# CSCI 342 Group Project

## Follow My Foot Steps

### Design Documents and Explanations

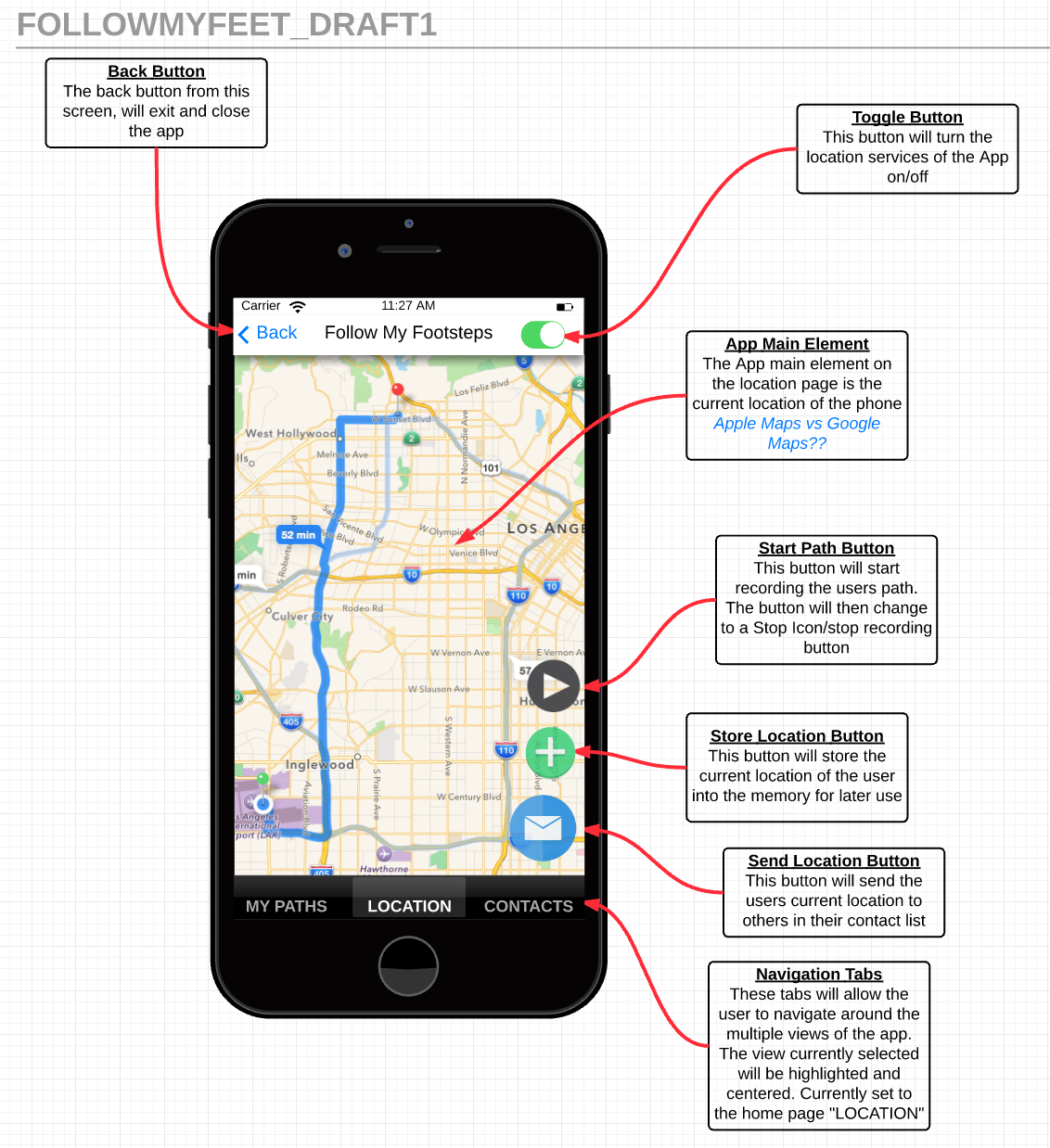
### Introduction

Follow My Footsteps is the first part of a larger travelling companion application targeted at backpackers. Essentially it is a modified map application with a focus on the route and point of interest functions of other map apps, the idea stems from an old backpacker’s trend where travellers leaving a city would pass on their paper maps onto new arrivals that had different paths and POIs than what would be found in a travel guide. With the increase of mobile technology this trend has all but died so Follow My Footsteps looks to rectify this, users will be able to “record” their days and are capable of marking new POIs when they have finished they are able to pass on this route to others with the app installed. Another function of Follow My Footsteps is the ability to send your current location to other users, this function will come in handy when the users are in new cities and wish to direct their friends to their location.

