FLORAPEDIA

Ashwini Shekhar Phadke

SUID : 575445713

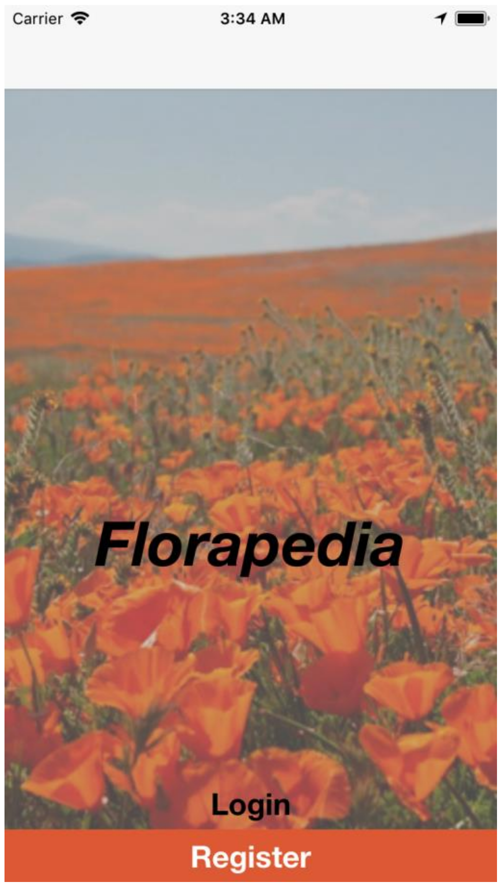
Contents

Introduction

Florapedia is an app designed to give its users an encyclopedia and a journal on the go. It has three menus on the tabbar to choose from journal, encyclopedia and trail. Users can note their findings and store descriptions about it in the journal menu. Encyclopedia offers users to point and click the flower they want to identify and the Wikipedia information about it will be pulled in real time giving the name and brief description about the flower. They can also share the clicked picture on Facebook. In the trail menu, the user can put annotations on the map to denote the interesting findings and even customize the annotation title.

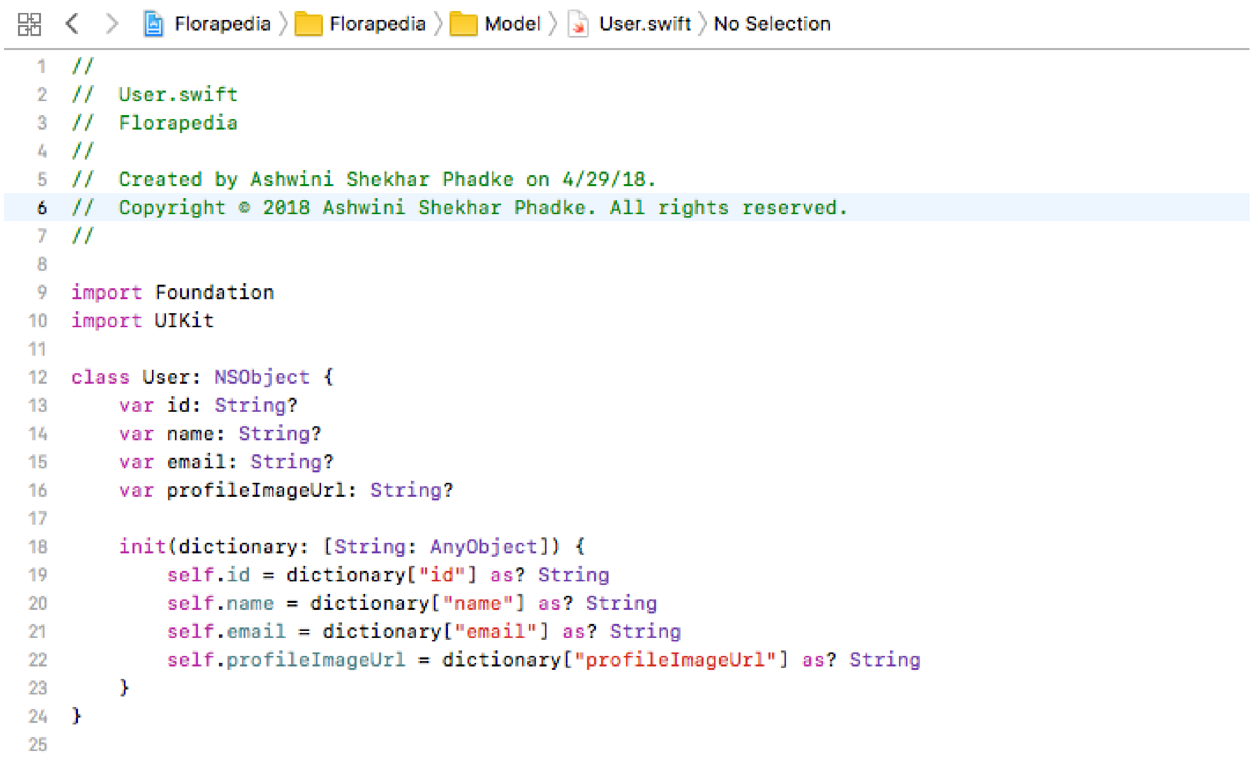
The app intents to aid nature enthusiasts in discovering new plants and save the notes of their interesting findings!

Welcome Screen



Register User and Logging In

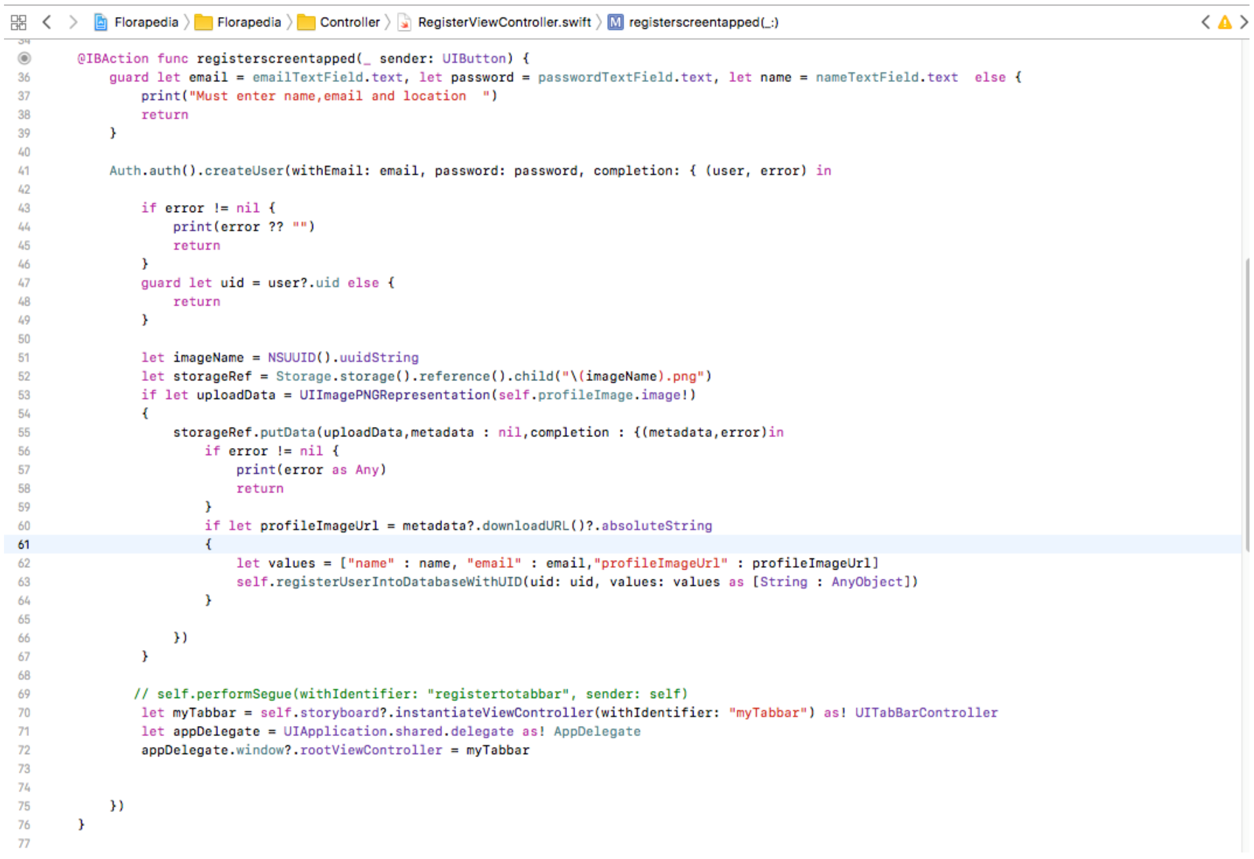
User Model



```swift
//
//  User.swift
//  Florapedia
//
//  Created by Ashwini Shekhar Phadke on 4/29/18.
//  Copyright © 2018 Ashwini Shekhar Phadke. All rights reserved.
//

import Foundation
import UIKit

class User: NSObject {
    var id: String?
    var name: String?
    var email: String?
    var profileImageUrl: String?

    init(dictionary: [String: AnyObject]) {
        self.id = dictionary["id"] as? String
        self.name = dictionary["name"] as? String
        self.email = dictionary["email"] as? String
        self.profileImageUrl = dictionary["profileImageUrl"] as? String
    }
}
```

When a new user signs up in the app, using the register screen he/she has to enter name, image and email id to signup to the app. The details entered by the user are populated in the user object and then stored to firebase

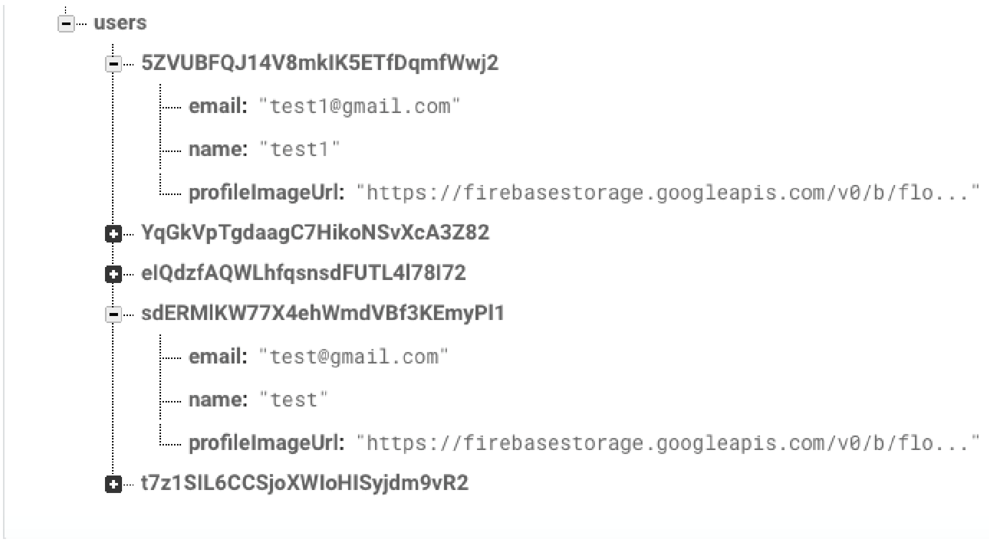
The authentication method used is firebase email authentication.



