Assignment 2 - Getting the User’s Actual Location

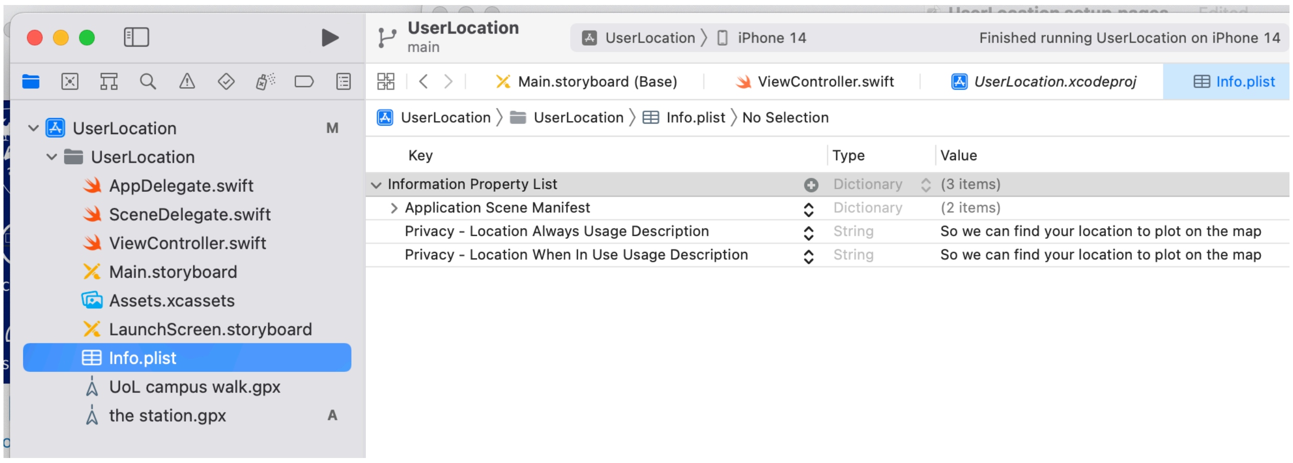
For this assignment, I’ve suggested that you order the sections in your table, by distance from the user - an activity that is standard for an App that displays location based items. But how do you determine the user’s location (and how, in particular, do we do it in the simulator)?

In your App, make sure that your table and map are installed on your storyboard for your main view. Make sure that you control drag from the table to the View Controller yellow icon, and make a datasource and delegate. Do the same from the map (as a delegate).

You’ll also need to control drag from the table and from the map to your view controller source and make an outlet for each (myTable and myMap).

Apps can’t just know where the user is located (it’s a potential intrusion on the user’s privacy), and so we have to request that the user allows iOS to tell us.

Select the info.plist from the list of files on the left hand side. You will need to add two additional items to it. Hover over the up/down arrows by the final item, click the plus and then select “Privacy - Location When in Use Usage Description” (if you type “Privacy” the list will scroll to those) and repeat this with “Privacy - Location Always Usage Description”. Add a meaningful string describing why you need each of these.



- UserLocation M
  - UserLocation
    - AppDelegate.swift
    - SceneDelegate.swift
    - ViewController.swift
    - Main.storyboard
    - Assets.xcassets
    - LaunchScreen.storyboard
    - Info.plist
    - UoL campus walk.gpx
    - the station.gpx A

Key	Type	Value
Information Property List	Dictionary	(3 items)
Application Scene Manifest	Dictionary	(2 items)
Privacy - Location Always Usage Description	String	So we can find your location to plot on the map
Privacy - Location When In Use Usage Description	String	So we can find your location to plot on the map

Now we need to update the ViewController source to add protocols for the various types of delegation etc. that we will be adopting. Update your ViewController.swift file to show the following (you can find the full source code later in this document):