### Conceptual Model

The conceptual model is of the first screen of the application (Fig 1). Within the Navigation Bar there would be two elements excluding the name of the app, the first, is a simple back button that would exit the app. The second is a toggle button that enabled the user to switch the location services of the app on and off, this would a) allow the user to conserve battery power, b) prevent the excessive mobile data fees whilst traveling abroad and c) for the paranoid user gives them full control of when they can be found.

The main element of the app is a map view, using iOS’s MapKit. Overlaid on the map view is three buttons, the top will begin “recording” the users path which when pressed will morph into a stop button. The second button, store, will store the user’s current location, this will work both when “recording” and when not allowing the user to not only pass on entire paths but individual locations that another user can then use, possibly getting there a different way from the previous user. The send button when pressed will open up the stored data table view, with “Current Location” pinned to the top, the user can then choose to send either their current location, paths they have recorded or received from others as well as any other POIs they have saved.

Fig 1

Along the bottom of the view is the in app navigation panel, on default it will open onto the map view. Pressing the My Paths button will open all the stored data that the user has created. The contacts button will open the users contacts.

### Design Steps

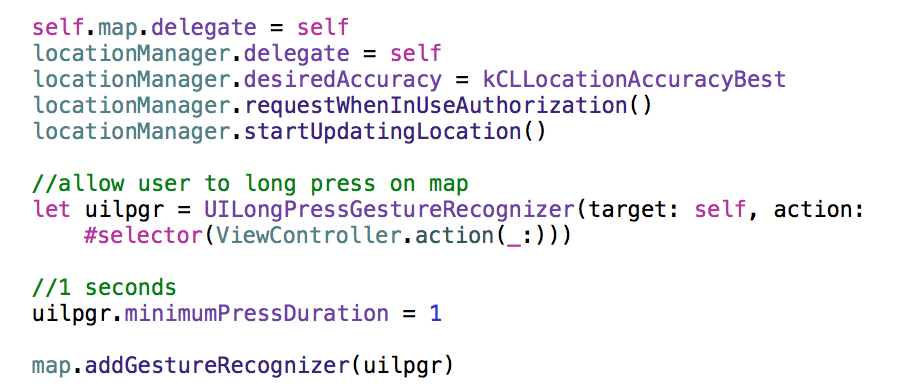
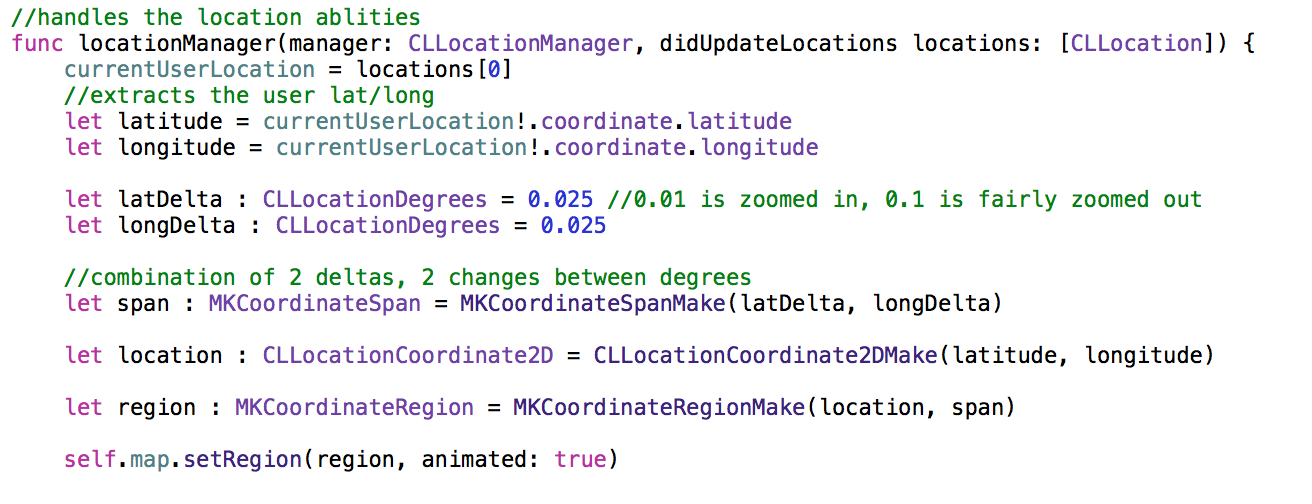
First the map view needed to be implemented, there was a discussion of using Google Maps SDK and API, however, due to time constraints iOS’s MapKit was chosen for now. Below is the code used to set the delegate as well the locations accuracy and the authorization request. Code was also added that allowed the user to long press on the map to drop an annotation.

