National College of Ireland

BSc in Computing

2015/2016

Location Football

Technical Report

Shane Noonan

X12435988

shanenooonan@hotmail.com

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name:
Student ID:
Supervisor:

SECTION 2 Confirmation of Authorship
The acceptance of your work is subject to your signature on the following declaration:
I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:____
Date:____________

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College’s Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Table of Contents

Executive Summary

This applications goal is to address the problem of local football club information and tickets purchasing being scattered, unreliable and sometimes unreachable. This application aims to address this by providing a one stop place to find out all the information you need to choose which local football club to go and see or to just find relevant information on the football clubs around your location.

I used Swift programming language to code this project and Xcode 7 as the build environment. I implemented a cloud based database on Parse.com. I used the web application Import.io to scrape league table data from websites and all graphics where designed using the open sourced image software platform Gimp. An iPhone simulator was used initially for testing before iPhone 5 was used for user acceptance testing.

1. Background

The idea first stemmed from my second year software project where I worked on a web application that showed all the top European clubs on a google map interface and provided their stadium location with a news feed and relevant external links. However, I then realised that the app was somewhat irrelevant as there are plenty of football apps and websites covering the top clubs in the world. Then when on holiday in Asia and wanting to go see or even find out where and when their local football teams play, I found it messy and scattered to find the necessary information. This is when I thought about altering my second year project to be mobile and location based and to be more focused on match and ticket information and booking. As I am not based in Asia for the duration of this project I decided to base the application here in Ireland, were it can be used to find out information on the professional football clubs that we have here but that the application is set up in a way that it can easily adapt to add clubs from any country throughout the world.

Initially I decided to build my mobile application using HTML5 CSS3 and AngularJS through the Ionic Framework. The idea was that building through this platform would allow my app to run on both Android and iOS devices. However, I faced some difficulty when testing the application and also serious problems when trying to run initial application features on iOS devices. This made me worry and feel a bit uncomfortable with using a relatively new platform to build my final year project. I then decided to change my approach and had to make the choice on whether to build my project for Android using Java, which is what I had been learning throughout my degree or to take an all new approach and build it for iOS with Swift.

I decided to build my application for iPhone instead of Android because of a variety of reasons:

1. I have a strong interest in all things Apple and was keen to learn about iOS development.
2. Previously used Android Studio on other projects and hated it. Xcode seemed more developed and its Simulator was by far superior.
3. I had been quite poor at Java programming so learning a relatively new language such as Swift wasn't such a big deal as I would have been at a beginner's level no matter what language I chose.
4. Compared to Android users, iOS users are typically more loyal, engage more and spend more time per app.
5. Overall, iOS apps tend to earn more revenue than Android apps.
6. iOS users are more likely to update their OS, allowing developers to stop supporting older devices sooner.
7. Although Android currently dominates market share at over 80% (while iOS is at about 15%), iOS dominates the profit share, generating 85% more revenue for app makers than Android.

2. Aims

The primary aim of this application is to provide and easy to use and attractive way of finding information on football clubs around your current location.

To achieve this aim the application must have comply with the following objectives:

- Show local clubs on a map interface around the user's current location.
- Allow the user to select clubs to find out more information.
- Provide a link to the selected clubs official website.

- Provide league table details on clubs.
- Provide a link to book tickets to see selected club.
- Provide directions to the selected club location.

3. Technologies

I will be using the latest version of Apples Xcode IDE to develop this IOS application.

Here I will be using a variety of Apples Cocoa and CocoaTouch frameworks including Mapkit, Corelocation, and also third party frameworks such as Alamofire and Parse.

I will be using the Swift programming language to code within the Xcode environment.

I will be using Parse.com as my backend to hold football club and user information as well as importing their iOS framework into my application project.

I will be using Import.io in order to scrape relevant data from websites.

I will be using the image editing software Gimp to make my apps graphics.

4. Structure

The first section, gives the reader a general overview to the project in relation to what it is. It outlines to the reader what the background to the project is, the main aims of the project and an overview of the technologies that were used.

