedImage = selectedImage{

        if let index = self.images.index(of: selectedImage){
            let selectedAsset = self.assets[index]

            let imageManager = PHImageManager.default()

            let targetSize = CGSize(width: 600, height: 600)
            imageManager.requestImage(for: selectedAsset, targetSize: targetSize, contentMode: .default, options: nil) { (image, info) in

                header.photoImageView.image = image

            }
        }
    }
    return header
}
```



Search Feature

- Search Bar
- UI Collection View
    - User Name
    - User Profile Picture
    - Number of Posts posted
- On tapping each user, it displays that user's profile information



Search Bar Functionality:

```
    func searchBarCancelButtonClicked(_ searchBar: UISearchBar) {

        searchBar.text = ""
        handleRefresh()
        searchBar.setShowsCancelButton(false, animated: true)
        // Remove focus from the search bar.
        searchBar.endEditing(true)
    }

    func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {

        if searchText.isEmpty{
            filteredUsers = users
            //handleRefresh()
        }
        else{
            filteredUsers = self.users.filter({ (users) -> Bool in

                return users.username.lowercased().contains(searchText.lowercased())
            })
        }

        filteredUsers = self.users.filter { (user) -> Bool in

            return user.username.contains(searchText)
        }

        self.collectionView?.reloadData()

    }
```

Refresh Animation:

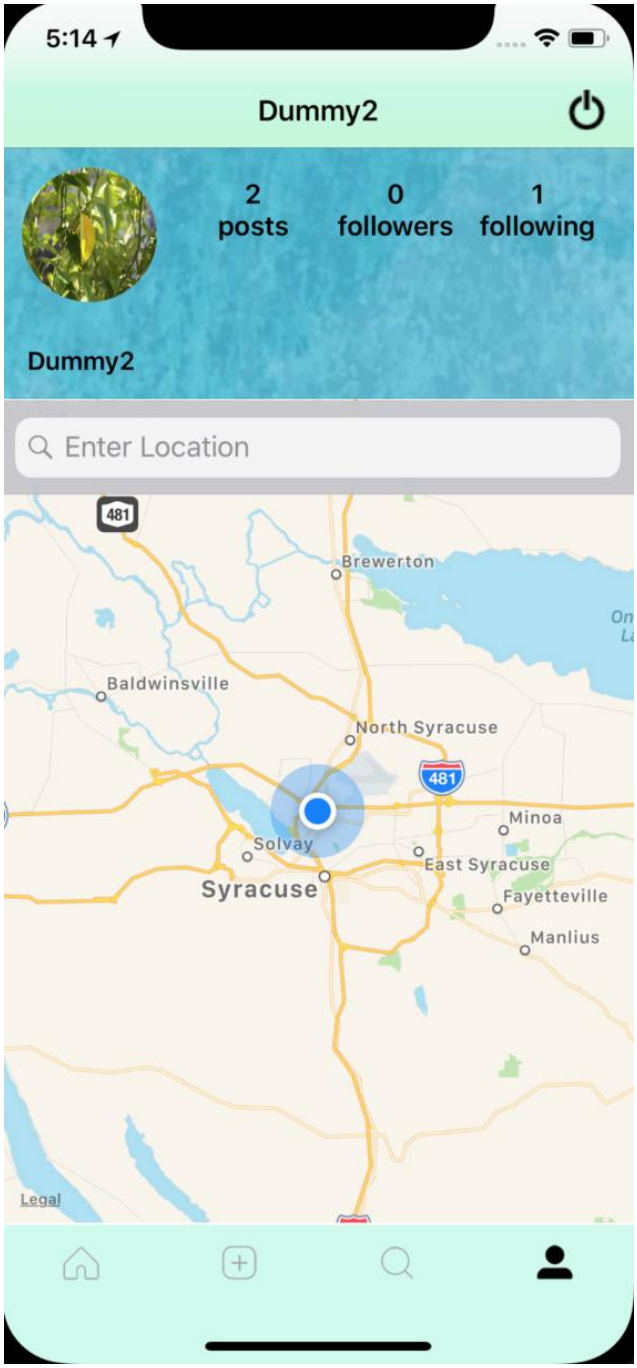
```
    //Refresh Animation
    let refreshControl = UIRefreshControl()
    refreshControl.addTarget(self, action: #selector(handleRefresh), for: .valueChanged)
    collectionView?.refreshControl = refreshControl
```

Handle Refresh Function:

```
lazy var searchBar: UISearchBar = {

    let searchBar = UISearchBar()
    searchBar.placeholder = "Enter User Name"
    searchBar.tintColor = .gray
    UITextField.appearance(whenContainedInInstancesOf: [UISearchBar.self]).backgroundColor =
    searchBar.delegate = self
    searchBar.isHidden = false
    return searchBar
}()

@objc func handleRefresh(){

    users.removeAll()
    filteredUsers.removeAll()
    fetchAllUsers()
}
```

User Profile Information

- UI View Controller
  - User Profile Picture
  - User Name
  - Number of posts posted on the app
  - Number of followers
- Map Kit
  - User ‘s current location when nothing is searched for
  - Displays a particular location when typed for that location in the search bar
- Log Out Bar Button Item



Displays user's current location

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {

    let location = locations[0]
    let center = location.coordinate
    let span  = MKCoordinateSpan(latitudeDelta: 0.5, longitudeDelta: 0.5)
    let region = MKCoordinateRegion(center: center, span: span)
    mapView.setRegion(region, animated: true)
    mapView.showsUserLocation = true

}

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

    locationManager.delegate = self
    locationManager.requestWhenInUseAuthorization()
    locationManager.startUpdatingLocation()

    fetchUser()

    setUpLogOutButton()
```

Search for a specific location

```
let mapView = MKMapView()

 lazy var searchMapBar: UISearchBar = {

     let searchBar = UISearchBar()
     searchBar.placeholder = "Enter Location"
     searchBar.tintColor = .gray
     UITextField.appearance(whenContainedInInstancesOf: [UISearchBar.self]).backgroundColor =
     searchBar.delegate = self

     return searchBar
 }()

 func searchBar(_ searchBar: UISearchBar, textDidChange searchText: String) {

     if searchText.isEmpty{

         locationManager.startUpdatingLocation()
     }
     else{

         let geocoder = CLGeocoder()
         geocoder.geocodeAddressString(searchText) { (placemarks:[CLPlacemark]?, error:Error?) in
             if error == nil {

                 let placemark = placemarks?.first

                 let anno = MKPointAnnotation()
                 anno.coordinate = (placemark?.location?.coordinate)!
                 anno.title = searchText

                 let span = MKCoordinateSpanMake(0.075, 0.075)
                 let region = MKCoordinateRegion(center: anno.coordinate, span: span)

                 self.mapView.setRegion(region, animated: true)
                 self.mapView.addAnnotation(anno)
                 self.mapView.selectAnnotation(anno, animated: true)



             }else{
                 print(error?.localizedDescription ?? "error")
             }
         }
     }
 }
```

User Information

Display number of posts/followers/following

```
func setFollowers(){

    let ref = Database.database().reference().child("followers")
    ref.observeSingleEvent(of: .value, with: { (snapshot) in

        if snapshot.hasChild(self.userId!){

            let newRef = Database.database().reference().child("followers").child(self.userId!)
            newRef.observeSingleEvent(of: .value, with: { (snapshot) in

                let followersCount = Int(snapshot.childrenCount)
                let attributedText = NSMutableAttributedString(string: "\(followersCount)\n", attributes: [NSAttributedStringKey.font :
                    UIFont.boldSystemFont(ofSize: 16)])

                attributedText.append(NSAttributedString(string: "followers", attributes: [NSAttributedStringKey.foregroundColor:
                    UIColor.black,NSAttributedStringKey.font: UIFont.boldSystemFont(ofSize: 16)]))

                self.followers.attributedText = attributedText

                newRef.observeSingleEvent(of: .childRemoved, with: { (snapshot) in

                    let followersCount = Int(snapshot.childrenCount)
                    let attributedText = NSMutableAttributedString(string: "\(followersCount)\n", attributes: [NSAttributedStringKey.font :
                        UIFont.boldSystemFont(ofSize: 16)])

                    attributedText.append(NSAttributedString(string: "followers", attributes: [NSAttributedStringKey.foregroundColor:
                        UIColor.black,NSAttributedStringKey.font: UIFont.boldSystemFont(ofSize: 16)]))

                    self.followers.attributedText = attributedText

                }, withCancel: nil)


            }, withCancel: nil)
        }


    }, withCancel: nil)


}
```

Displaying following users and number of posts is similar to displaying number of followers