Fig 2

The function locationManager was then created to extract the user’s current location

Fig 3

Once it was possible to find the user and add annotations, the ability to save these locations was addressed. When the user long pressed on the map a prompt opens allowing them to add a name as well as a description. Once the information was filled pressing ‘Create Location’ will save the location within CoreData.

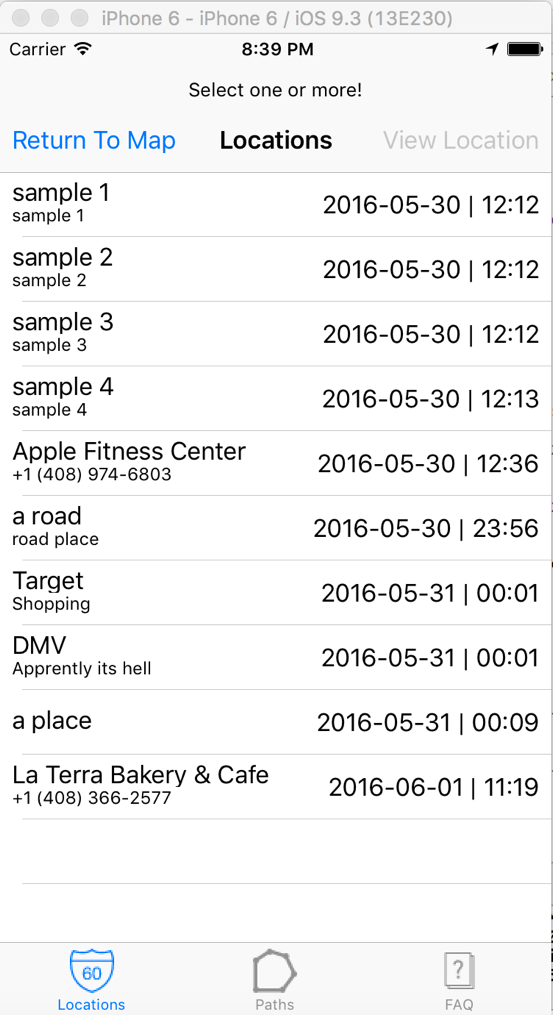
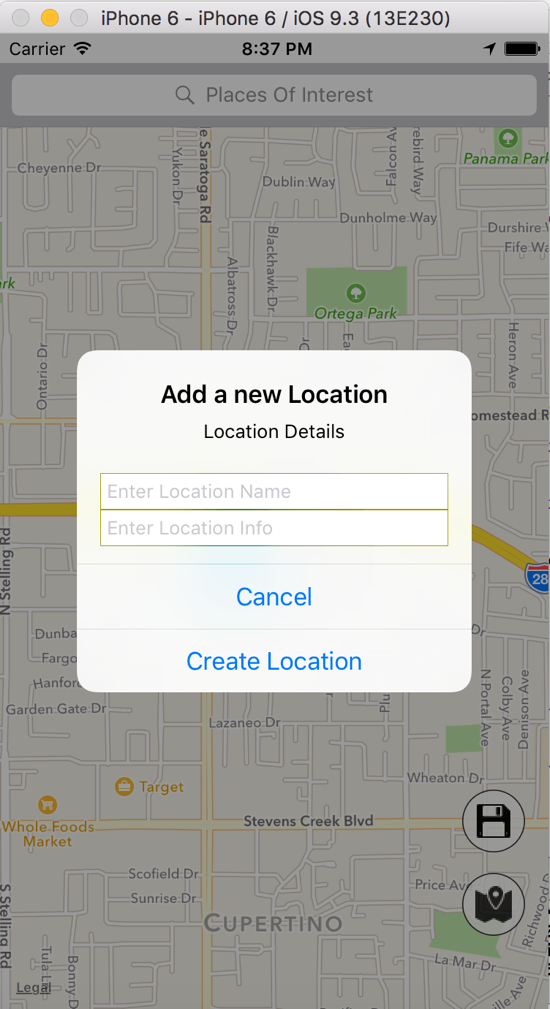