The second section details the functional requirements, describing what they are and how they are implemented. It then goes on to explain what data requirements, environmental requirements and usability requirements are needed. This section will go on to detail the design and architecture of the system, how the system was implemented and how the user interface is designed.

The third section contains the conclusions to the project. While the forth section will describe any further developments planned in relation to the project.

The fifth section shows the bibliography of all the resources used to complete the project and finally the sixth and final chapter contains all the appendices to this document, such as the project proposal, project plan, and the monthly reflective journals.

2. System

In this section, I will outline the structure of this technical report. I will present a detailed description of all the requirements for this iOS Application. I will then look at the Design and Architecture of the System and how the different components tie together to form the overall working environment. I will also look in detail, at the technologies used and the implementation of the project. I will discuss what testing methods I used throughout the development process and finally, I will describe the Graphical User Interface.

2.1. Requirements

2.1.1. Functional requirements

In this section I will talk about the functional requirements. A functional requirement is a statement that identifies what the system must do.

Requirement 1 <Registration & Login>

Description & Priority

This requirement is the Registration and login menu of the application and features a login and register page. Its priority is to allow the user to register and then login to access the application main interface.

Use Case

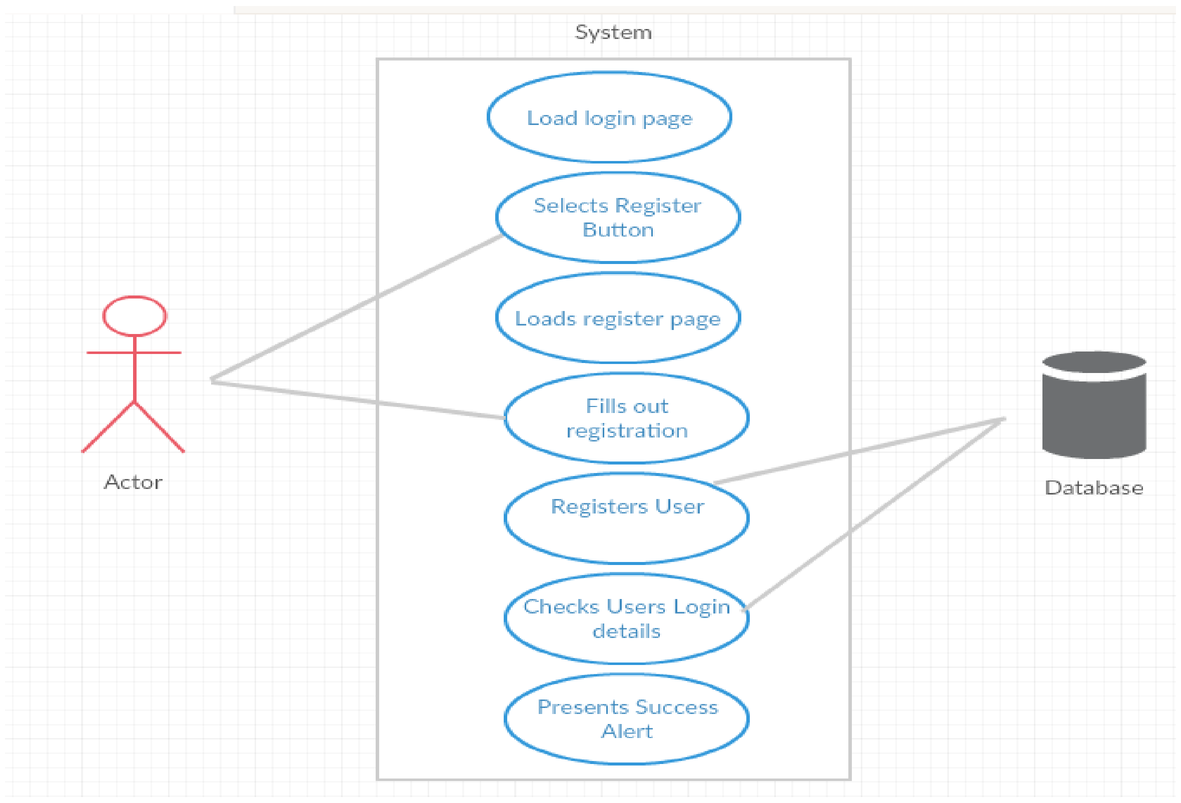
Scope

The scope of this use case is to let the user register to use the app and then login with their created credentials.