Florapedia 〉 Florapedia 〉 Controller 〉 RegisterViewController.swift 〉 registerscreentapped(_:)

```
    @IBAction func registerscreentapped(_ sender: UIButton) {
        guard let email = emailTextField.text, let password = passwordTextField.text, let name = nameTextField.text  else {
            print("Must enter name,email and location  ")
            return
        }

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

            if error != nil {
                print(error ?? "")
                return
            }
            guard let uid = user?.uid else {
                return
            }

            let imageName = NSUUID().uuidString
            let storageRef = Storage.storage().reference().child("\(imageName).png")
            if let uploadData = UIImagePNGRepresentation(self.profileImage.image!)
            {
                storageRef.putData(uploadData,metadata : nil,completion : {(metadata,error)in
                    if error != nil {
                        print(error as Any)
                        return
                    }
                    if let profileImageUrl = metadata?.downloadURL()?.absoluteString
                    {
                        let values = ["name" : name, "email" : email,"profileImageUrl" : profileImageUrl]
                        self.registerUserIntoDatabaseWithUID(uid: uid, values: values as [String : AnyObject])
                    }

                })
            }

           // self.performSegue(withIdentifier: "registertotabbar", sender: self)
            let myTabbar = self.storyboard?.instantiateViewController(withIdentifier: "myTabbar") as! UITabBarController
            let appDelegate = UIApplication.shared.delegate as! AppDelegate
            appDelegate.window?.rootViewController = myTabbar


        })
    }
```

Once the register button is tapped, the user is authenticated by Firebase using the Auth.auth()createuser method. The user is saved to the database and the profile image is stored on the storage in firebase.



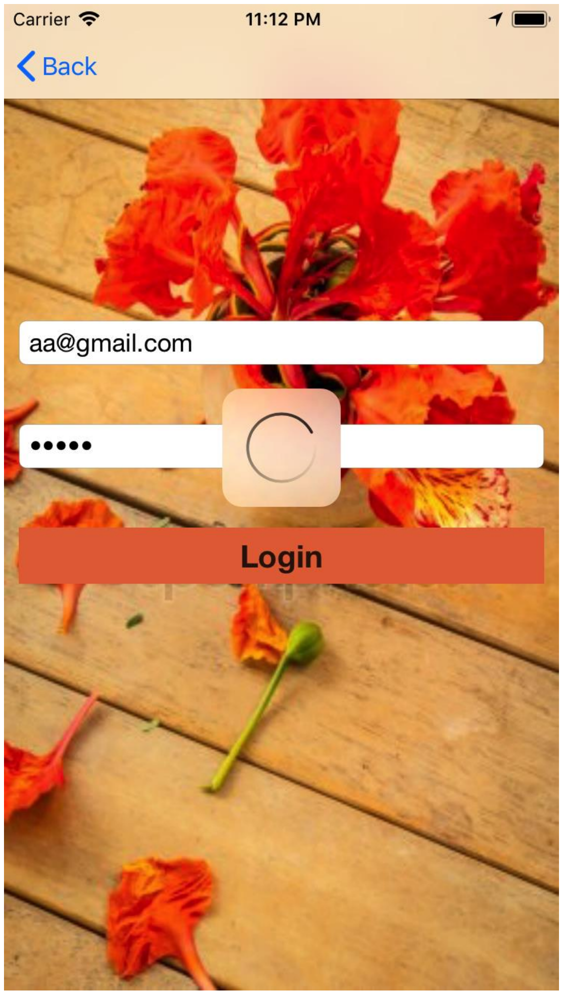
```
users
  5ZVUBFQJ14V8mkIK5ETfDqmfWwj2
    email: "test1@gmail.com"
    name: "test1"
    profileImageUrl: "https://firebasestorage.googleapis.com/v0/b/flo..."
  YqGkVpTgdaagC7HikoNSvXcA3Z82
  elQdzfAQWLhfqsnsdFUTL4l78l72
  sdERMlKW77X4ehWmdVBf3KEmyPl1
    email: "test@gmail.com"
    name: "test"
    profileImageUrl: "https://firebasestorage.googleapis.com/v0/b/flo..."
  t7z1SIL6CCSjoXWloHISyjdm9vR2
```

Login View Controller

If the user is already registered, then from the welcome screen he can directly proceed to then login screen .The email id and password which are already registered are used to authenticate the user.

For better user experience, a loading wheel is presented to the user when the authentication is being processed at the backend. This is achieved by incorporating SVProgressHUD pod. The method SVProgressHUD.show() and SVProgressHUD.dismiss() method is called on clicking the login button.

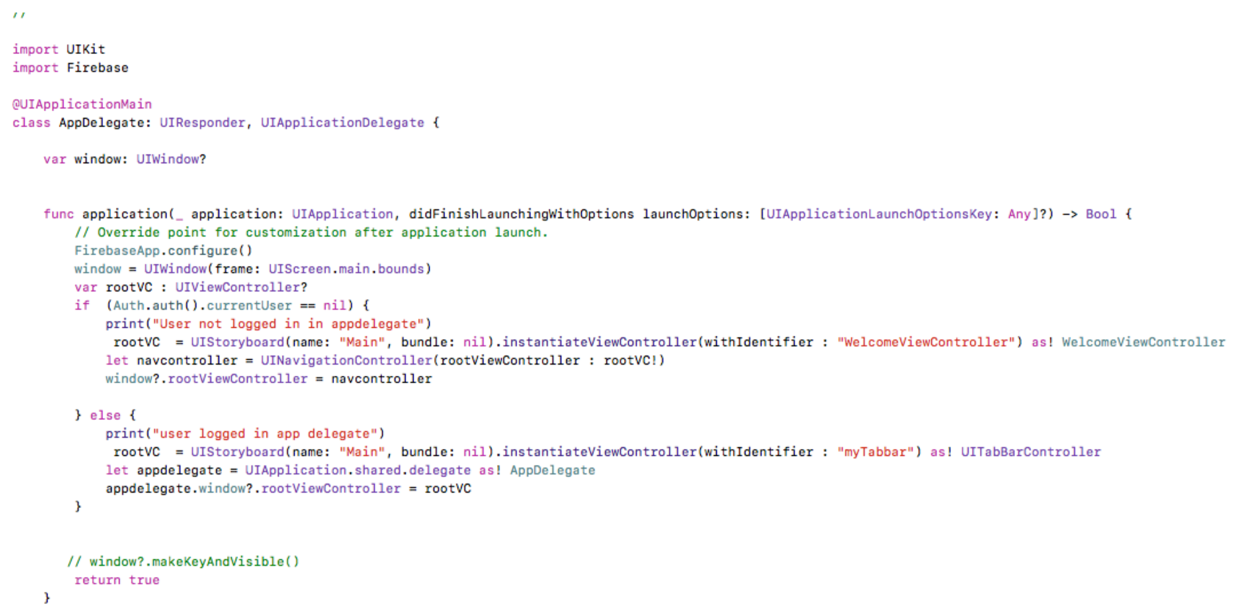
This ensures that the user knows some authentication is going on the app has not frozen!



Relogging in the app

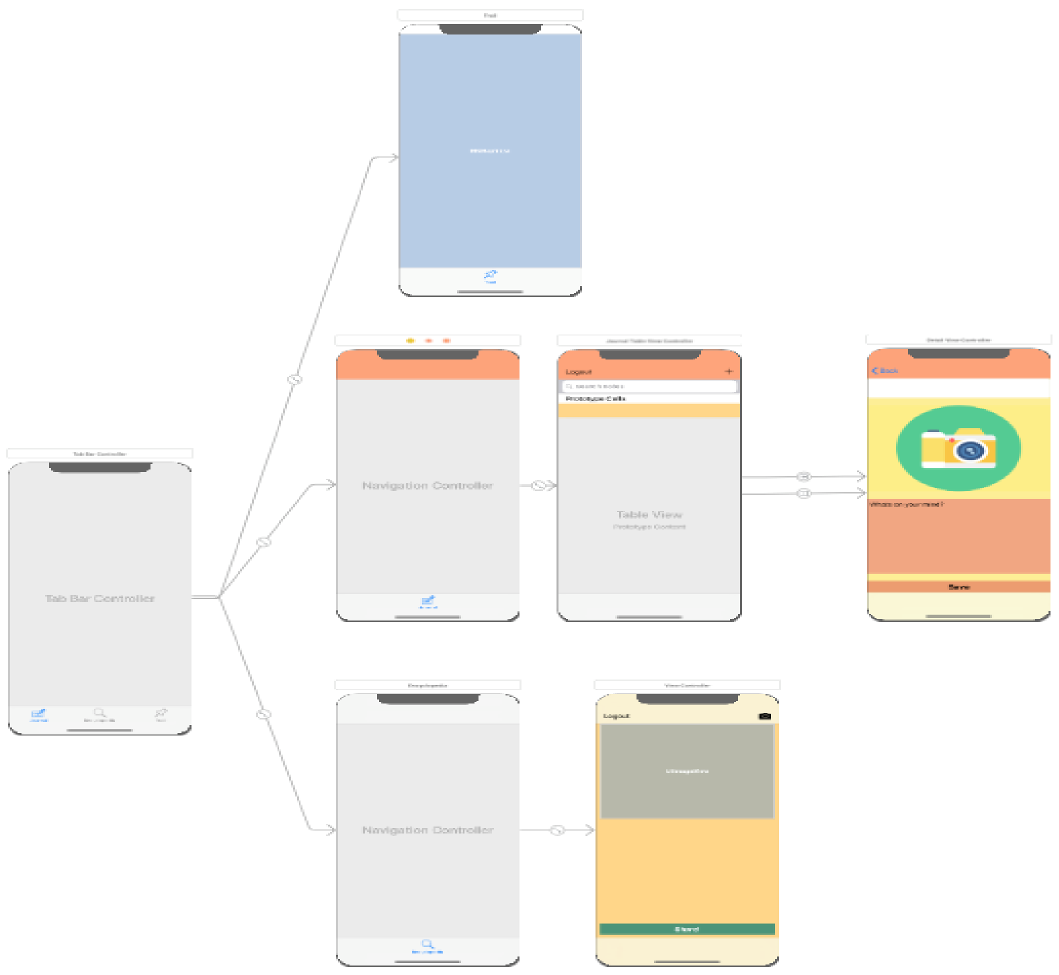
If the user has already logged in then upon closing app and starting it again, user does not have to login back again. The logged in state is saved and checked in the app delegate. If the user has logged out then he is presented with the login screen else the user is directly taken to the tabbar controller to use the app. This makes the user experience more enriching.