Fig 4 Fig 5

Now with the ability to store locations, the app must be able to direct the users there, here is where we ran into some issues with the pathing, Fig 6 shows the first attempt at pathing that was not working as required. As can be seen there are multiple times when the path seems to lead away from the destination before turning back and reaching the correct location. To solve this a “best path” algorithm was devised by Nicholas Judd. This algorithm simply determines what is the best and fastest way to reach each destination marked as part of the path.

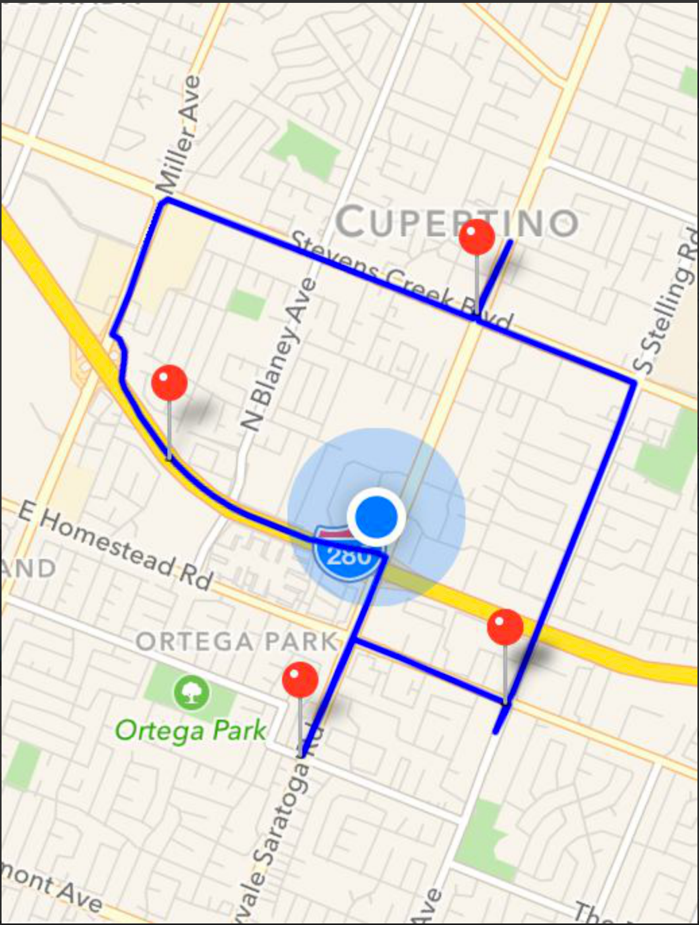


Fig 6

Below is the optimal path code

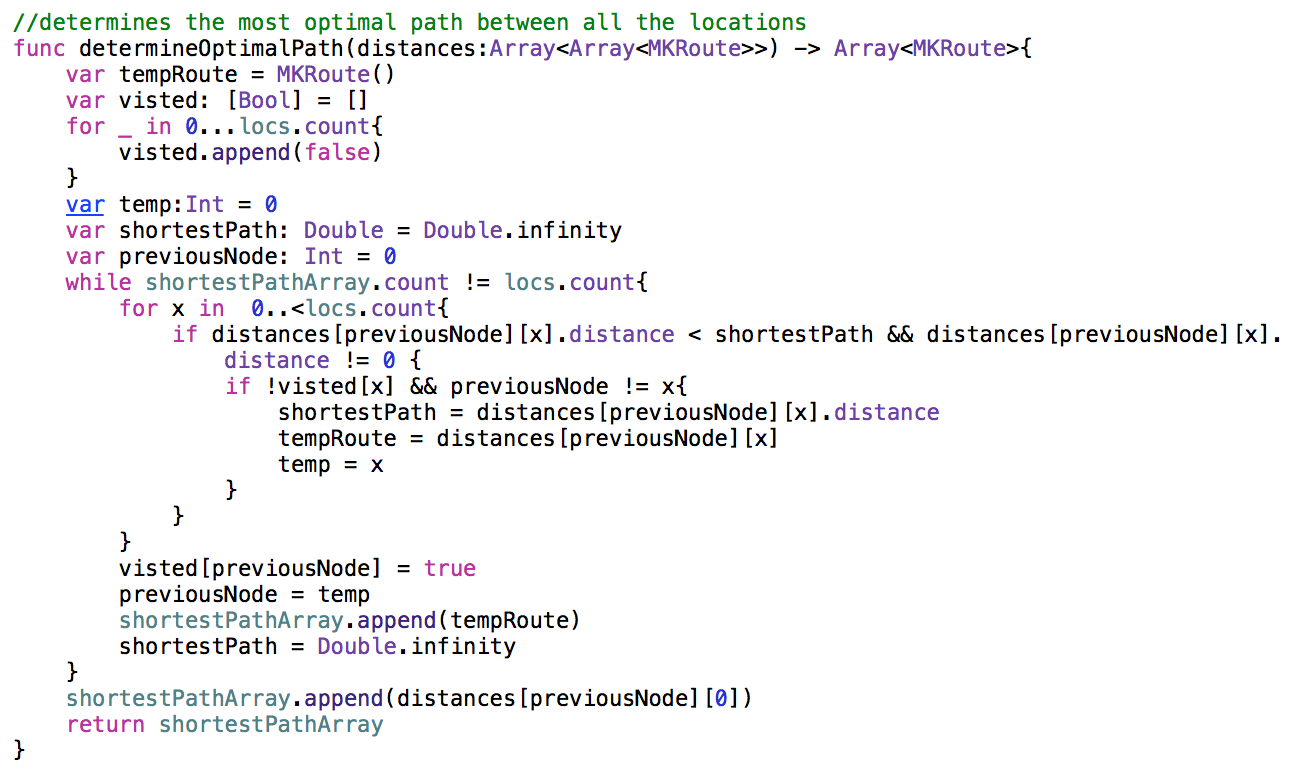


Fig 7

After passing the hurdle that was path creation, it was required to store these paths for later retrieval. The paths which are created by selecting more than one location from storage. Once the save button is pressed a prompt much like the one for storing locations appears allowing the user to add a name and description of their path.

