Christopher Su
CSC680

# Project Documentation: Vacation Finder

## Overview:

This project takes in user input and displays the location on a map and 3 links related to the location, google events happening in the area, trails around the area, and a travel guide of the area. It does this by first taking in user input, then using an API call to autocomplete the location and find the closest match. It then calls a second API call to get specific information such as map location and information URLS that it then displays on the screen. If there is no result for the user input, an error message is an output, and the map auto displays San Francisco.

## Constraints:

I was unable to get the information URLS to work as hyperlinks, so it is currently planned as a future feature.

For some reason, searching san Francisco shows an invalid input on the API itself (error also shows on postman), when demoing please search for a different city.

## Code Walkthrough:

View Controller:
Simply takes in input and when the button is pressed, sends the location field to the infoviewcontroller

```swift
//
//  ViewController.swift
//  TrailTrail
//
//  Created by Christopher Su on 5/5/22.
//

import UIKit

class ViewController: UIViewController {

    let networking = Networking()

    @IBOutlet weak var location: UITextField!

    @IBAction func button(_ sender: UIButton) {
        let userInput: String? = location.text
        if let stringInput = userInput {
            print(stringInput)
        }

    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }



    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
        {
        guard let InfoViewController = segue.destination as?
            InfoViewController else {
            return
        }
        InfoViewController.location = location.text
        print(segue.destination)
    }
}
```

InfoViewController:

Takes the location input from the ViewController and uses a network call to get the information about the location. Then calls update() to update the information on the screen.

```swift
import UIKit
import MapKit

class InfoViewController: UIViewController {

    // this is for the networking call
    let networking = Networking()
    // map creations
    let map = Map()
    // this is what contains the info
    @IBOutlet weak var events: UILabel!
    @IBOutlet weak var trails: UILabel!
    @IBOutlet weak var guide: UILabel!

    var attributes = AttributesSpec(name:"",longitude: 0,latitude:
        0,slug:"",google_events_url: "",alltrails_url: "", url: "")
    // this is from the segue
    var location: String?

    @IBOutlet weak var mapView: MKMapView!

    @IBOutlet weak var name: UILabel!
    @IBAction func back(_ sender: Any) {
        dismiss(animated: true, completion: nil)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.

        guard let location = location else {
            return
        }
        // set default region to sf
        mapView.setRegion(map.createDefaultMap(), animated: true)

        networking.fetchInfo( callback: { info2 in
            self.update(with: info2)
        }, location:location)
    }
```

```swift
    func update(with info2: SpecDati) {
        DispatchQueue.main.async {
            // remember to error check
            if(info2.id != "0") {
                // set the data if there is no errors!
                self.attributes = info2.attributes

                // set the label
                self.name.text = self.attributes.name
                self.events.text = "Events: " +
                    self.attributes.google_events_url
                self.trails.text = "Trails: " +
                    self.attributes.alltrails_url
                self.guide.text = "Travel Guide: " +
                    self.attributes.url

                // set the map
                self.mapView.setRegion(self.map.createMap(location:
                    self.attributes), animated: true)
                self.mapView.addAnnotation(self.map.makePin(location:
                    self.attributes))
            } else {
                self.name.text = "INVALID INPUT :( TRY AGAIN!!"
                print("Did not find a location")
            }
        }
    }


}
```