The following code in the app delegate shows the check which is performed everytime the app is launched and thus provides a better user experience

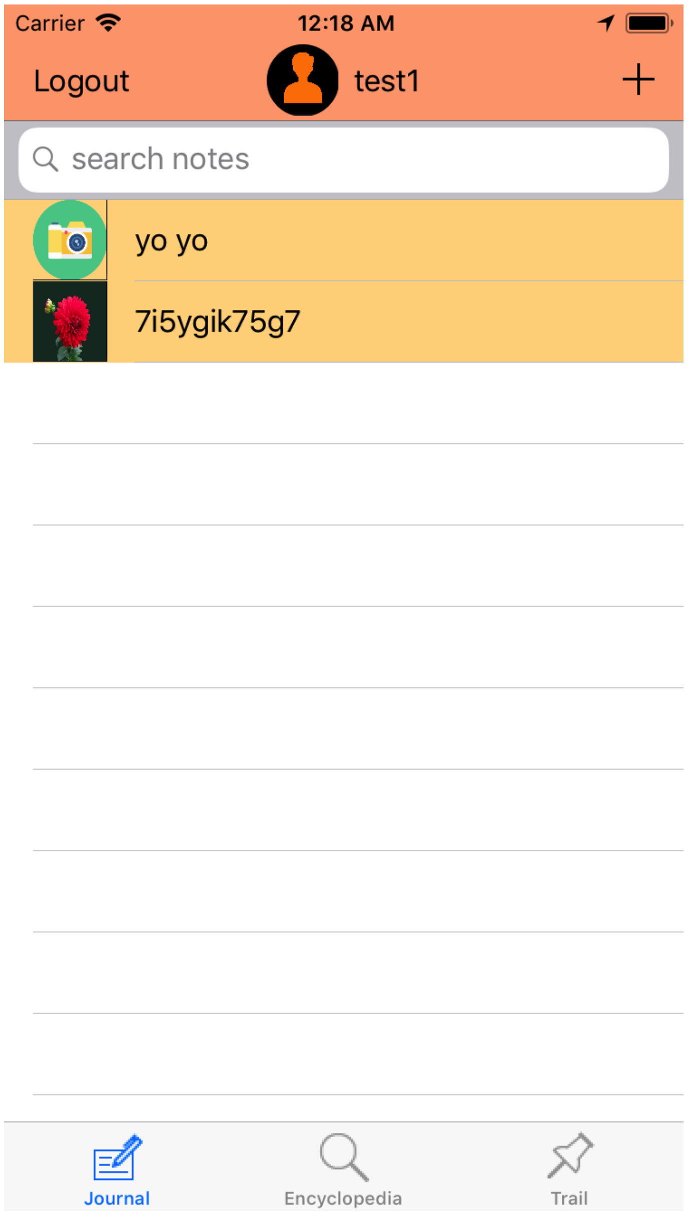


```
//

import UIKit
import Firebase

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        FirebaseApp.configure()
        window = UIWindow(frame: UIScreen.main.bounds)
        var rootVC : UIViewController?
        if  (Auth.auth().currentUser == nil) {
            print("User not logged in in appdelegate")
             rootVC  = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier : "WelcomeViewController") as! WelcomeViewController
            let navcontroller = UINavigationController(rootViewController : rootVC!)
            window?.rootViewController = navcontroller

        } else {
            print("user logged in app delegate")
             rootVC  = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier : "myTabbar") as! UITabBarController
            let appdelegate = UIApplication.shared.delegate as! AppDelegate
            appdelegate.window?.rootViewController = rootVC
        }


       // window?.makeKeyAndVisible()
        return true
    }
```

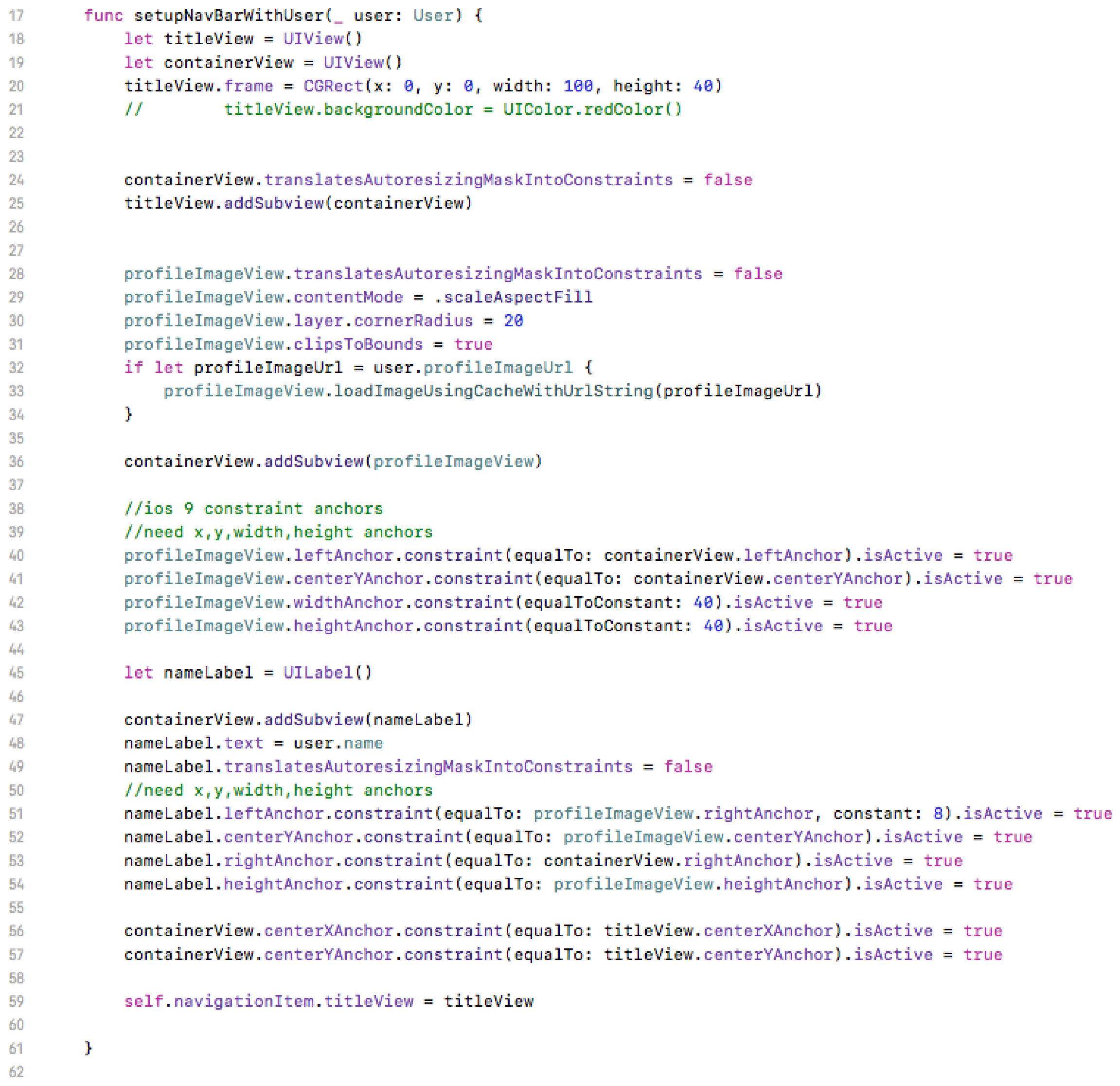
Once, the user has successfully logged in or registered the root view controller is changed to to the tabbar controller. The items in the tababar controller embed a navigation controller in them so as to facilitate the back movement on the screen.


Tab Bar Controller
Navigation Controller
Logout
Prototype Cells
Table View
Prototype Content
Back
Whats on your mind?
Save
Navigation Controller
Logout
Shared

JournalTableViewController (Master -Detail Format)



The journal table view displays all the titles of the note created by the user along with the picture uploaded by the user. It appears as the first option on the tabbar. The navigation bar displays the username and profile picture. The data is fetched from the firebase, under Users database. The navigation bar is set up using the following method



```
    func setupNavBarWithUser(_ user: User) {
        let titleView = UIView()
        let containerView = UIView()
        titleView.frame = CGRect(x: 0, y: 0, width: 100, height: 40)
        //        titleView.backgroundColor = UIColor.redColor()


        containerView.translatesAutoresizingMaskIntoConstraints = false
        titleView.addSubview(containerView)


        profileImageView.translatesAutoresizingMaskIntoConstraints = false
        profileImageView.contentMode = .scaleAspectFill
        profileImageView.layer.cornerRadius = 20
        profileImageView.clipsToBounds = true
        if let profileImageUrl = user.profileImageUrl {
            profileImageView.loadImageUsingCacheWithUrlString(profileImageUrl)
        }

        containerView.addSubview(profileImageView)

        //ios 9 constraint anchors
        //need x,y,width,height anchors
        profileImageView.leftAnchor.constraint(equalTo: containerView.leftAnchor).isActive = true
        profileImageView.centerYAnchor.constraint(equalTo: containerView.centerYAnchor).isActive = true
        profileImageView.widthAnchor.constraint(equalToConstant: 40).isActive = true
        profileImageView.heightAnchor.constraint(equalToConstant: 40).isActive = true

        let nameLabel = UILabel()

        containerView.addSubview(nameLabel)
        nameLabel.text = user.name
        nameLabel.translatesAutoresizingMaskIntoConstraints = false
        //need x,y,width,height anchors
        nameLabel.leftAnchor.constraint(equalTo: profileImageView.rightAnchor, constant: 8).isActive = true
        nameLabel.centerYAnchor.constraint(equalTo: profileImageView.centerYAnchor).isActive = true
        nameLabel.rightAnchor.constraint(equalTo: containerView.rightAnchor).isActive = true
        nameLabel.heightAnchor.constraint(equalTo: profileImageView.heightAnchor).isActive = true

        containerView.centerXAnchor.constraint(equalTo: titleView.centerXAnchor).isActive = true
        containerView.centerYAnchor.constraint(equalTo: titleView.centerYAnchor).isActive = true

        self.navigationItem.titleView = titleView

    }
```