Description

This use case describes the actions of registering as a user and then logging in with those credentials.

Use Case Diagram



Flow Description

Precondition

The system is installed on user's mobile device.

Activation

This use case starts when an <Actor>starts the app.

Main flow

1. The system loads the login page.
2. The <Actor> selects the register button
3. The system loads the register page
4. The <Actor> fills out required details and selects register button.
5. The system checks the required details, registers the user and presents a success alert.
6. The <Actor> selects to go back to the login page.
7. The system loads the login page
8. The <Actor> puts in created credentials and selects login button.
9. The system checks user credentials, accepts the User and presents a success alert.

Alternate flow

2A : <Already registered>

1. The system loads the login page.
2. The use case continues at position 8 of the main flow

Termination

The system presents the main interface after accepting user's login.

Post condition

The system goes into a wait state.

Requirement 2 <Main Interface and Club selection>

Description & Priority

After the user logs in the main interface should take the users current location and display their location as well as the football clubs as pins on a map interface.

Use Case

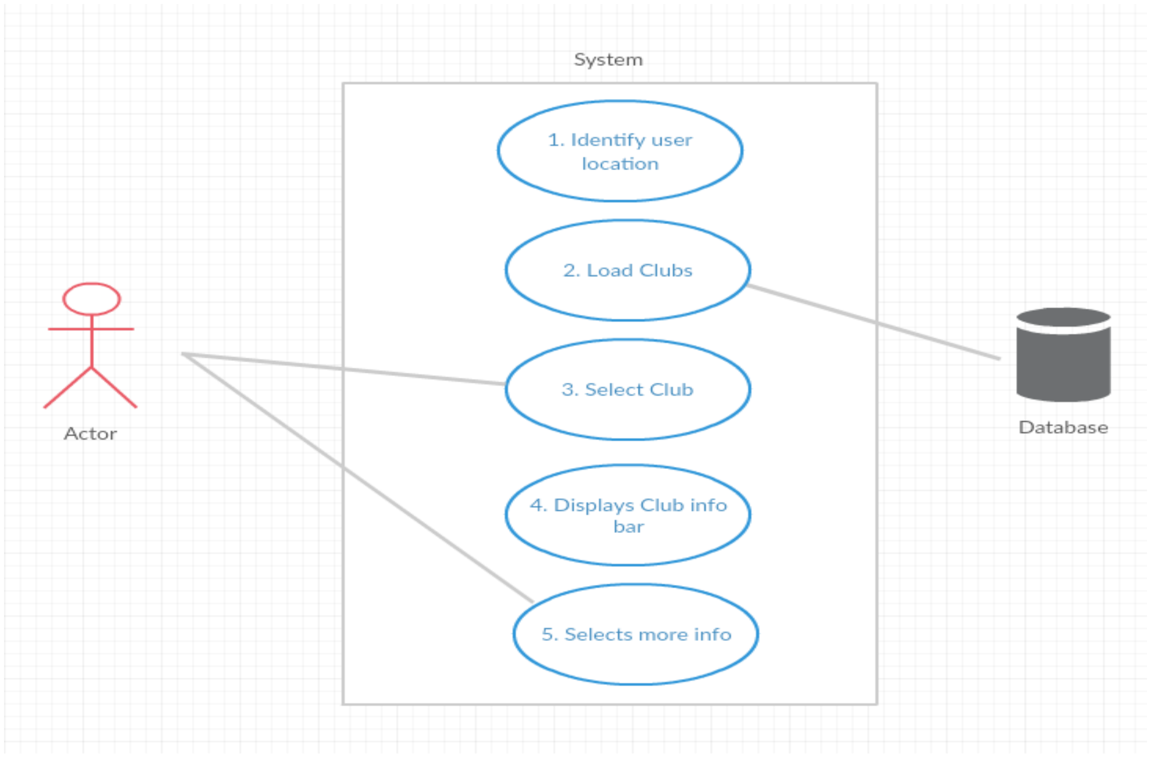
Scope

The scope of this use case is to locate the user and display their location as well as clubs loaded from the database on a map interface and allow the user to make selection on each of the clubs displayed on the map.