Networking:
Networking handles all of the network calls. It first builds a URL request with Basic Auth and then uses the information got back to make a second API call that gets the information. It then returns a Dati Object.

```swift
// This is the networking call
struct Networking {

    let baseURLString = "https://api.roadgoat.com/api/v2/destinations/auto_complete?q="

    let baseURLStringCall2 = "https://api.roadgoat.com/api/v2/destinations/"


    func fetchInfo(callback: @escaping (SpecDati) -> (),location: String) {
        // This would block user interactons, we want to do this on a background queue.

        // build the url based on user input
        let loc = location.lowercased().replacingOccurrences(of: " ", with: "-")
        // Make the url based on the input
        let url = URL(string: "\(baseURLString)\(loc)")!
        var request = URLRequest(url: url)
        let authData = ("d3171df05b97abf67b97a1ff1d7483f1" + ":" + "e149c445718433c51c2ffd49c7de77b4").data(using:
            .utf8)!.base64EncodedString()
        request.httpMethod = "GET"
        request.addValue("Basic \(authData)", forHTTPHeaderField: "Authorization")

        // make api call to get auto-complete responses to the input
        let task = URLSession.shared.dataTask(with: request) { maybeData, maybeResponse, maybeError in
            guard let data: Data = maybeData else {
                print("ERROR")
                return
            }

            // decode the response
            let decoder = JSONDecoder()
            do {
                let response = try decoder.decode(Info.self, from: data)
                let Info: Info = response

                if(Info.data.count > 0) {
                    // network call 2 based on the slug from the first network call that we got back
                    let url2 = URL(string: "\(baseURLStringCall2)\(Info.data[0].attributes.slug)")!
                    var request2 = URLRequest(url: url2)
                    let authData = ("d3171df05b97abf67b97a1ff1d7483f1" + ":" +
                        "e149c445718433c51c2ffd49c7de77b4").data(using: .utf8)!.base64EncodedString()
                    request2.httpMethod = "GET"
                    request2.addValue("Basic \(authData)", forHTTPHeaderField: "Authorization")
                    print(url2)
```

```
                let task2 = URLSession.shared.dataTask(with: request2) { maybeData, maybeResponse, maybeError in
                    print("TASK IS RNA")
                    guard let data: Data = maybeData else {
                        print("ERROR")
                        return
                    }
        //            print(String(data: data, encoding: .utf8))
                    let decoder = JSONDecoder()
                    do {
                        let response = try decoder.decode(Spec.self, from: data)
                        print("SUCCESS for 2!")
                        let Spec: Spec = response

                        print("NETWORK Spec ")
                        print(Spec.data.attributes.name)


                        callback(Spec.data)
                    } catch {
                        print("THERE WAS EROORRO")
                    }
                }
                task2.resume()
            } else {
                let errorAtt = AttributesSpec(name:"",longitude: 0,latitude: 0,slug:"",google_events_url:
                    "",alltrails_url: "",url:"")
                let error = SpecDati(id:"0", attributes: errorAtt)
                callback(error)
            }


        } catch {

        }
    }
    task.resume()
}
```

Map:
Map handles everything map-related. When called, it will return a coordinate for the map to be set based on user input.

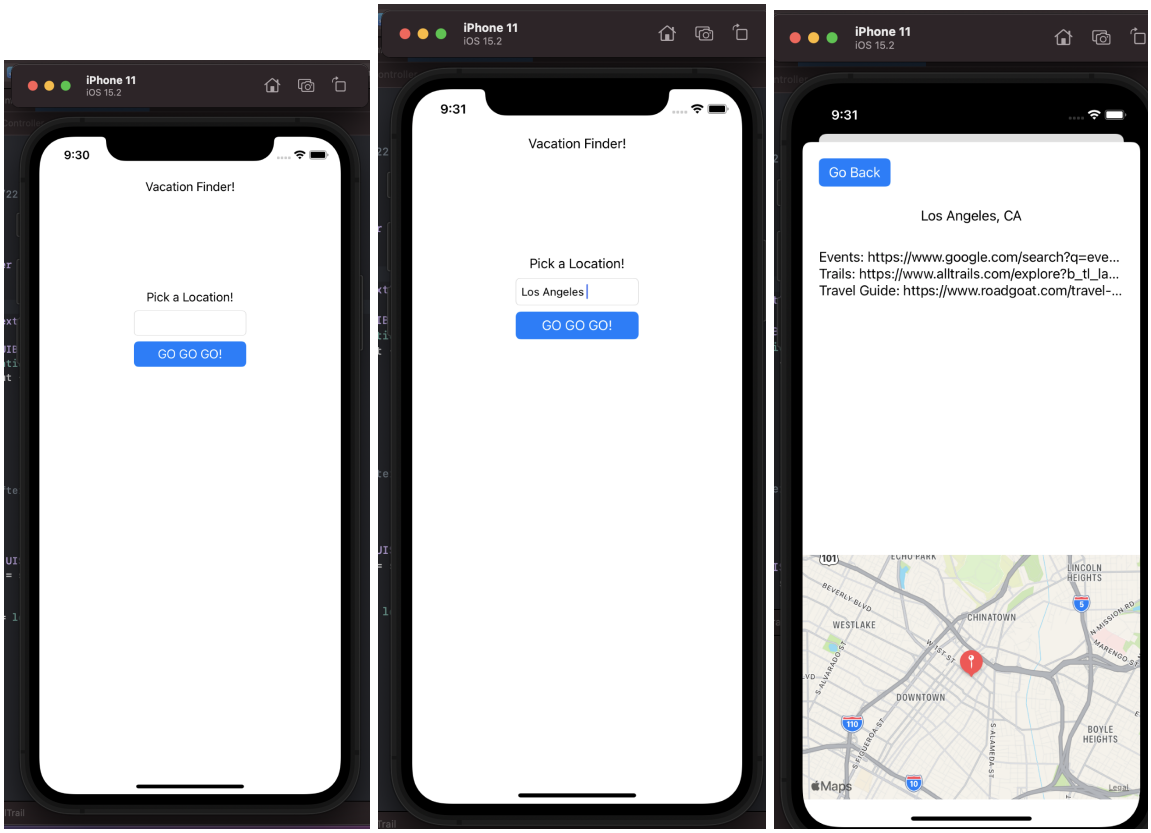
Info:
This is where all the information structs are that are filled w/ information from API calls.