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate, MKMapViewDelegate,
    CLLocationManagerDelegate {
```

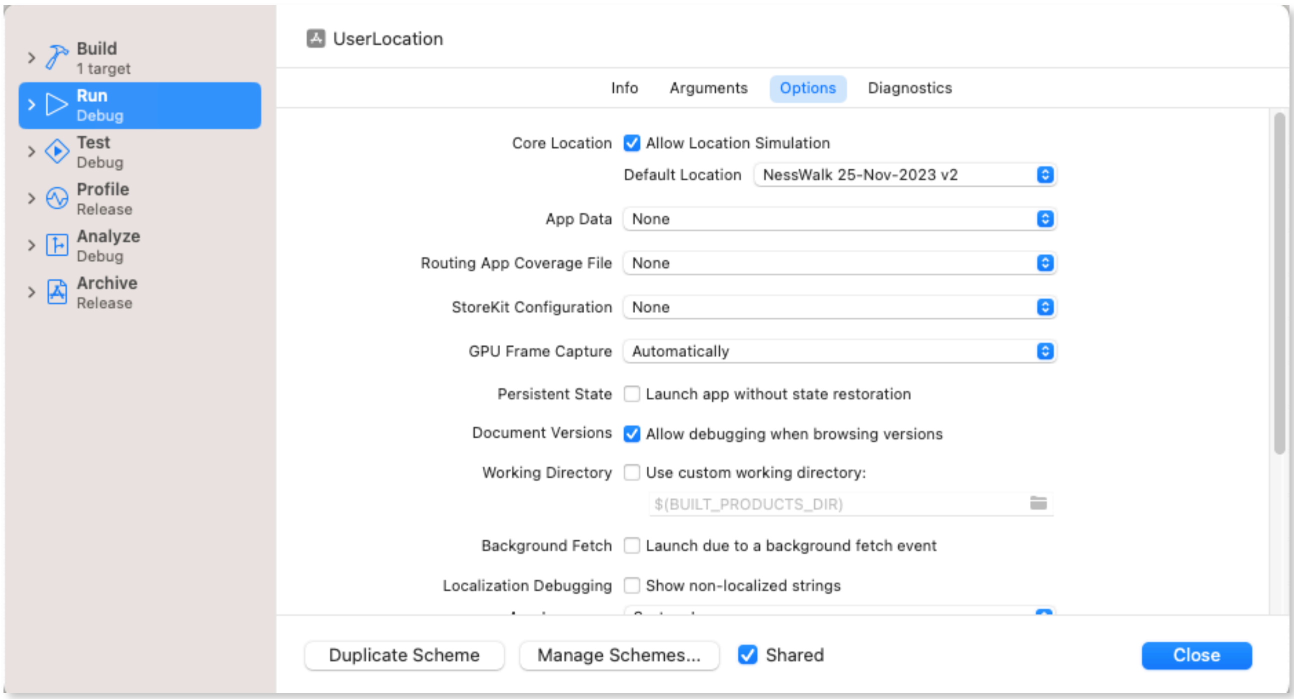
As usual Xcode will alert you to the fact that you need to provide some methods for these - let it create the stubs for you. You’ll need, of course, to supply the code (or use mine - see later in this document).

We want to get CoreLocation to do stuff for us, so we add CoreLocation to the import list. We also import the “NessWalk 25-Nov-2023 v2” document - drag it into the left hand side file list and add it to your project. Make sure that the “copy item if needed” button is selected. It’s a gpx file which contains location based information.

We need to setup CoreLocation so that it will start sending us location messages when the user moves. We have to write some code to create a locationManager instance, and then set it up with some reasonable default values. We then have to request the user to let us get their location, and if they confirm it, we will get location messages passed to us (calling a method that we have to supply). Here's my full code from ViewController.swift.

```
//
//  ViewController.swift
//  UserLocation
//
//  Created by Phil Jimmieson on 20/11/2022.
//

import UIKit
import MapKit
import CoreLocation

class ViewController: UIViewController,UITableViewDataSource, UITableViewDelegate,
MKMapViewDelegate, CLLocationManagerDelegate {

    // MARK: Map & Location related stuff

    @IBOutlet weak var myMap: MKMapView!

    var locationManager = CLLocationManager()
    var firstRun = true
    var startTrackingTheUser = false

    func locationManager(_ manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {
        let locationOfUser = locations[0] //this method returns an array of locations
        //generally we always want the first one (usually there's only 1 anyway)
        let latitude = locationOfUser.coordinate.latitude
        let longitude = locationOfUser.coordinate.longitude
        //get the users location (latitude & longitude)
        let location = CLLocationCoordinate2D(latitude: latitude, longitude: longitude)

        if firstRun {
            firstRun = false
            let latDelta: CLLocationDegrees = 0.0025
            let lonDelta: CLLocationDegrees = 0.0025
            //a span defines how large an area is depicted on the map.
            let span = MKCoordinateSpan(latitudeDelta: latDelta, longitudeDelta: lonDelta)

            //a region defines a centre and a size of area covered.
            let region = MKCoordinateRegion(center: location, span: span)

            //make the map show that region we just defined.
            self.myMap.setRegion(region, animated: true)

            //the following code is to prevent a bug which affects the zooming of the map to the
user's location.
            //We have to leave a little time after our initial setting of the map's location and
span,
            //before we can start centering on the user's location, otherwise the map never zooms in
because the
            //intial zoom level and span are applied to the setCenter( ) method call, rather than
our "requested" ones,
            //once they have taken effect on the map.