Follow/Unfollow Users

```
@objc func handleEditFollow(){

    guard  let currentUserID = Auth.auth().currentUser?.uid else { return }
    guard let userId = user?.uid else {return}


    if editFollowButton.titleLabel?.text == "UnFollow"{

        Database.database().reference().child("following").child(currentUserID).child(userId).removeValue { (error, reference) in

            if let error = error{
                print("Can't unfollow user", error)
            }
            print("Successfully unfollowed user")
            self.setFollowStyle()
        }
    }

    else {

        let ref = Database.database().reference().child("following").child(currentUserID)
        let values = [userId: 1]

        ref.updateChildValues(values) { (error, reference) in
            if let error = error{
                print("Failed to follow user", error)
            }

            print("Successfully folowed user", self.user?.username ?? "")
            self.editFollowButton.backgroundColor =
            self.editFollowButton.setTitle("UnFollow", for: .normal)
            self.editFollowButton.setTitleColor(.black, for: .normal)
            self.editFollowButton.layer.borderColor = UIColor(white: 0, alpha: 0.2).cgColor

        }
    }

}
```

Log Out

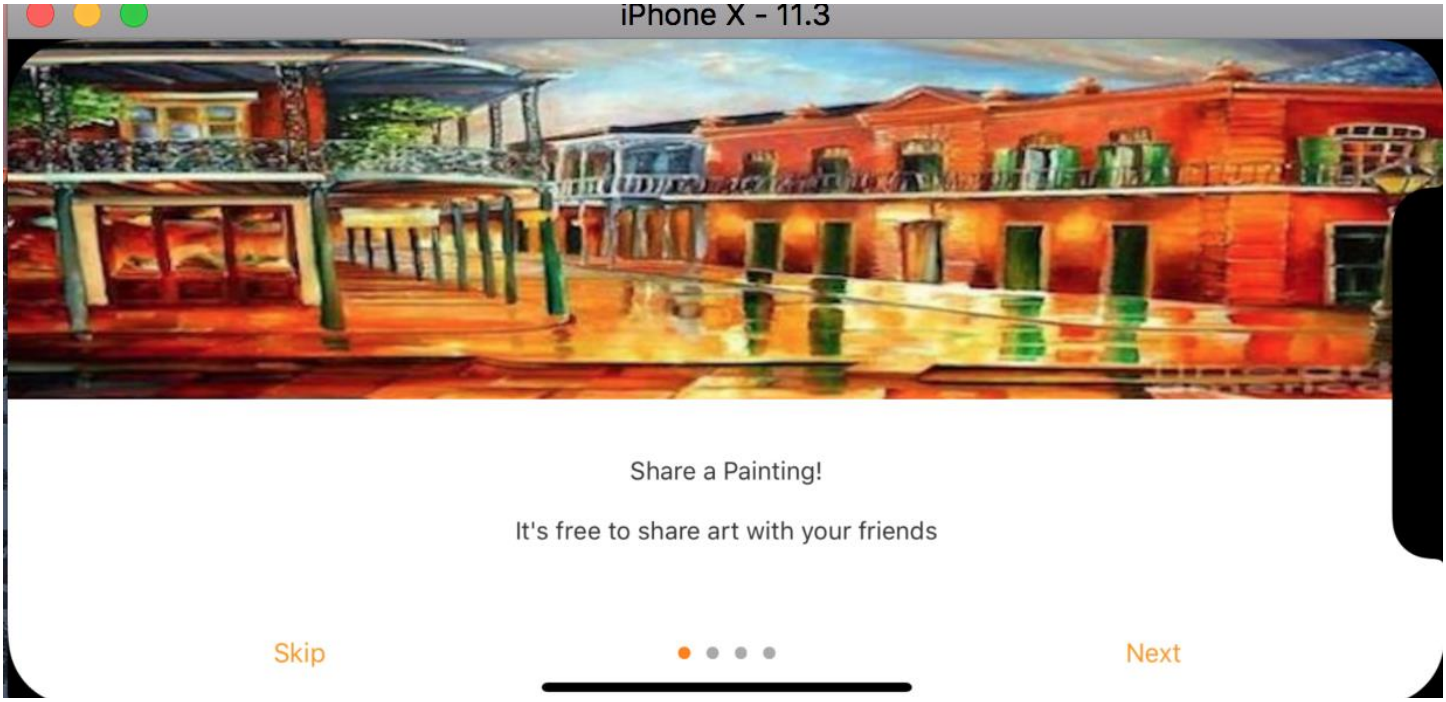
```
@objc func handleLogOut(){

    print("inside log out function!")

    let alertController = UIAlertController(title: nil, message: nil, preferredStyle: .actionSheet)
    present(alertController, animated: true, completion: nil)
    alertController.addAction(UIAlertAction(title: "Log Out", style: .destructive, handler: { (_) in

        print("performing log out!")

        do{

            try Auth.auth().signOut()
            //let logInController = LogInController()
            let logInController = LoginGuideViewController()
            let navController = UINavigationController(rootViewController: logInController)
            self.present(navController, animated: true, completion: nil)

        } catch let signOutError {
            print("Error while Logging out", signOutError)
        }

    }))

    alertController.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: nil))

}
```