Description

This use case describes how the system gathers the user's location and then re-trieves the clubs coordinate data from the database and then presents it on a google map interface. The user can then interact with this interface choosing to select clubs from their coordinate annotations and allowing them to choose to find out more in-formation about the selected club.

Use Case Diagram



Flow Description

Precondition

The system accepts the users login details.

Activation

This use case starts when an <Actor> logs into the system.

Main flow

1. The system identifies the user's location, puts it onto the map interface.
2. The system loads clubs locations by onto the map interface.
3. The <Actor> selects a club on the map interface
4. The system displays an info-bar for the selected club.
5. The <Actor> selects the "more info" button from the selected club info-bar.

Alternate flow

A1 : <A1>

1. The system identifies the user's location, then loads their location and clubs close by onto the map interface.
2. The <Actor> selects a club on the map interface.
3. The system displays a info-bar for the selected club

4. The <Actor> unselects the club.
5. The system closes the selected club’s info-bar.
6. The use case continues at position 3 of the main flow

Termination

The system presents the next interface.

Post condition

The system goes into a wait state

Requirement 3 <League data selection>

Description & Priority

This requirement is when the user selects the League button from the club information menu. A live league data table then appears were the user can view the selected clubs league position and league table details.

Use Case

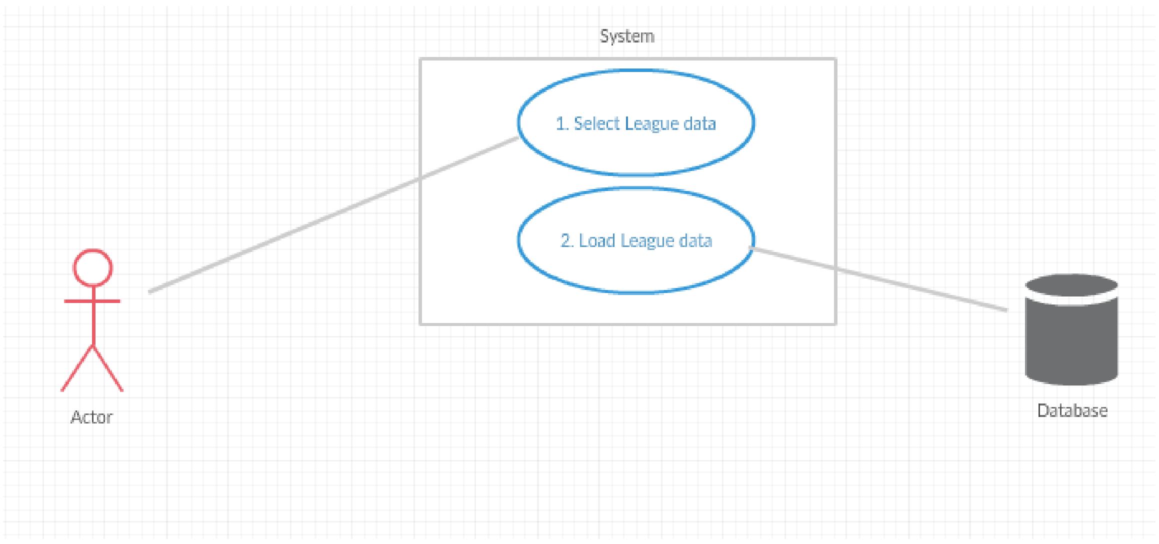
Scope

The scope of this use case is to allow the user to execute the league option displayed on the club information menu.

Description

This use case describes the user choosing to select the league button from the selected club menu and how the system accepts this and then retrieves live league data relating to the club chosen and presents it to the user.

Use Case Diagram



Flow Description

Precondition

The <Actor> selects the more info button.

Activation

This use case starts when the system loads the club menu page displaying the options available.

Main flow

1. The <Actor> makes the League selection from the options menu.
2. The system loads the League option to display the live club league table data.