Table cell

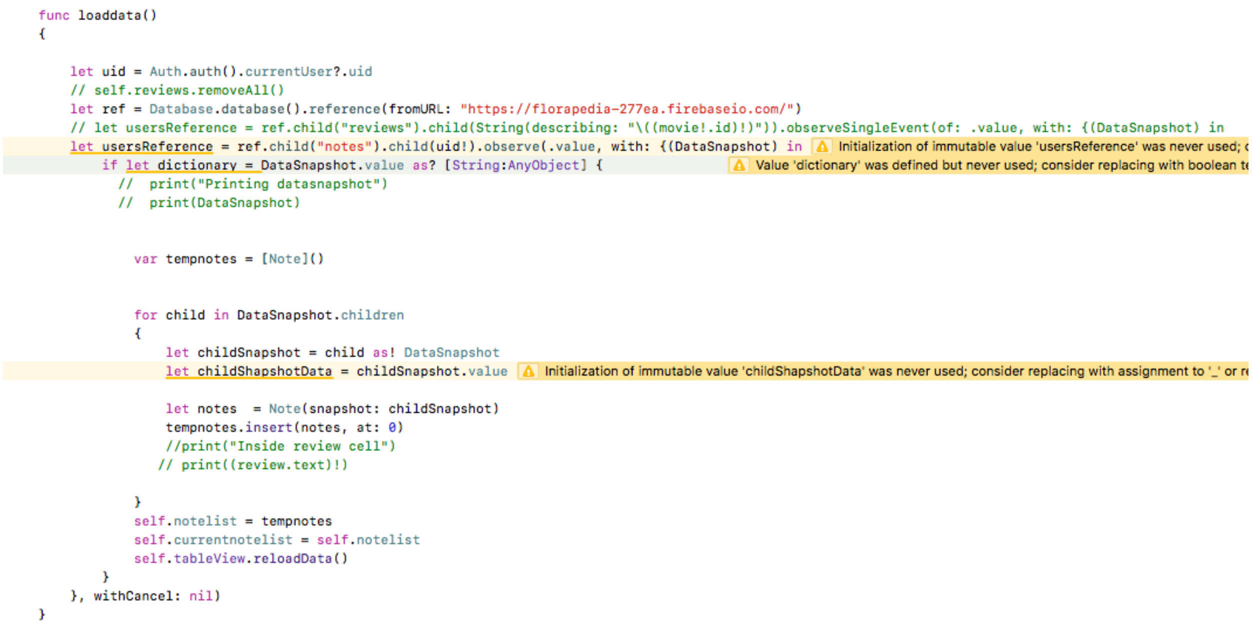
The cell is populated in the cell for row at method. The data is fetched from the user in real time

The loaddata() method is calle din the viewwillappear() so as to update the table view with the latest data.

In the viewdidload method, all the delegate and datasources are specified to self.

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

    tableView.dataSource = self
    tableView.delegate = self
        searchbar.delegate = self
        currentnotelist = notelist
        loaddata()
        tableView.reloadData()
        tableView.allowsMultipleSelectionDuringEditing = true
        self.hideKeyboard()
    }
```

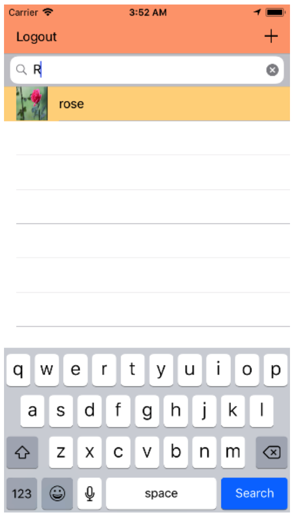
loaddata() method is used to populate data in the tableview cells

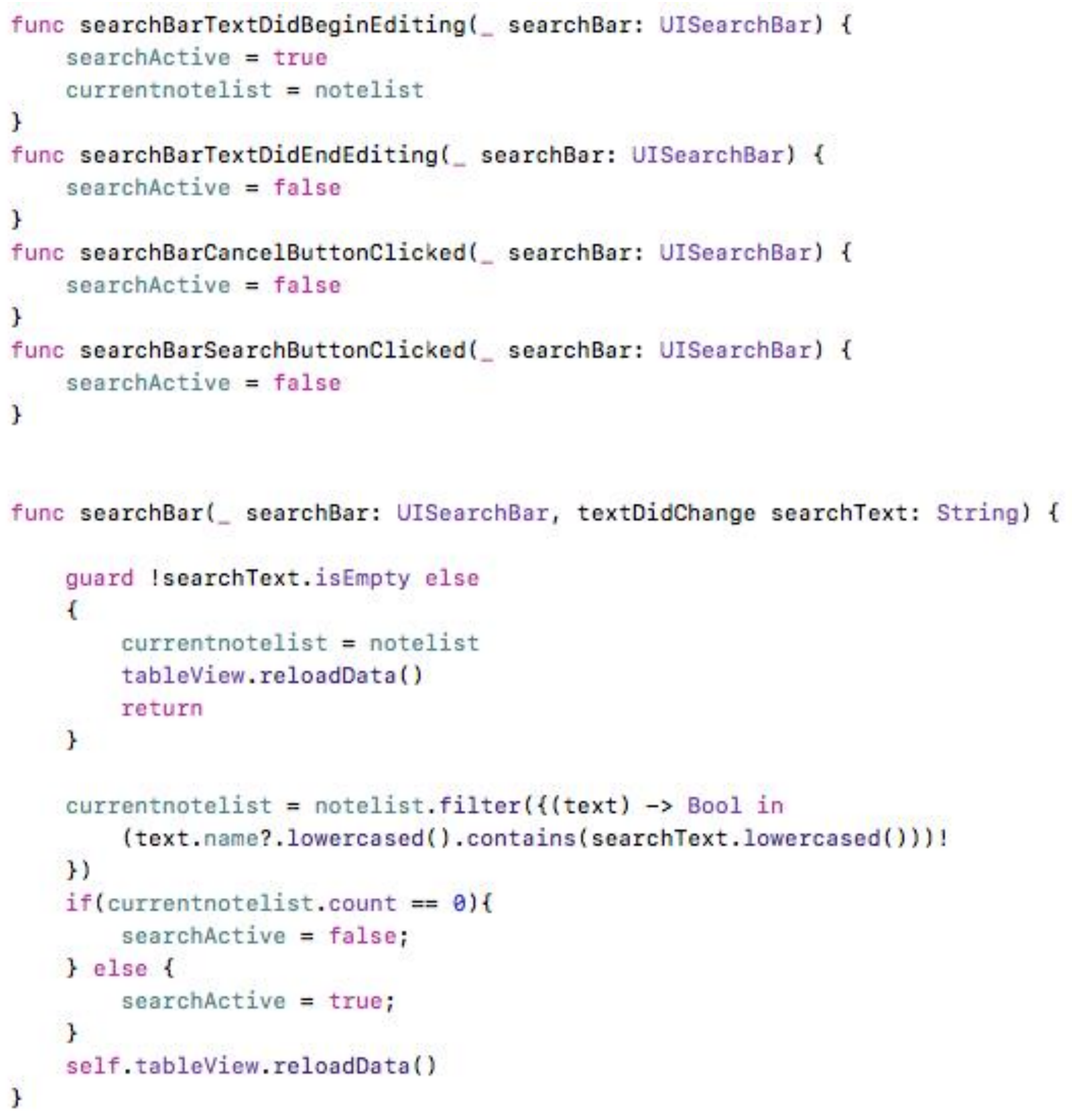


```
func loaddata()
{

    let uid = Auth.auth().currentUser?.uid
    // self.reviews.removeAll()
    let ref = Database.database().reference(fromURL: "https://florapedia-277ea.firebaseio.com/")
    // let usersReference = ref.child("reviews").child(String(describing: "\((movie!.id)!)")).observeSingleEvent(of: .value, with: {(DataSnapshot) in
    let usersReference = ref.child("notes").child(uid!).observe(.value, with: {(DataSnapshot) in   ⚠ Initialization of immutable value 'usersReference' was never used; c
        if let dictionary = DataSnapshot.value as? [String:AnyObject] {   ⚠ Value 'dictionary' was defined but never used; consider replacing with boolean te
          //  print("Printing datasnapshot")
          //  print(DataSnapshot)


            var tempnotes = [Note]()


            for child in DataSnapshot.children
            {
                let childSnapshot = child as! DataSnapshot
                let childShapshotData = childSnapshot.value   ⚠ Initialization of immutable value 'childShapshotData' was never used; consider replacing with assignment to '_' or re

                let notes  = Note(snapshot: childSnapshot)
                tempnotes.insert(notes, at: 0)
                //print("Inside review cell")
               // print((review.text)!)