            //we setup a timer to set our boolean to true in 5 seconds.
            _ = Timer.scheduledTimer(timeInterval: 5.0, target: self, selector:
#selector(startUserTracking), userInfo: nil, repeats: false)
        }

        if startTrackingTheUser == true {
            myMap.setCenter(location, animated: true)
        }
    }

    //this method sets the startTrackingTheUser boolean class property to true. Once it's true,
   //subsequent calls to didUpdateLocations will cause the map to centre on the user's location.
    @objc func startUserTracking() {
        startTrackingTheUser = true
    }

    //MARK: Table related stuff

    @IBOutlet weak var theTable: UITableView!
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 10
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "myCell", for: indexPath)
        var content = UIListContentConfiguration.subtitleCell()
        content.text = "testing"
        content.secondaryText = "more testing"
        cell.contentConfiguration = content
        return cell
    }

    // MARK: View related Stuff

    override func viewDidLoad() {
        super.viewDidLoad()
        // Make this view controller a delegate of the Location Manager, so that it
        //is able to call functions provided in this view controller.
        locationManager.delegate = self as CLLocationManagerDelegate

        //set the level of accuracy for the user's location.
        locationManager.desiredAccuracy = kCLLocationAccuracyBestForNavigation

        //Ask the location manager to request authorisation from the user. Note that this
        //only happens once if the user selects the "when in use" option. If the user
        //denies access, then your app will not be provided with details of the user's
        //location.
        locationManager.requestWhenInUseAuthorization()

        //Once the user's location is being provided then ask for updates when the user
        //moves around.
        locationManager.startUpdatingLocation()

        //configure the map to show the user's location (with a blue dot).
        myMap.showsUserLocation = true
    }
}
```

Finally, you should alt-click on the name of the simulator you're about to run this on - this will open a panel where you can configure various run time options. Select the gpx file (in this case, “Nesswalk 25-Nov-2023 v2”) that you want to use for the default location (this can be overridden in the simulator from a menu - hence why it's called the default).

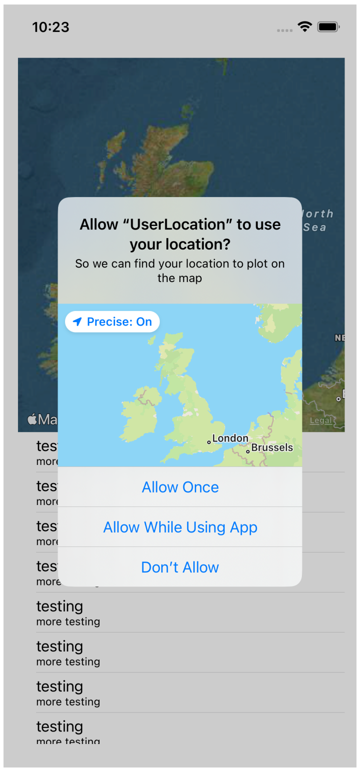


UserLocation

Core Location: [✓] Allow Location Simulation
Default Location: NessWalk 25-Nov-2023 v2
App Data: None
Routing App Coverage File: None
StoreKit Configuration: None
GPU Frame Capture: Automatically
Persistent State: [ ] Launch app without state restoration
Document Versions: [✓] Allow debugging when browsing versions
Working Directory: [ ] Use custom working directory:
$(BUILT_PRODUCTS_DIR)
Background Fetch: [ ] Launch due to a background fetch event
Localization Debugging: [ ] Show non-localized strings

[✓] Shared

You should now be able to run this App in the simulator and get prompted to permit it to know your location.


10:23

Allow "UserLocation" to use your location?

So we can find your location to plot on the map

testing
more testing

testing
more testing

testing
more testing

testing
more testing

If you click allow, the map should zoom in to the the Ness Gardens visitor centre, from where my walk started.