Termination

The system presents the option the user requested.

Post condition

The system goes into a wait state.

Requirement 4 <Club website selection>

Description & Priority

This requirement is when the user selects the Club website button from the club information menu. An external link to the official website of the club selected then appears in an internal webview.

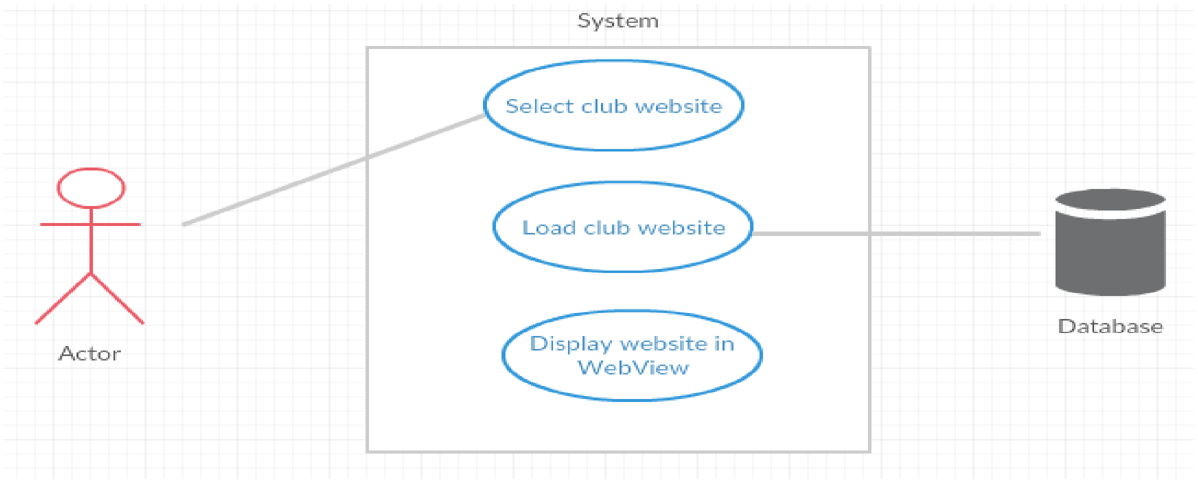
Scope

The scope of this use case is to allow the user to execute the club website option displayed on the club information menu.

Description

This use case describes the user selecting the club website option, the system retrieving the relevant club website information from the database and then presenting it to the user. The system opens the link using an internal webview.

Use Case Diagram



Flow Description

Precondition

The <Actor> selects the more info button.

Activation

This use case starts when the system loads the club information page displaying the options available.

Main flow

1. The <Actor> makes the club website selection from the options menu.
2. The system loads club website details and presents the website in an internal webview.

Termination

The system presents the option the user requested.

Post condition

The system goes into a wait state.

Requirement 5 <Ticket selection>

Description & Priority

This requirement is when the user selects the Tickets button from the club information menu. An external link to a ticket provider for the club selected then appears in an internal webview.