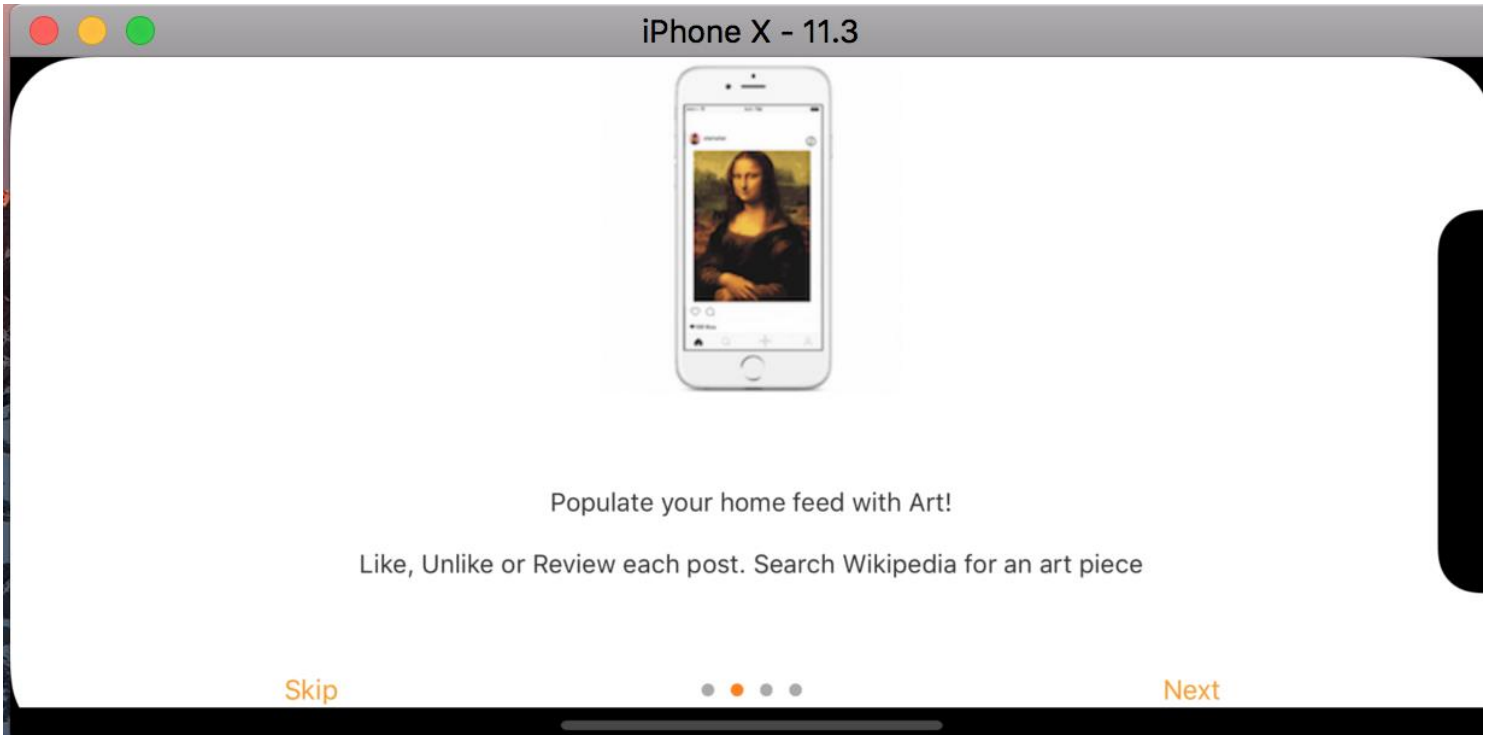
Log In and Sign Up buttons change colors when all the fields are filled