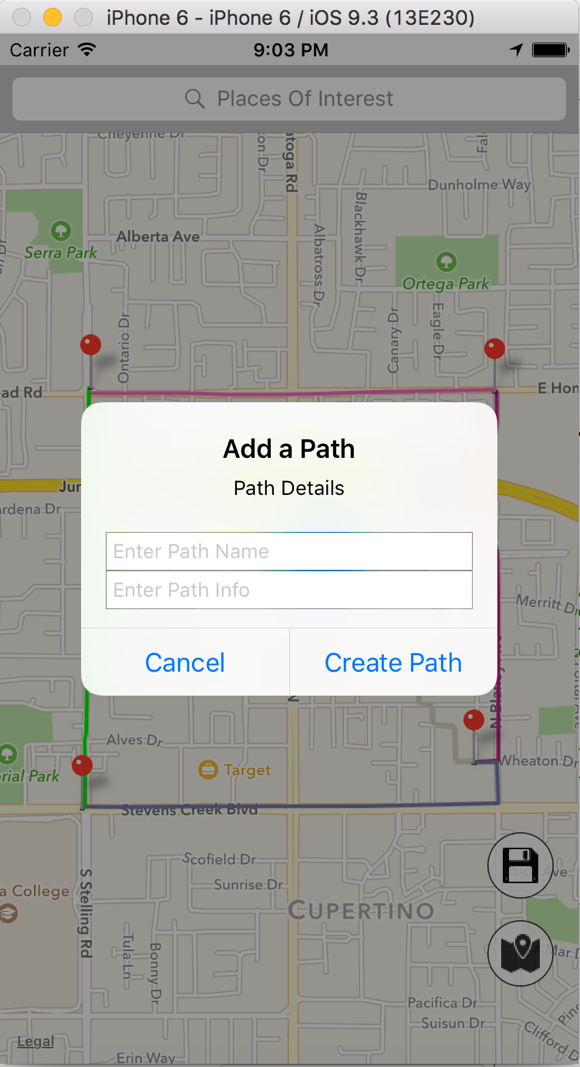
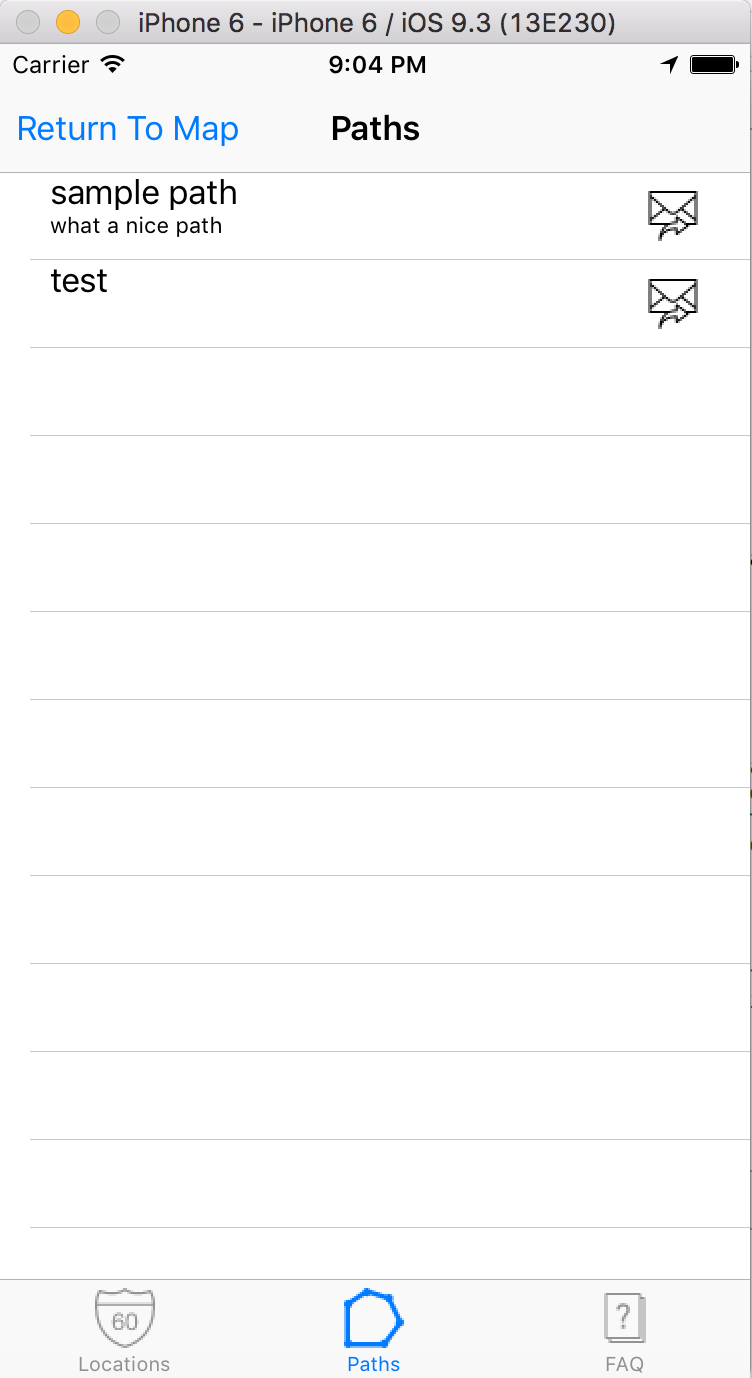