            }
            self.notelist = tempnotes
            self.currentnotelist = self.notelist
            self.tableView.reloadData()
        }
    }, withCancel: nil)
}
```

Searchbar

The searchbar at the top of the tableview helps in sorting through the note titles in the tableview.The following code snippet describes the implementation of search bar.



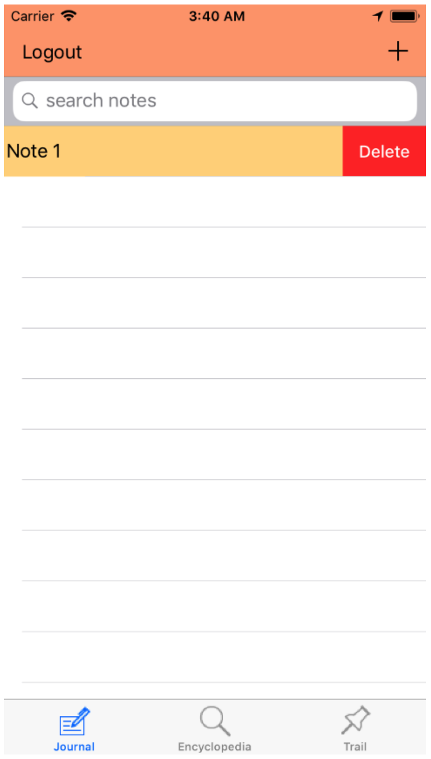


```
func searchBarTextDidBeginEditing(_ searchBar: UISearchBar) {
    searchActive = true
    currentnotelist = notelist
}
func searchBarTextDidEndEditing(_ searchBar: UISearchBar) {
    searchActive = false
}
func searchBarCancelButtonClicked(_ searchBar: UISearchBar) {
    searchActive = false
}
func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
    searchActive = false
}


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

    guard !searchText.isEmpty else
    {
        currentnotelist = notelist
        tableView.reloadData()
        return
    }

    currentnotelist = notelist.filter({(text) -> Bool in
        (text.name?.lowercased().contains(searchText.lowercased()))!
    })
    if(currentnotelist.count == 0){
        searchActive = false;
    } else {
        searchActive = true;
    }
    self.tableView.reloadData()
}
```

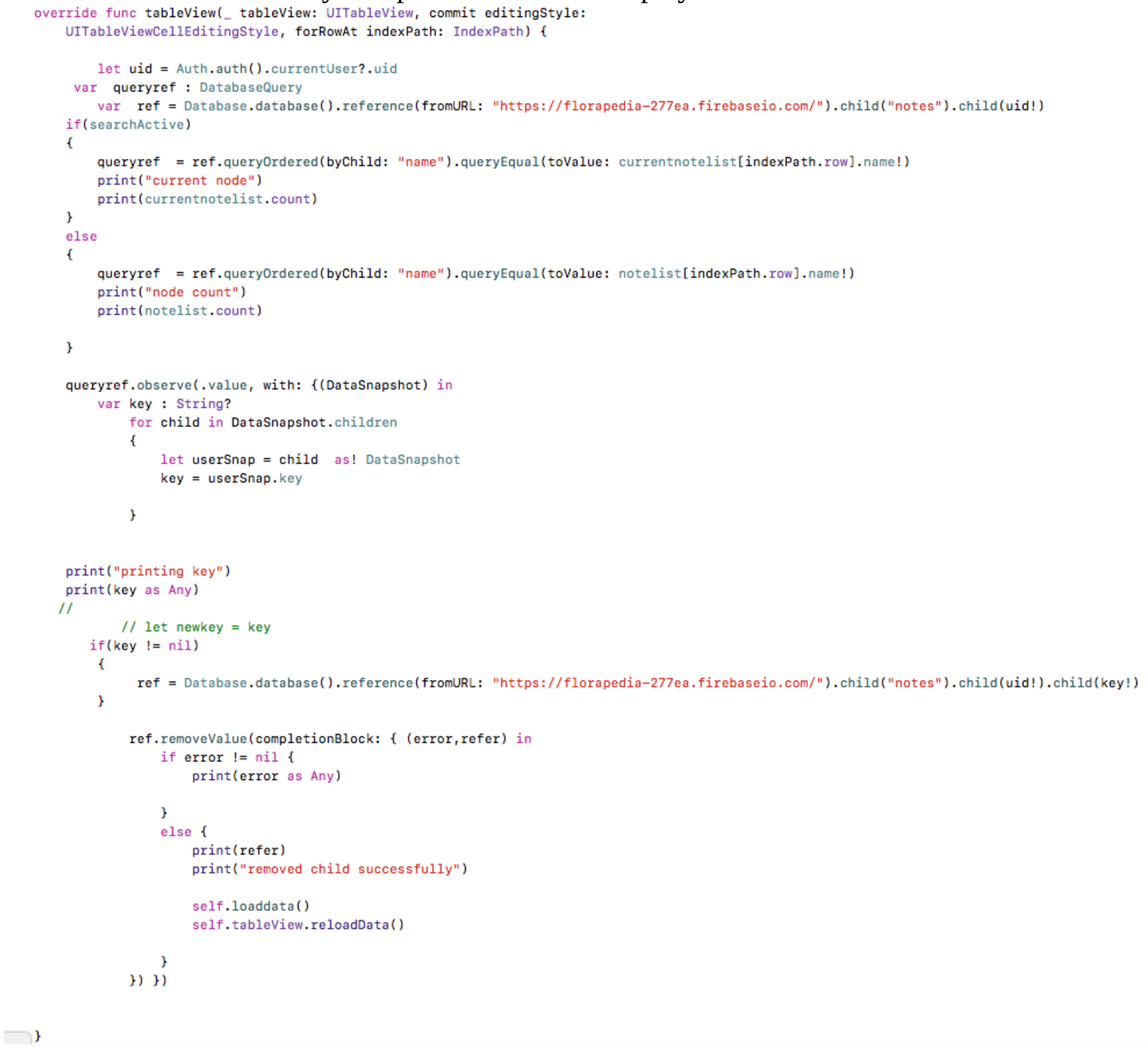
The currentnotelist[Notes]() stores the updated values of the notes retrieved from the firebase and displays on screen. The searchactive flag is made active whenever search is in progress and thus, the cells use the currnetnotelist [] to fetch data to populate them.

Delete note

The cells in the tableview can be deleted by swiping them. That gesture is achieved by implementing caneditrow() method in tableview. When the row is selected , at the backend the child associated with note in the firebase is removed and the view is reloaded.



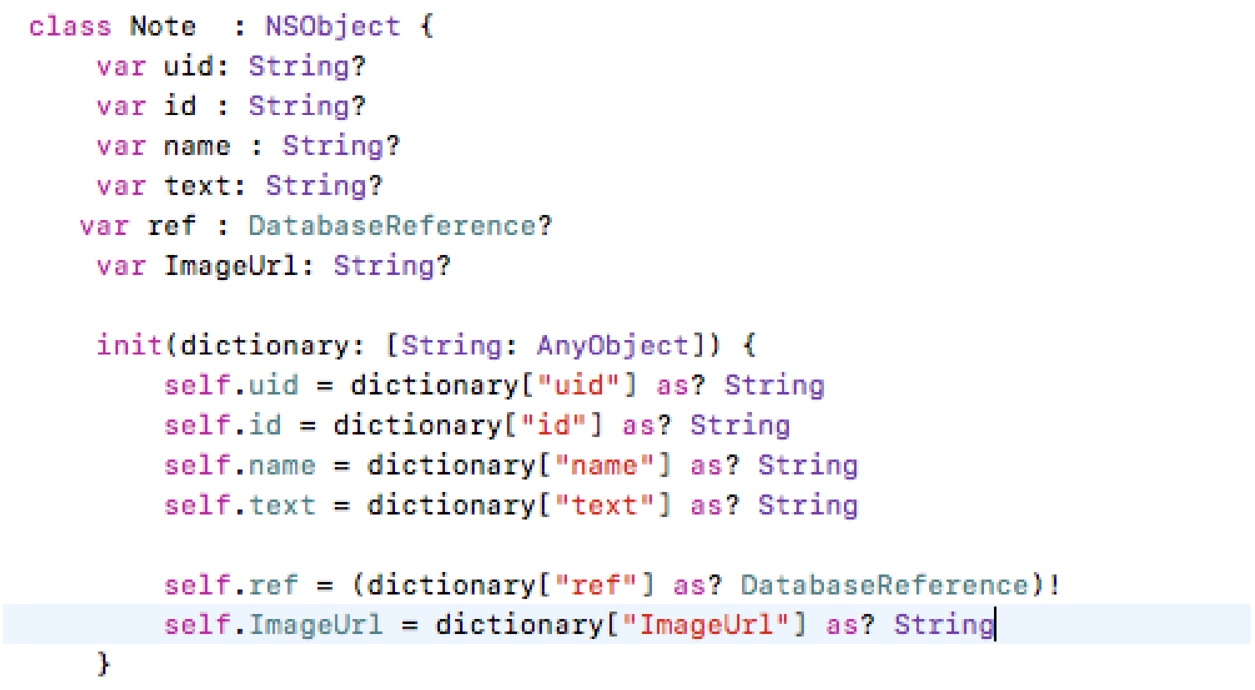
The database structure stores all the notes under the specific user key. So data security is obtained as the data of only that particular user is displayed on screen.



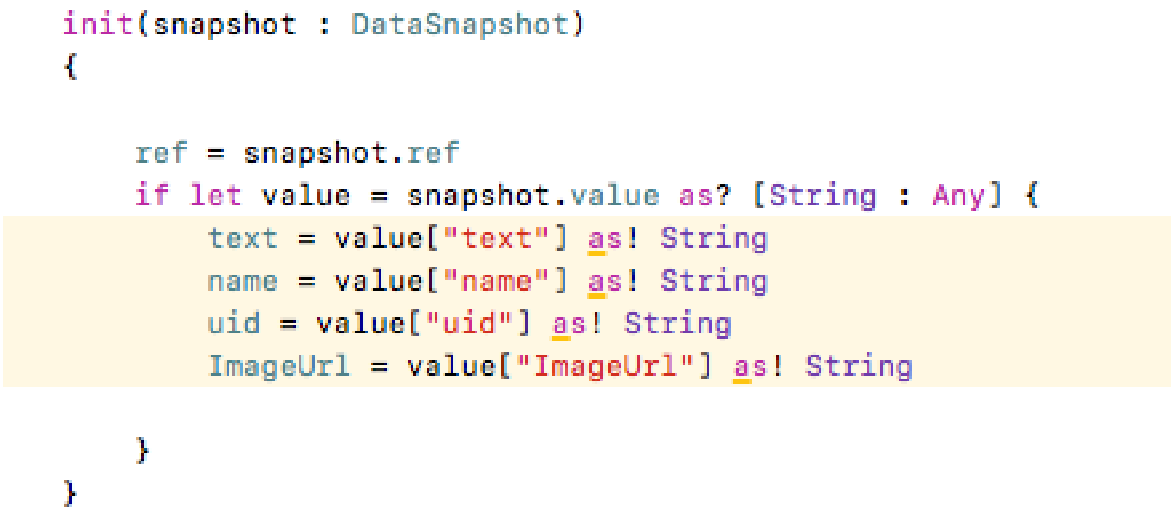
```
    override func tableView(_ tableView: UITableView, commit editingStyle:
        UITableViewCellEditingStyle, forRowAt indexPath: IndexPath) {

            let uid = Auth.auth().currentUser?.uid
         var  queryref : DatabaseQuery
            var  ref = Database.database().reference(fromURL: "https://florapedia-277ea.firebaseio.com/").child("notes").child(uid!)
        if(searchActive)
        {
            queryref  = ref.queryOrdered(byChild: "name").queryEqual(toValue: currentnotelist[indexPath.row].name!)
            print("current node")
            print(currentnotelist.count)
        }
        else
        {
            queryref  = ref.queryOrdered(byChild: "name").queryEqual(toValue: notelist[indexPath.row].name!)
            print("node count")
            print(notelist.count)