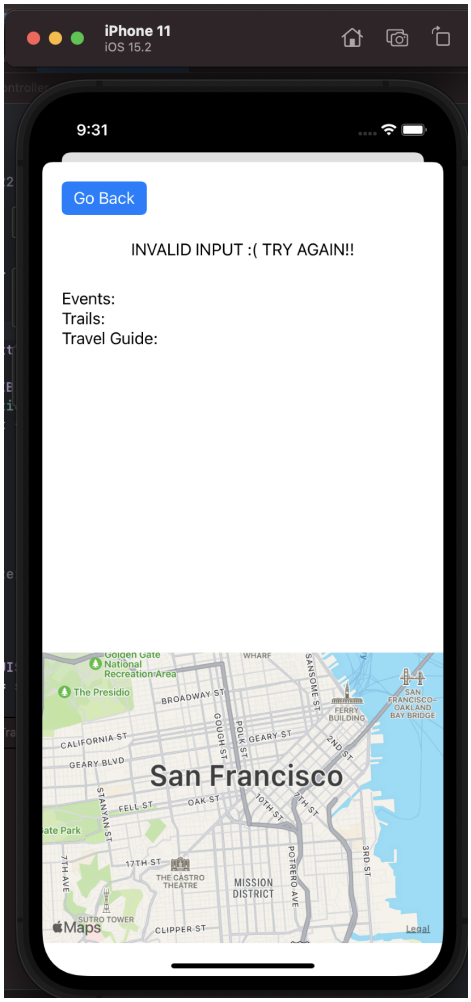
Info, Dati, Attributes are for the first API call

Spec, SpecDati, AttributesSpec are for the second API call

```swift

import Foundation

// this is for first api call, just getting autocomplete information
struct Info: Codable {
    var data: [Dati]
}

struct Dati: Codable {
    var id: String
    var attributes: Attributes
//    var attributes: Attributes
}

struct Attributes: Codable {
    var slug: String
}


// This is for the specific location (2nd api call)
struct Spec: Codable {
    var data: SpecDati
}

struct SpecDati: Codable {
    var id: String
    var attributes: AttributesSpec
}

struct AttributesSpec: Codable {
    let name: String
    var longitude: Double
    var latitude: Double
    var slug: String
    var google_events_url: String
    var alltrails_url: String
    var url: String
}

```

## Application Demo:




iPhone 11
iOS 15.2
9:31
Go Back
INVALID INPUT :( TRY AGAIN!!
Events:
Trails:
Travel Guide:
San Francisco
Maps
Legal