Fig 8 Fig 9

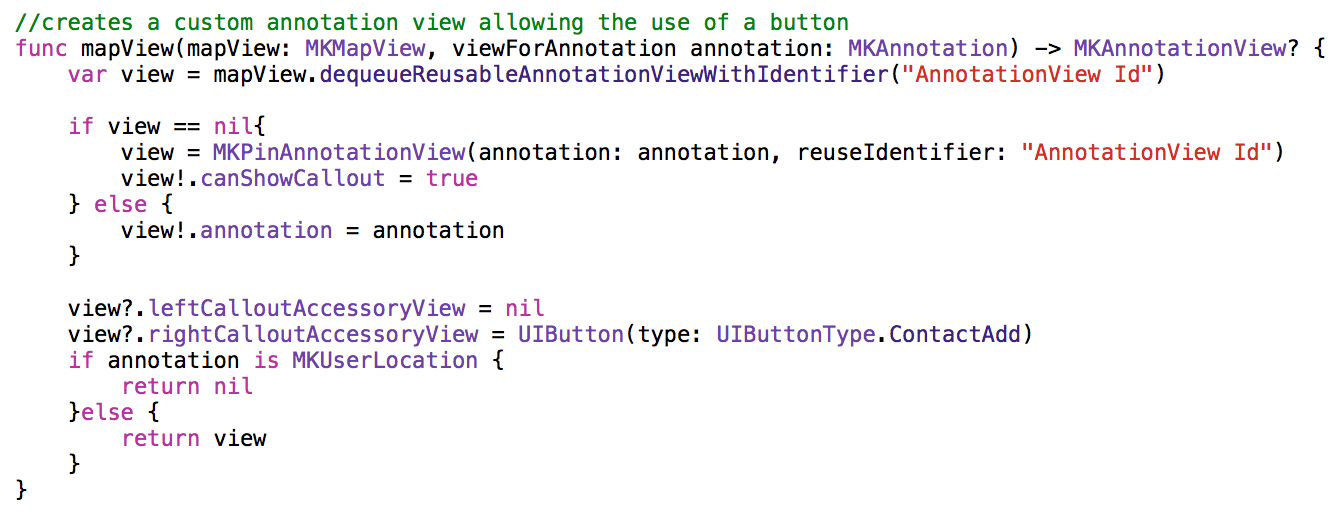
The following step was to implement a search feature, the ability to search for POIs is pretty standard with any map app, however within Follow My Footsteps it was required that you not only could search but also add this annotation to the stored location or path. This was implemented with the function displayed below (Fig 10).

Fig 10

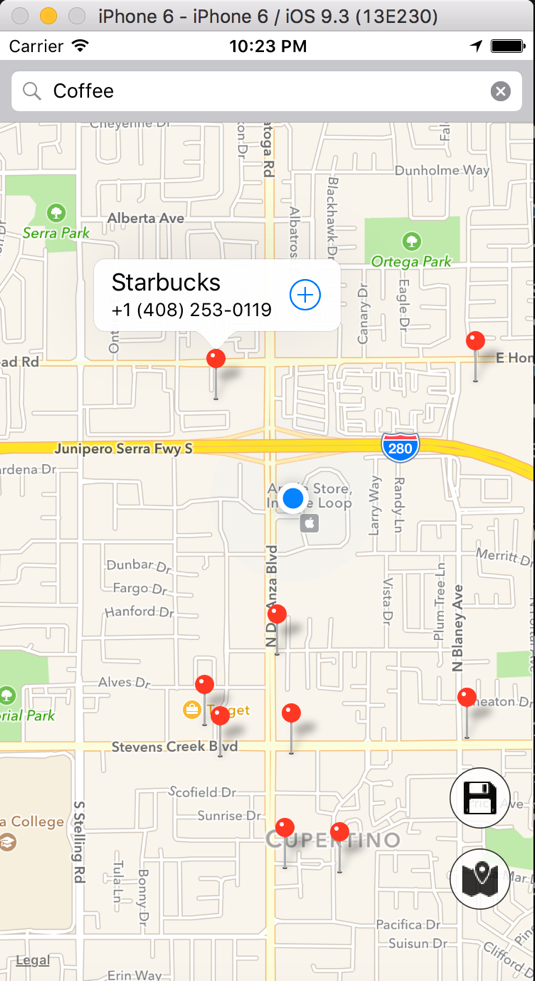


Fig 11

Finally, the ability to send paths was implemented, at present it is only possible to send paths over a multi-peer connection. Users will go to their stored paths screen, and select the send button, this will open a following screen with all phones on the same Wi-Fi or near enough for Bluetooth that has the app installed. The receiving phone will prompt the user to accept or decline the transfer, once the transfer is complete the receiving user will be able to set the sent path to the map.