        }

        queryref.observe(.value, with: {(DataSnapshot) in
            var key : String?
                for child in DataSnapshot.children
                {
                    let userSnap = child  as! DataSnapshot
                    key = userSnap.key

                }


        print("printing key")
        print(key as Any)
       //
               // let newkey = key
           if(key != nil)
            {
                 ref = Database.database().reference(fromURL: "https://florapedia-277ea.firebaseio.com/").child("notes").child(uid!).child(key!)
            }

                ref.removeValue(completionBlock: { (error,refer) in
                    if error != nil {
                        print(error as Any)

                    }
                    else {
                        print(refer)
                        print("removed child successfully")

                        self.loaddata()
                        self.tableView.reloadData()

                    }
                }) })


    }
```

DetailViewController

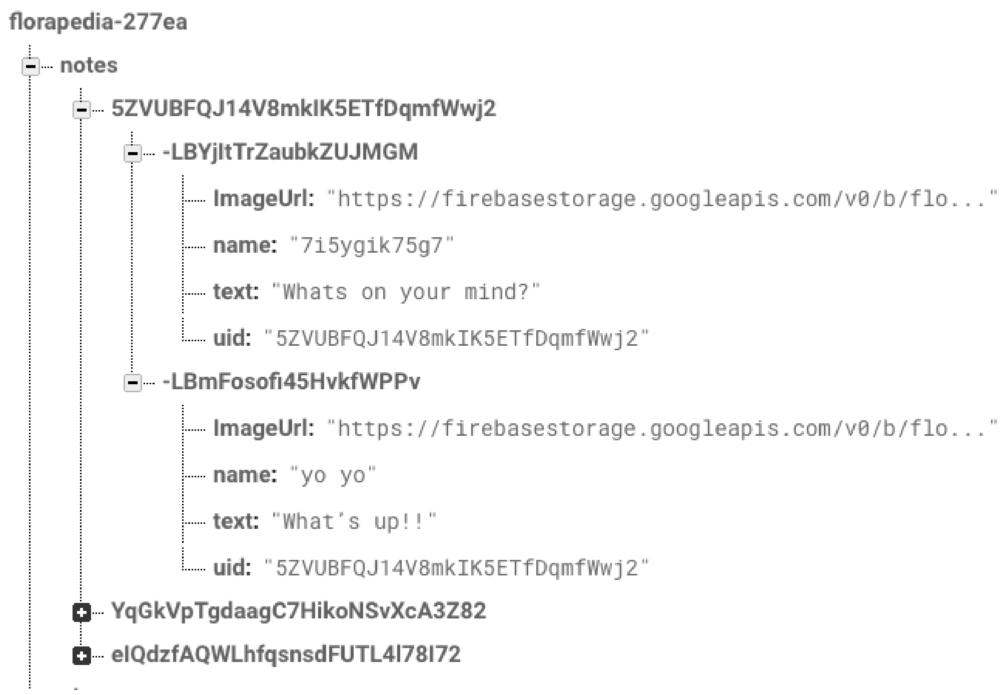
The notes are edited and posted in this controller once clicking the save button The following snippet shows the model of Notes



```
class Note  : NSObject {
    var uid: String?
    var id : String?
    var name : String?
    var text: String?
   var ref : DatabaseReference?
    var ImageUrl: String?

    init(dictionary: [String: AnyObject]) {
        self.uid = dictionary["uid"] as? String
        self.id = dictionary["id"] as? String
        self.name = dictionary["name"] as? String
        self.text = dictionary["text"] as? String

        self.ref = (dictionary["ref"] as? DatabaseReference)!
        self.ImageUrl = dictionary["ImageUrl"] as? String
    }
```



```
init(snapshot : DataSnapshot)
{

    ref = snapshot.ref
    if let value = snapshot.value as? [String : Any] {
        text = value["text"] as! String
        name = value["name"] as! String
        uid = value["uid"] as! String
        ImageUrl = value["ImageUrl"] as! String

    }
}
```



```
florapedia-277ea
  notes
    5ZVUBFQJ14V8mkIK5ETfDqmfWwj2
      -LBYjltTrZaubkZUJMGM
        ImageUrl: "https://firebasestorage.googleapis.com/v0/b/flo..."
        name: "7i5ygik75g7"
        text: "Whats on your mind?"
        uid: "5ZVUBFQJ14V8mkIK5ETfDqmfWwj2"
      -LBmFosofi45HvkfWPPv
        ImageUrl: "https://firebasestorage.googleapis.com/v0/b/flo..."
        name: "yo yo"
        text: "What's up!!"
        uid: "5ZVUBFQJ14V8mkIK5ETfDqmfWwj2"
    YqGkVpTgdaagC7HikoNSvXcA3Z82
    elQdzfAQWLhfqsnsdFUTL4l78I72
```

In its view did load method, the notes object is checked, if the notes object is not nil that means that user wants to see the details of the note corresponding to the cell in journal view. In this case, the save button is hidden as it is not necessary. Also, the image view is made clickable by adding the gesturerecognizer.

```
override func viewDidLoad() {
        super.viewDidLoad()
        if(self.notes != nil)
        {
            NoteName.text = self.notes?.name
            
            if let ImageUrl = self.notes?.ImageUrl {
                flowerPostimage?.loadImageUsingCacheWithUrlString(ImageUrl)
            }
            flowerPosttext.text = self.notes?.text
            flowerPostsavebuton.isHidden = true
        }
        flowerPostimage.addGestureRecognizer(UITapGestureRecognizer(target: self, action: 
#selector(handleSelectProfileImageView)))
        flowerPostimage.isUserInteractionEnabled = true
        
    }
```

Hide keyboard

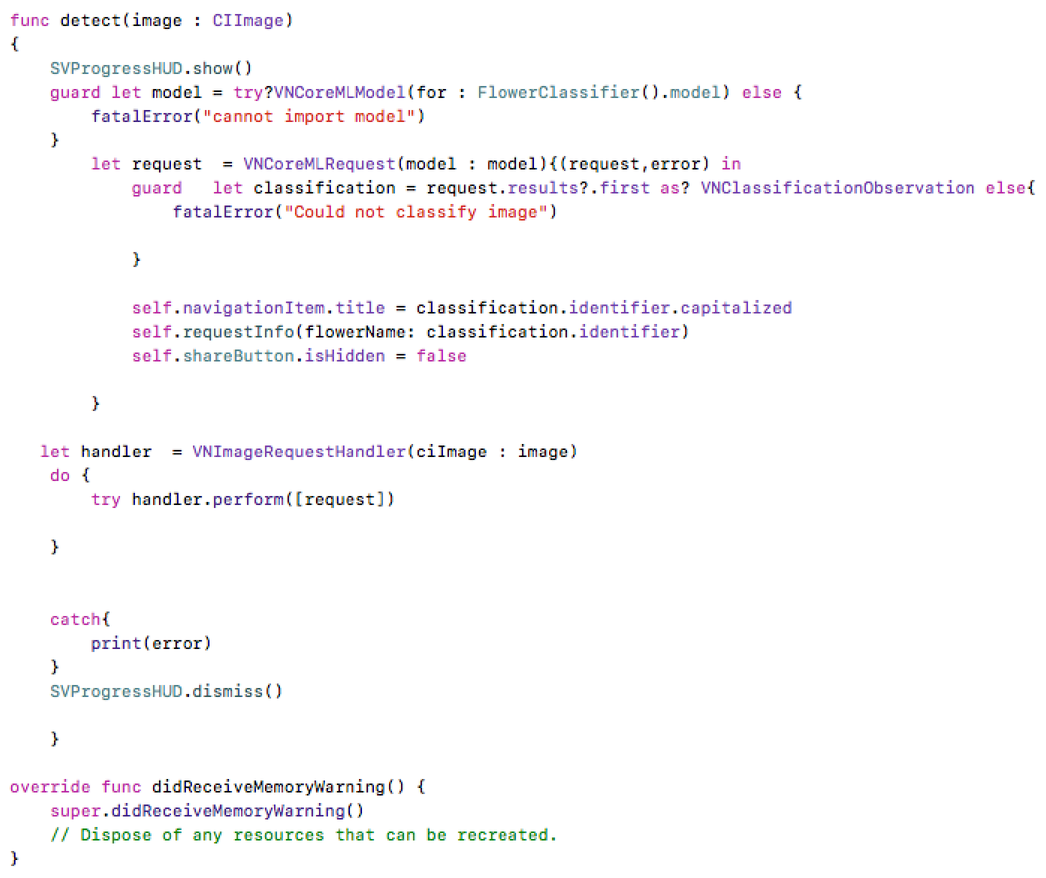
When the user taps on textfield then keyboard appears on screen and when the user taps away then the keyboard is hidden. The code snippet below displays this functionality .It adds to enriched user experience

```
extension UIViewController
{
    func hideKeyboard()
    {
        let tap: UITapGestureRecognizer = UITapGestureRecognizer(
            target: self,
            action: #selector(UIViewController.dismissKeyboard))
        tap.cancelsTouchesInView = false
        view.addGestureRecognizer(tap)
    }