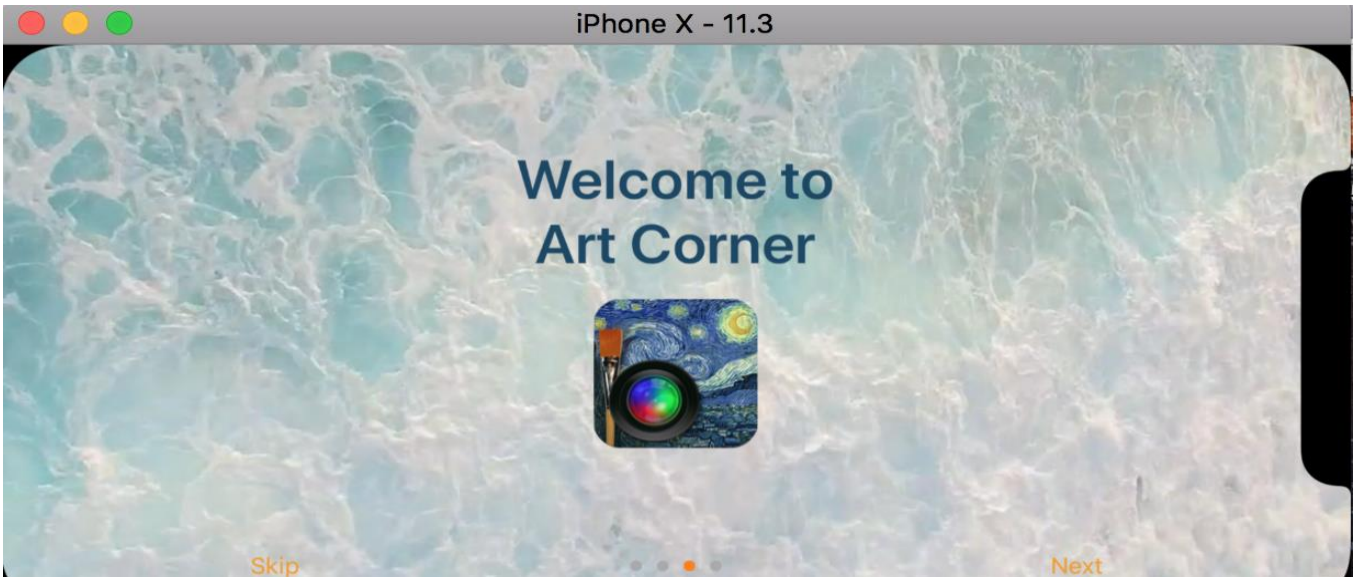
```
@objc func handleInputChange(){

    let isInfoValid = emailTextField.text?.count ?? 0 > 0 && userNameTextField.text?.count ?? 0 > 0 && passwordTextField.text?.count ?? 0 > 0

    if isInfoValid {

        signUpButton.isEnabled = true
        signUpButton.backgroundColor =
        signUpButton.setTitleColor(.white, for: .normal)
    }

    else{
        signUpButton.isEnabled = false
        signUpButton.backgroundColor =
        signUpButton.setTitleColor(.white, for: .normal)
    }

}
```