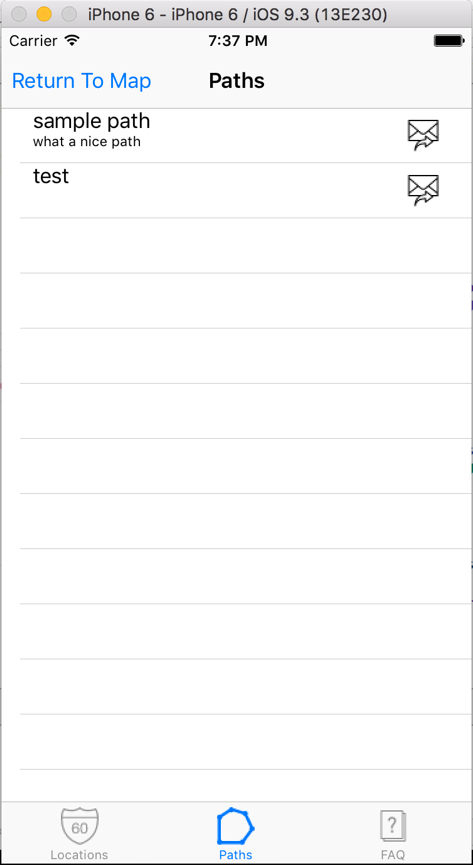
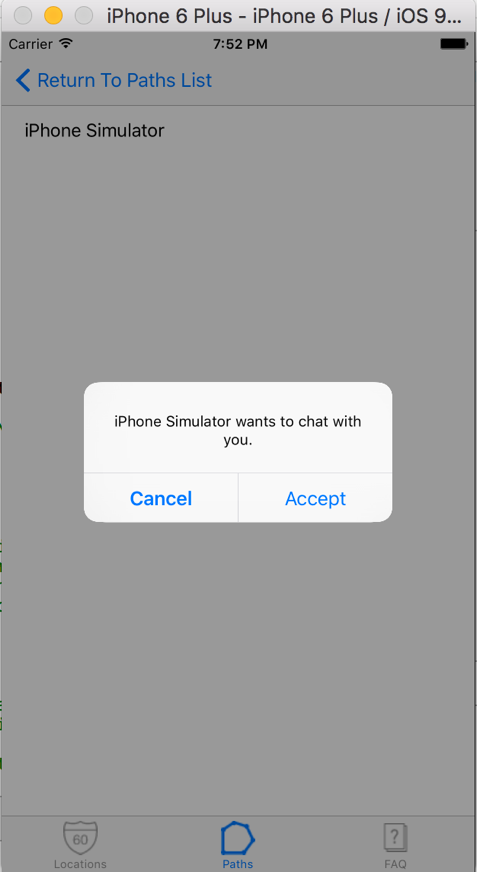


Fig 12 Fig 13

### Notes on HIG

### Designing for iOS

* The Map View is easy to understand, as it is similar to other map apps
* Text and Icons are all precise and easy to understand
* Each View takes the whole screen excluding any navigation bars and tabs

### iOS App Anatomy

* The four following categories from <https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Anatomy.html#//apple_ref/doc/uid/TP40006556-CH24-SW1> have been fulfilled
* **Bars.** Bars contain contextual information that tells users where they are and controls that help users navigate or initiate actions.
* **Content views.** Content views contain app-specific content and can enable behaviors such as scrolling, insertion, deletion, and rearrangement of items.
* **Controls.** Controls perform actions or display information.
* **Temporary views.** Temporary views appear briefly to give users important information or additional choices and functionality.

### Adaptivity and Layout

* Auto-Layout enables the app to be used on all devices and landscapes without much change to the view
* Tabs and buttons are easy to interact with and stay within the recommended hit target of 44 x 44

### Starting and Stopping

* After loading screen app opens quickly to the main view of the map
* On first start up users are prompted to allow for location services only while the app is running
* When receiving paths, the users are prompted to accept first

### Navigation

* A navigation bar gives the users a way to traverse the hierarchy of the app
* A tab bar in the stored data screens allows the user to seamlessly switch from locations to paths
* Each screen has only one path

### Modal Contexts

* The modal experiences in the app are only when:
  + Its critical to get the users attention, i.e. to start the location services
  + To search for POIs
  + To enter details on locations and paths

### Interactivity and Feedback

* The app has the ability to receive a long press to drop an annotation
* Buttons are only in areas where they are needed
* All standard gestures are supported

### Animation

* Only animation is the loading screen

### Branding

* There is none

### Colour and Typography

* There is currently no additional colouring added, this will be changed in a further update

### Icons and Graphics

* The icon for the app is simple and relates to the app nicely
* Icons within the app are common and most users will understand what they do right away

### Terminology and Wording

* The terminology used is easily understood
* Directions and prompts are informal and simple

### Integrating with iOS

* Due to the nature of the apps functions it is difficult for it not to be compared to the Apple Maps application

Time Log