    @objc func dismissKeyboard()
    {
        view.endEditing(true)
    }
}
```

EncyclopediaViewController

This controller enables user to click a picture of a flower and the name and description og the flower from Wikipedia is shown to the user. The controller imports,CoreML,Vision which are machine learning frameworks. Also, pods alamofire and swiftyjson are used to parse the json object and fetch data from Wikipedia api. The flowerclassifier is a pre trained model which is included in the project. This is basically a caffe model which is converted to a .ml model as xcode requires only .ml model.

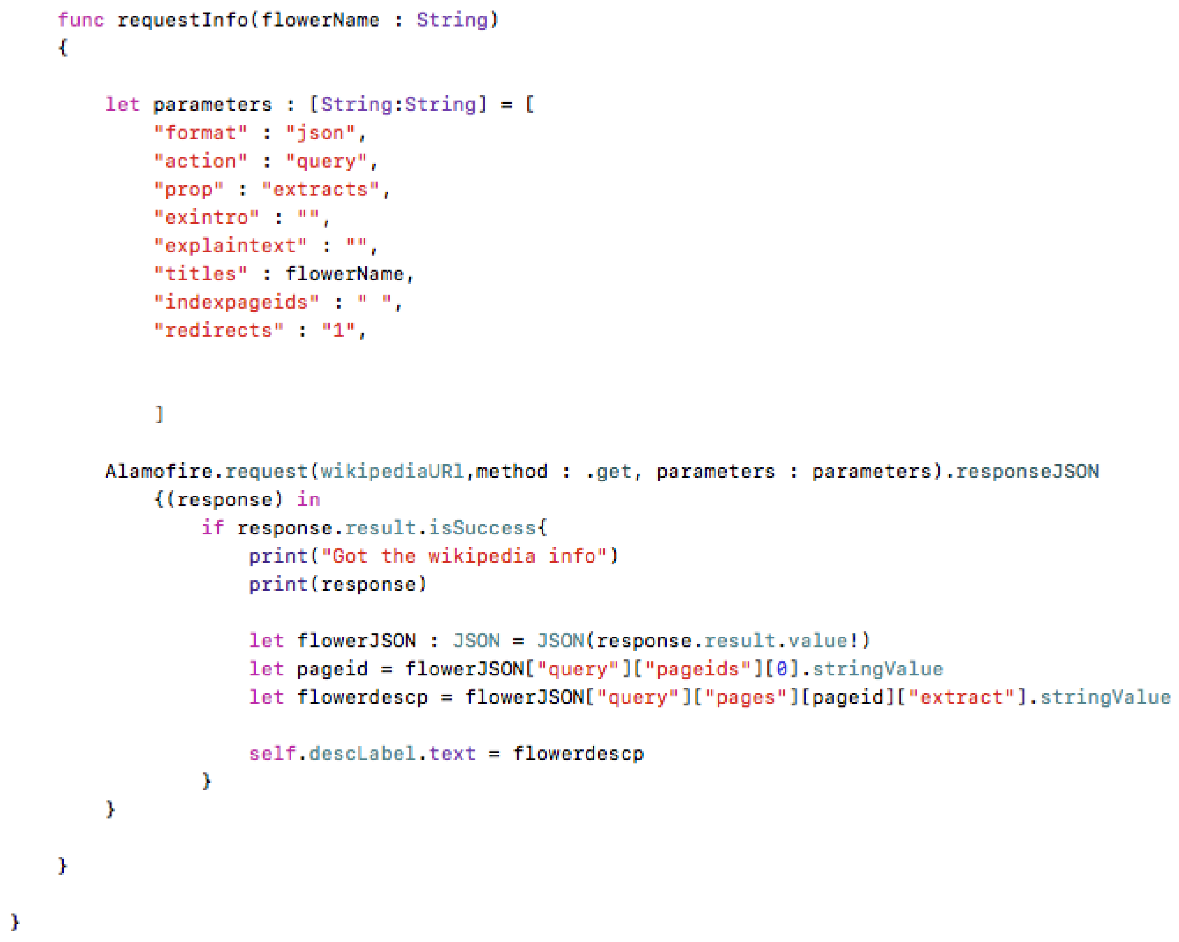
The UIimage clicked by the user is converted to CIimage and then processed by the flower classifier. The handler processes the request which is of type VNCoreMLmodel. The result of the classification is stored in the variable and casted as string to identify the name of the flower.



```
func detect(image : CIImage)
{
    SVProgressHUD.show()
    guard let model = try?VNCoreMLModel(for : FlowerClassifier().model) else {
        fatalError("cannot import model")
    }
        let request  = VNCoreMLRequest(model : model){(request,error) in
            guard   let classification = request.results?.first as? VNClassificationObservation else{
                fatalError("Could not classify image")

            }

            self.navigationItem.title = classification.identifier.capitalized
            self.requestInfo(flowerName: classification.identifier)
            self.shareButton.isHidden = false

        }

   let handler  = VNImageRequestHandler(ciImage : image)
    do {
        try handler.perform([request])

    }


    catch{
        print(error)
    }
    SVProgressHUD.dismiss()

    }

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}
```

In the requestinfo method, data is fetched from the Wikipedia url
let wikipediaURl = https://en.wikipedia.org/w/api.php

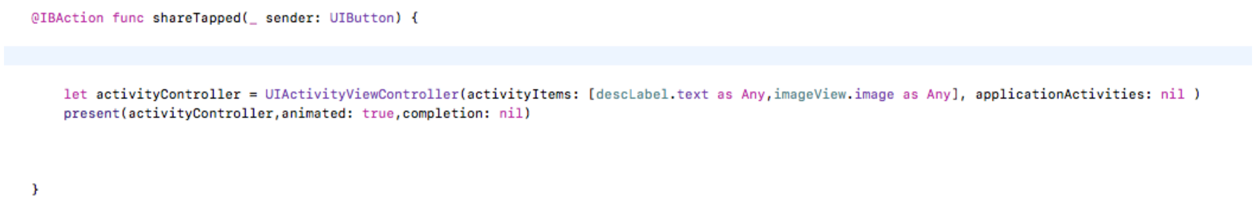
The Alamofire pod has a request method where the parameters are passed to the wiki api.The flower name identified by the classifer is sent as the “title” in the parameter list
The retrieved JSON response is parsed using swifty JSON and casted to string value and the retrieved extract(information about the flower) is populated in the textlabel on the screen



```
    func requestInfo(flowerName : String)
    {

        let parameters : [String:String] = [
            "format" : "json",
            "action" : "query",
            "prop" : "extracts",
            "exintro" : "",
            "explaintext" : "",
            "titles" : flowerName,
            "indexpageids" : " ",
            "redirects" : "1",


            ]

        Alamofire.request(wikipediaURl,method : .get, parameters : parameters).responseJSON
            {(response) in
                if response.result.isSuccess{
                    print("Got the wikipedia info")
                    print(response)

                    let flowerJSON : JSON = JSON(response.result.value!)
                    let pageid = flowerJSON["query"]["pageids"][0].stringValue
                    let flowerdescp = flowerJSON["query"]["pages"][pageid]["extract"].stringValue

                    self.descLabel.text = flowerdescp
                }
        }

    }

}
```

Sharing

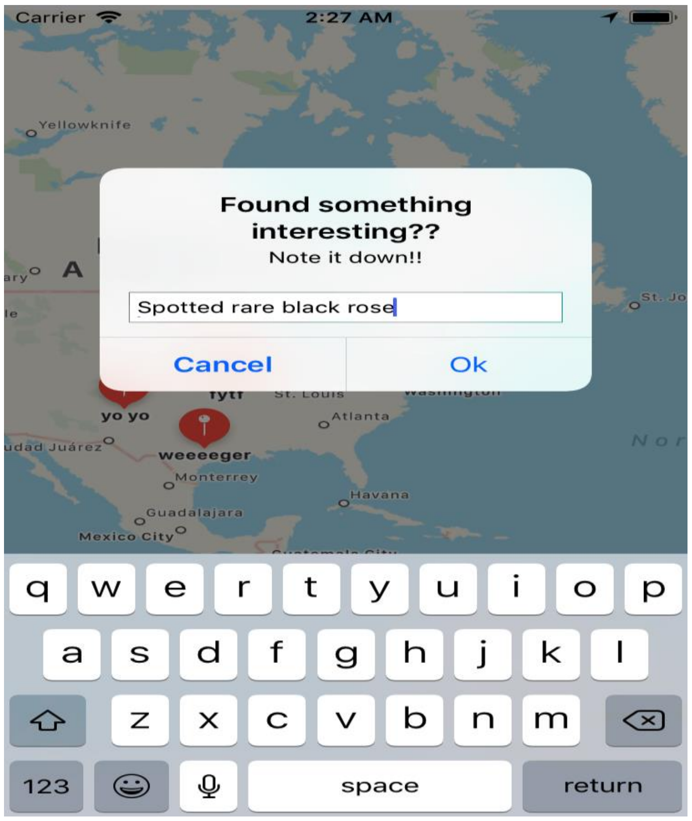
The image clicked and identified can be shared to facebook or twitter or just airdropped to other MAC users. The users can add additional text to elaborate the image

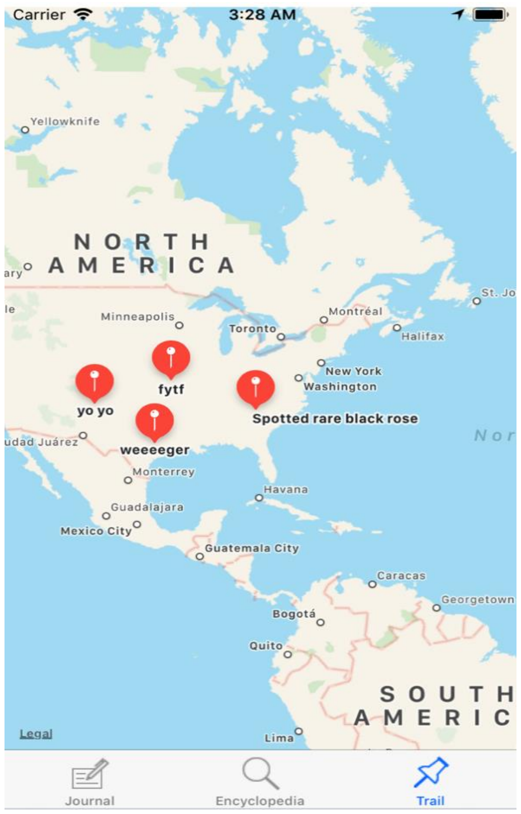


```
    @IBAction func shareTapped(_ sender: UIButton) {


        let activityController = UIActivityViewController(activityItems: [descLabel.text as Any,imageView.image as Any], applicationActivities: nil )
        present(activityController,animated: true,completion: nil)