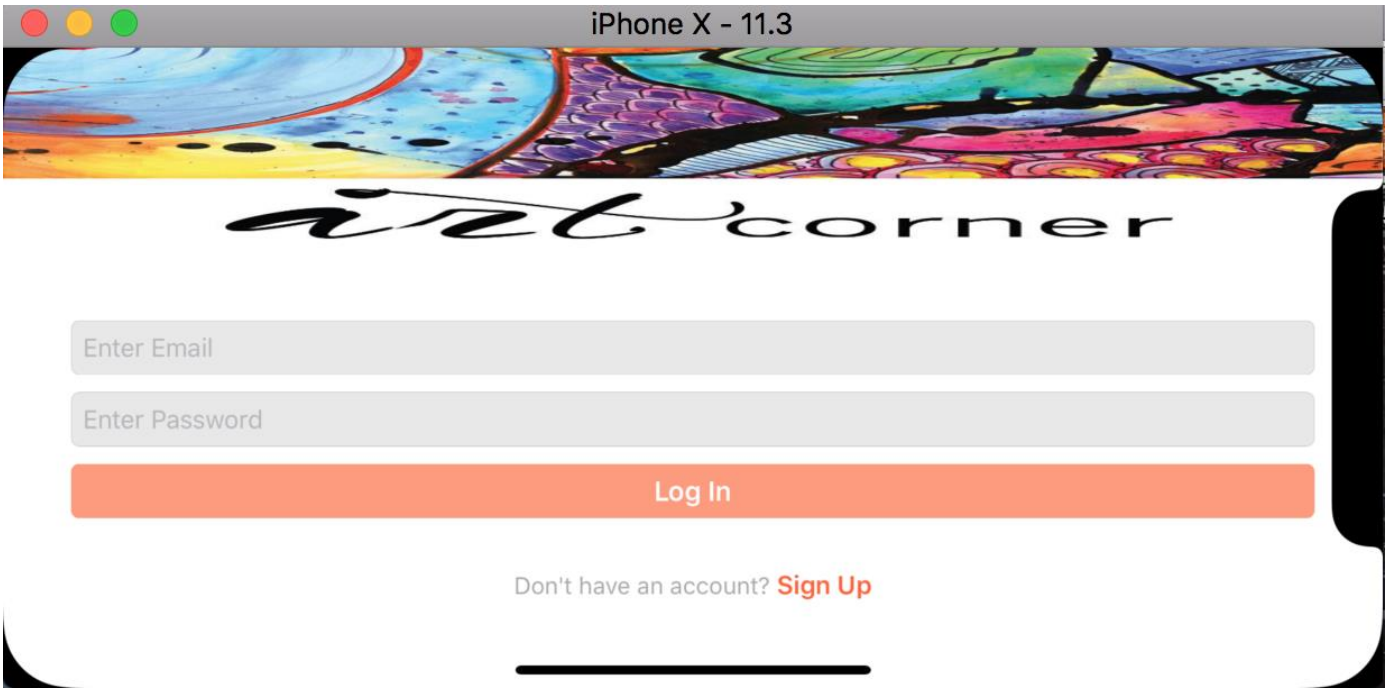
Firebase Tree Structure

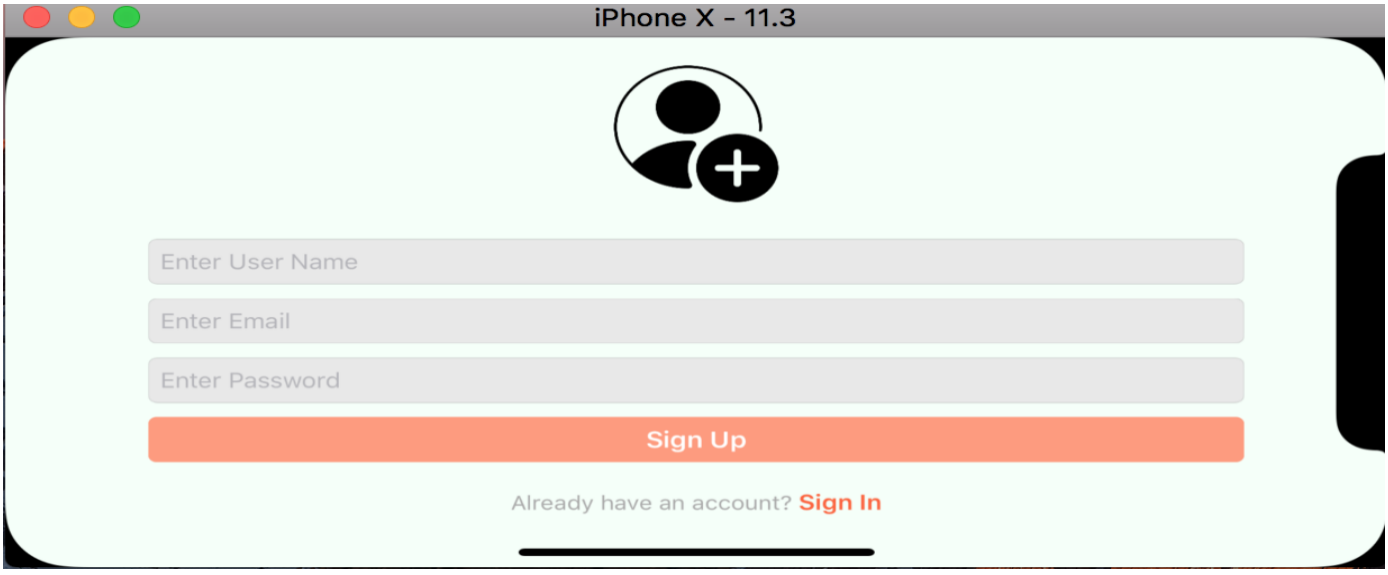
Auto Layout:

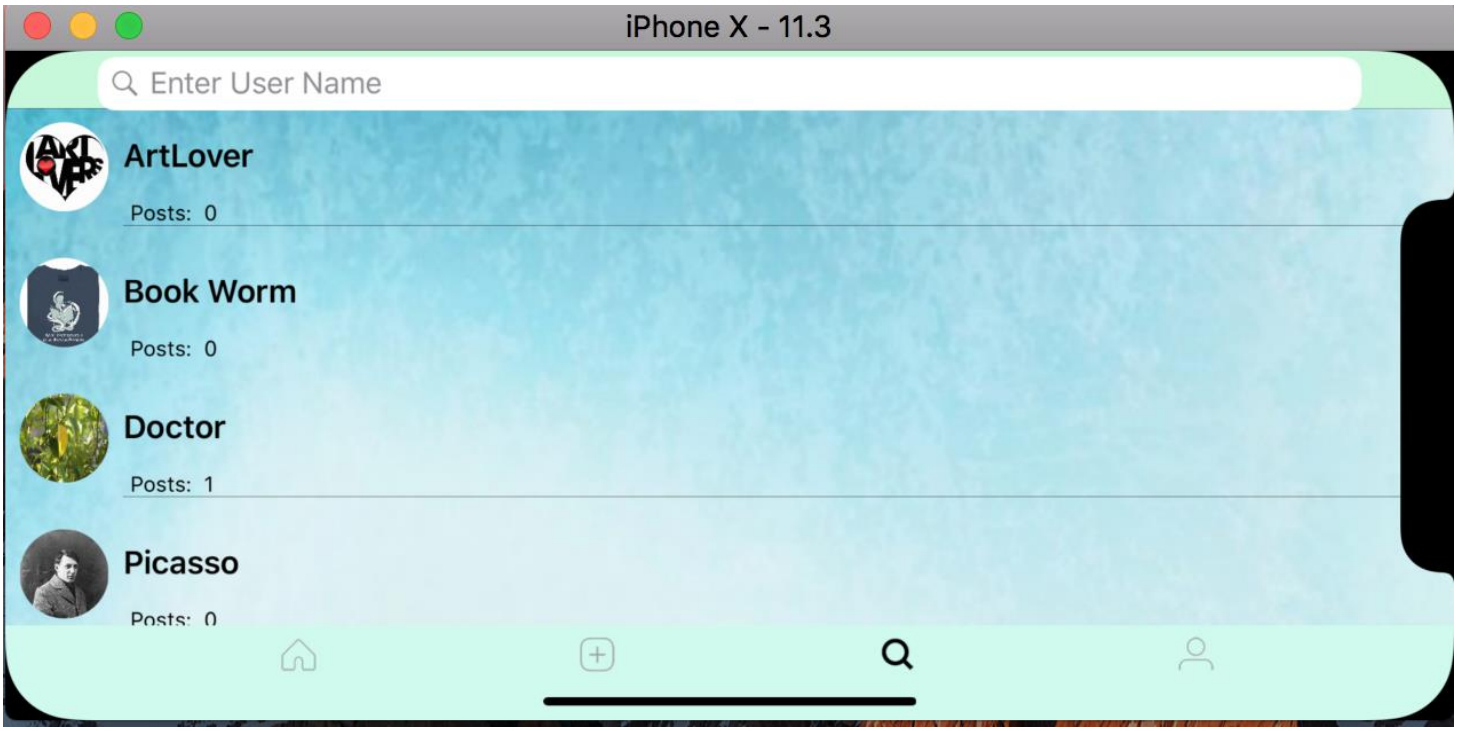

iPhone X - 11.3
Share a Painting!
It's free to share art with your friends
Skip
Next