Use Case

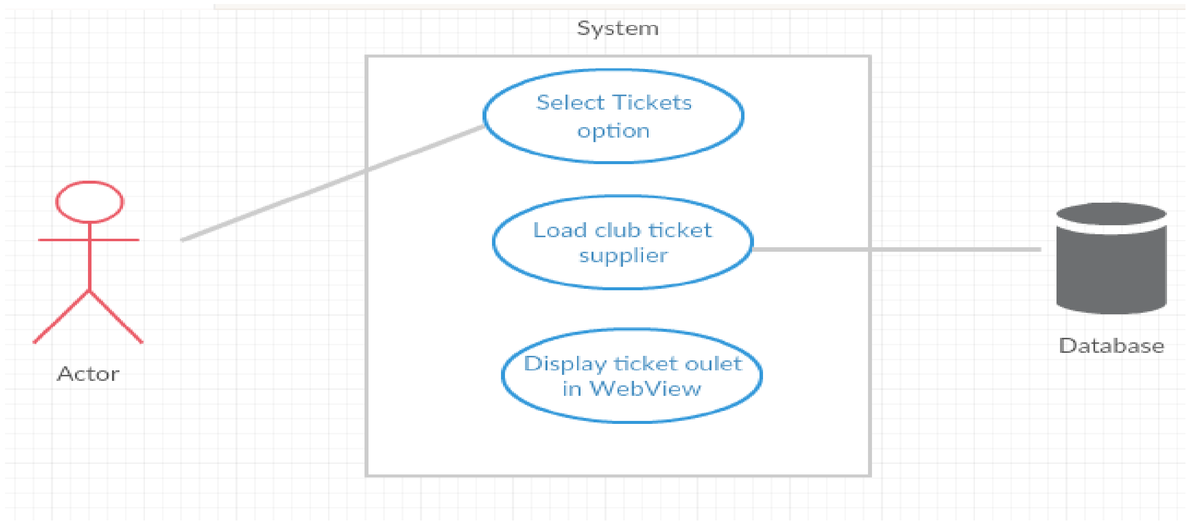
Scope

The scope of this use case is to allow the user to execute the options displayed on the club information menu.

Description

This use case describes the user selecting the tickets option, the system retrieving the relevant club ticket information from the database and then presenting it to the user in an internal webview.

Use Case Diagram



Flow Description

Precondition

The <Actor> selects the more info button.

Activation

This use case starts when the system loads the club menu page displaying the options available.

Main flow

1. The <Actor> makes the Ticket selection from the options menu.
2. The system loads the required ticket link and presents it to the user in an internal webview.

Termination

The system presents the option the user requested.

Post condition

The system goes into a wait state.

Requirement 6 <Directions selection>

Description & Priority

This requirement is when the user selects the Directions button from the club information menu. The system then takes the user and the club locations into account before drawing a directions route onto a map interface for the user.

Use Case

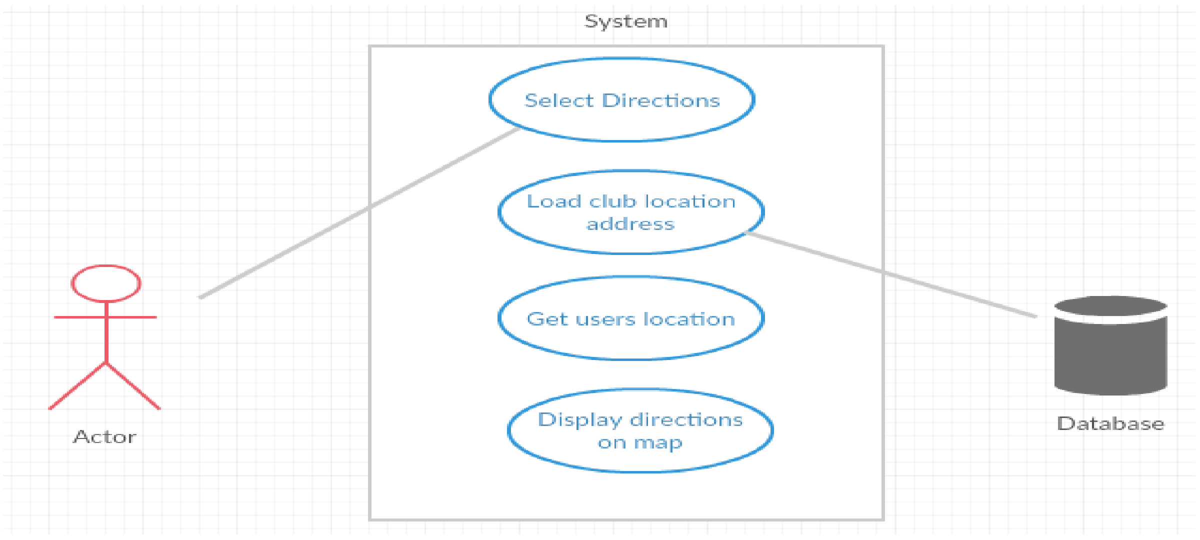
Scope

The scope of this use case is to allow the user to execute the directions option displayed on the club menu list.

Description

This use case describes the user selecting the directions option, the system retrieving the relevant club address from the database and before displaying a direction route from the user's location to the selected club's stadium.