    }
```

MapViewController

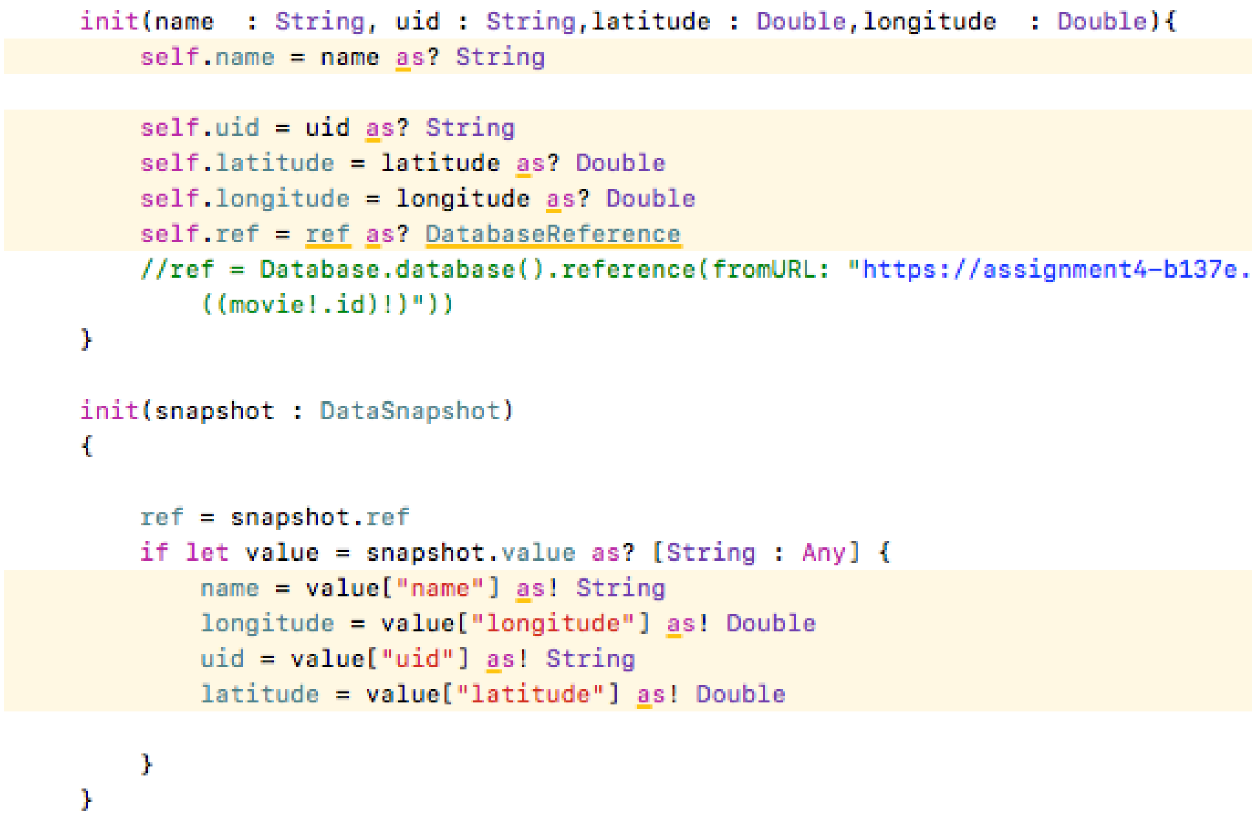
In this view, user can put annotation on the map with a custom title. The details of the annotation title, latitude and longitude details are saved on firebase and whenever the user logs in the previous annotations are shown to the user. The annotations only of that particular user are displayed on that map. This feature allows user to keep track of the interesting destinations visited.





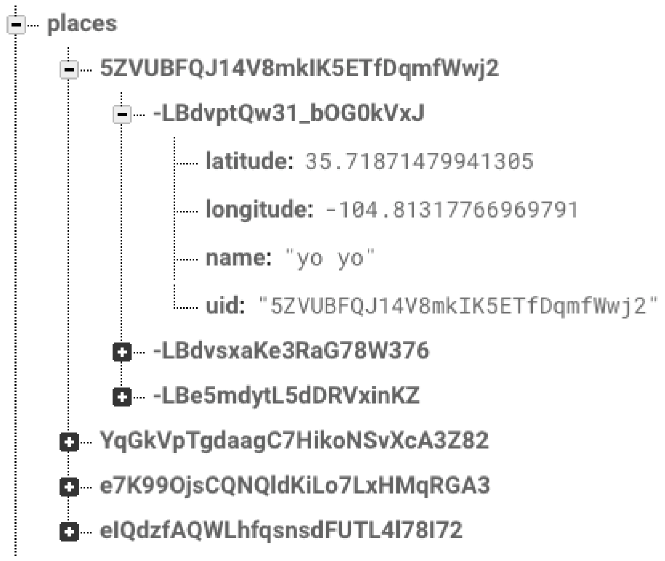
The below code snippets show the structure of the model created.

```
class LocationAnnotation  : NSObject {
    var uid: String?
    var latitude : Double?
    var longitude : Double?
    var name : String?
    var ref : DatabaseReference?
```



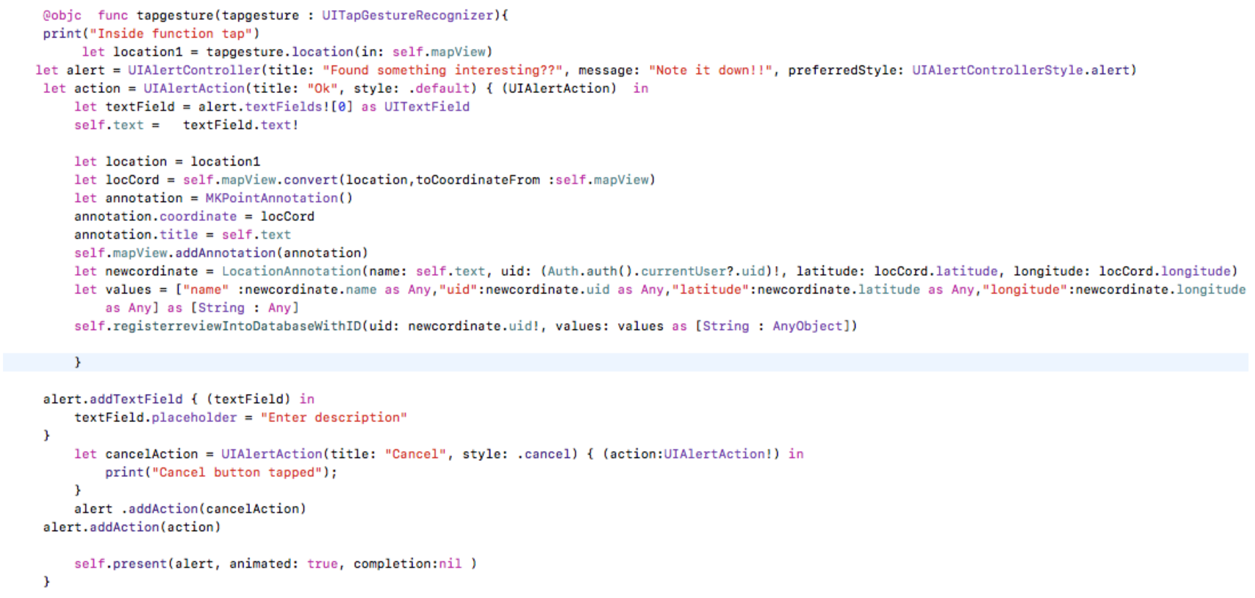
```
init(name  : String, uid : String,latitude : Double,longitude  : Double){
    self.name = name as? String

    self.uid = uid as? String
    self.latitude = latitude as? Double
    self.longitude = longitude as? Double
    self.ref = ref as? DatabaseReference
    //ref = Database.database().reference(fromURL: "https://assignment4-b137e.
        ((movie!.id)!)"))
}

init(snapshot : DataSnapshot)
{

    ref = snapshot.ref
    if let value = snapshot.value as? [String : Any] {
        name = value["name"] as! String
        longitude = value["longitude"] as! Double
        uid = value["uid"] as! String
        latitude = value["latitude"] as! Double

    }
}
```

Below is the screenshot of the database structure of the annotation stored in database.



```
places
    5ZVUBFQJ14V8mkIK5ETfDqmfWwj2
        -LBdvptQw31_bOG0kVxJ
            latitude: 35.71871479941305
            longitude: -104.81317766969791
            name: "yo yo"
            uid: "5ZVUBFQJ14V8mkIK5ETfDqmfWwj2"
        -LBdvsxaKe3RaG78W376
        -LBe5mdytL5dDRVxinKZ
    YqGkVpTgdaagC7HikoNSvXcA3Z82
    e7K99OjsCQNQldKiLo7LxHMqRGA3
    elQdzfAQWLhfqsnsdFUTL4l78l72
```

Whenever the map is tapped, the gesturerecognizer presents the user with a alert to enter details of the title and upon clicking ok, the data is added to database and an annotation is seen on the map.



```
    @objc  func tapgesture(tapgesture : UITapGestureRecognizer){
    print("Inside function tap")
         let location1 = tapgesture.location(in: self.mapView)
   let alert = UIAlertController(title: "Found something interesting??", message: "Note it down!!", preferredStyle: UIAlertControllerStyle.alert)
    let action = UIAlertAction(title: "Ok", style: .default) { (UIAlertAction)  in
        let textField = alert.textFields![0] as UITextField
        self.text =   textField.text!

        let location = location1
        let locCord = self.mapView.convert(location,toCoordinateFrom :self.mapView)
        let annotation = MKPointAnnotation()
        annotation.coordinate = locCord
        annotation.title = self.text
        self.mapView.addAnnotation(annotation)
        let newcordinate = LocationAnnotation(name: self.text, uid: (Auth.auth().currentUser?.uid)!, latitude: locCord.latitude, longitude: locCord.longitude)
        let values = ["name" :newcordinate.name as Any,"uid":newcordinate.uid as Any,"latitude":newcordinate.latitude as Any,"longitude":newcordinate.longitude
            as Any] as [String : Any]
        self.registerreviewIntoDatabaseWithID(uid: newcordinate.uid!, values: values as [String : AnyObject])

        }

    alert.addTextField { (textField) in
        textField.placeholder = "Enter description"
    }
        let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) { (action:UIAlertAction!) in
            print("Cancel button tapped");
        }
        alert .addAction(cancelAction)
    alert.addAction(action)

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

Autolayout

I have used storyboard to design the user interface. I have tried to incorporate appealing color schemes while designing the UI. All the UI elements are embedded in a stack so as to retain their form in various screens. I have set the autolayout and margin constraints to the safe areas of the view. The margin constraints make the view appear proportionately in all screens.

The ui components are added in stack to give a added layer of control to setting margin constraints on the screen.

Future Scope

1) As a part of further refining the app, I plan to add more classifiers so as to increase the scope of identification. Also, some additional apis can be incorporated to give user more information about gardening/plants etc
2) I also plan to add a Q&A portal to the app where users can post their questions and other users can solve the queries and rate the answers.

References

1) https://www.youtube.com/channel/UCuP2vJ6kRutQBfRmdcI92mA
2) https://stackoverflow.com/questions/32281651/how-to-dismiss-keyboard-when-touching-anywhere-outside-uitextfield-in-swift
3) https://github.com/Alamofire/Alamofire