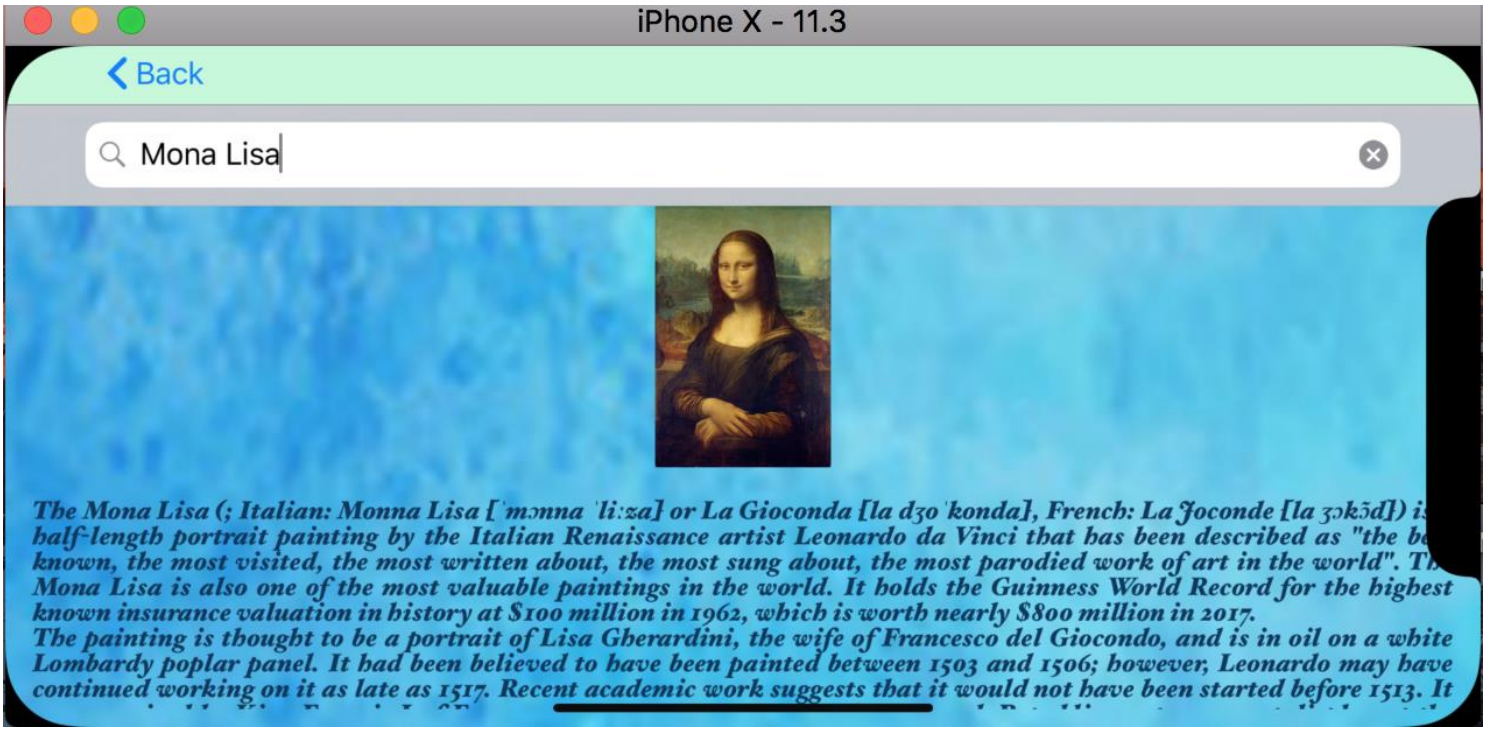

iPhone X - 11.3
Populate your home feed with Art!
Like, Unlike or Review each post. Search Wikipedia for an art piece
Skip
Next


iPhone X - 11.3
Welcome to
Art Corner
Skip
Next


iPhone X - 11.3
art corner
Enter Email
Enter Password
Log In
Don't have an account? Sign Up


iPhone X - 11.3
Enter User Name
Enter Email
Enter Password
Sign Up
Already have an account? Sign In


iPhone X - 11.3
Enter User Name
ArtLover
Posts: 0
Book Worm
Posts: 0
Doctor
Posts: 1
Picasso
Posts: 0


iPhone X - 11.3
Back
Mona Lisa
The Mona Lisa (; Italian: Monna Lisa [ˈmɔnna ˈliːza] or La Gioconda [la dʒoˈkonda], French: La Joconde [la ʒɔkɔ̃d]) is
half-length portrait painting by the Italian Renaissance artist Leonardo da Vinci that has been described as "the be
known, the most visited, the most written about, the most sung about, the most parodied work of art in the world". Th
Mona Lisa is also one of the most valuable paintings in the world. It holds the Guinness World Record for the highest
known insurance valuation in history at $100 million in 1962, which is worth nearly $800 million in 2017.
The painting is thought to be a portrait of Lisa Gherardini, the wife of Francesco del Giocondo, and is in oil on a white
Lombardy poplar panel. It had been believed to have been painted between 1503 and 1506; however, Leonardo may have
continued working on it as late as 1517. Recent academic work suggests that it would not have been started before 1513. It