Use Case Diagram



Flow Description

Precondition

The <Actor> selects the more info button.

Activation

This use case starts when the system loads the club menu page displaying the options available.

Main flow

1. The <Actor> makes the Directions selection from the options menu.
2. The system loads required club address from database and retrieves the user's location.
3. The system displays a directional route from the user's location to the selected clubs stadium on a map interface.

Termination

The system presents the option the user requested.

Post condition

The system goes into a wait state.

2.1.2. Data requirements

In this section, I will describe the data requirements, which are essential for the application to run efficiently.

- Parse.com Integration: The app will connect to a cloud based database powered by Parse.com. This database will store information about the users and also football club information. The application has the Parse SDK installed allowing the app access the data in order retrieve an edit information through the use of the application.

- Import.io Integration: The app will also have to connect with my Import.io API in order to get league table information. The application will have the Alamofire Framework installed allowing it to make a HTTP call to Import.io Restful API service and return the relevant information in JSON format.

- Other Data: The app will hold some data within the ViewControllers of the application, this allows for quick manipulation of data.

2.1.3. User requirements

In this section, I will outline the user requirements. These are essential requirements that the user must have in order to use the application.

- IPhone: The user must possess an iPhone.

- IOS 9: The user should preferably have the latest iOS software installed, but the application is not limited to iOS 9 and will be backwards compatible.

- AppStore Account: The user should have an Apple AppStore account in order to download the application.

- Internet Access: The user will need Internet access in order to download and use the application.

- Location Enabled: The user will have location features turned on in order for the application to display the user's current location. However the application can still function if user doesn't allow it but will be limited in what features work.

2.1.4. Environmental Requirements

In this section, I will outline the environmental requirements. These are the essential requirements that are needed to develop the application.

- MacBook: A MacBook is required to run Xcode.

- iPhone: An iPhone is needed to run application during development and testing.

- Xcode: Xcode is essential as it is the only way to build a native iOS application through the development environment provided by Apple.

- iPhone Simulator: This provides a way of testing the application quickly through the development. It is usually included in the Xcode package.

- Internet Access: This is required to access Apple documentation and other various resources as well as the apps Parse database and also league table data through Import.io

- Parse.com Account: This is required to add/update club or user information.

- Import.io Account: This is required in order to access/add league table data for the application to retrieve it in JSON format.

2.1.5. Usability requirements

This section will highlight usability requirements. These will provide the objectives during the interface and design process.

- Reliability: The system should be capable of allowing many users to access and use the app at the same time.

- Understandable: The interface will be easy to use and understand.

- Operable: Each action should be consistent. Error messages should explain problems if they occur.

- Attractiveness: The application layout and graphics should be visually attractive and appealing. The colour scheme should be appealing and the layout should be simple in order for customers of all levels be able to use it.

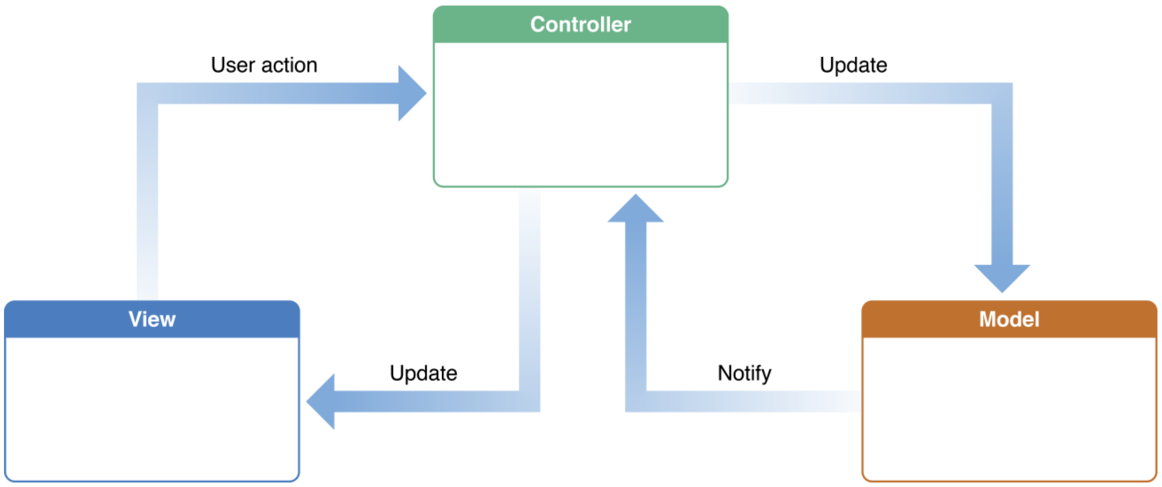
2.2. Design and Architecture

The main architectural aim of this iOS application is to make it is as lightweight and fast as possible. This is important as the user will have to have space on their device to hold the application. It will have to be fast and responsive to ensure user satisfaction. MVC is central to a good design for an iOS application.

Model: Represents the business logic of your application

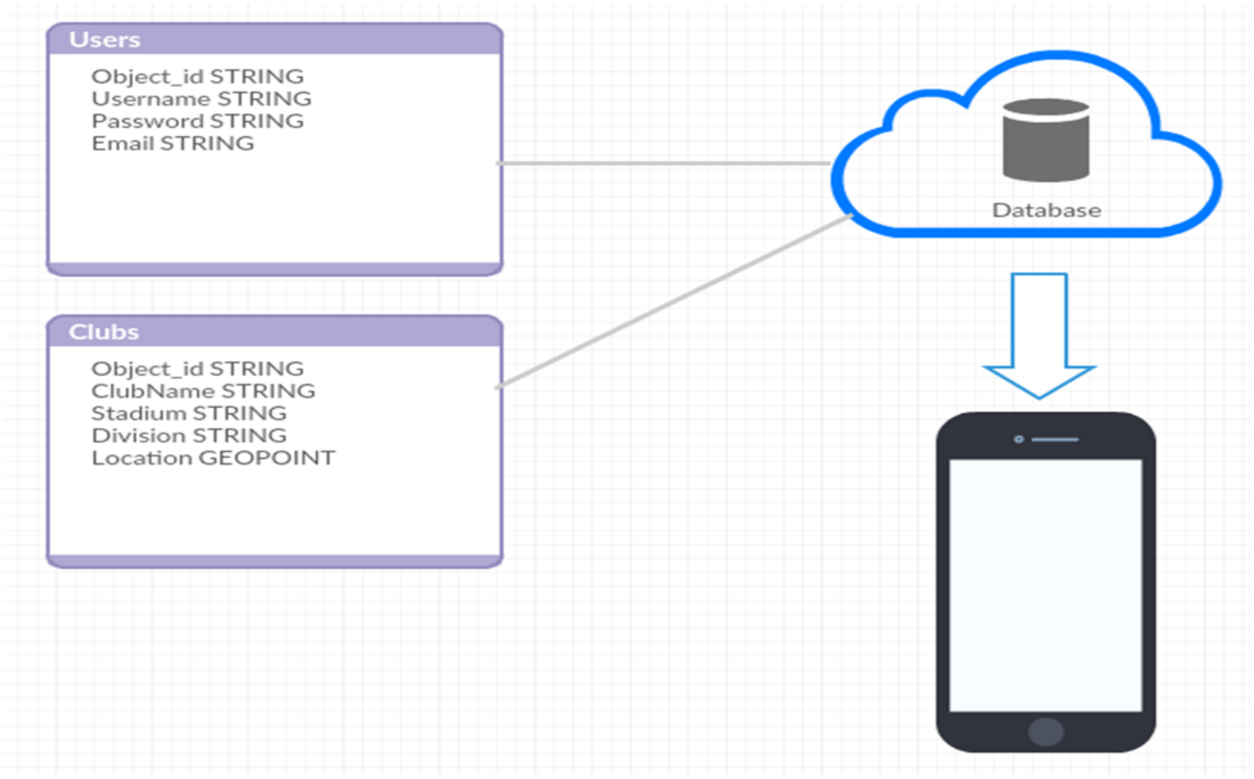
View: Represents what the user sees in the device

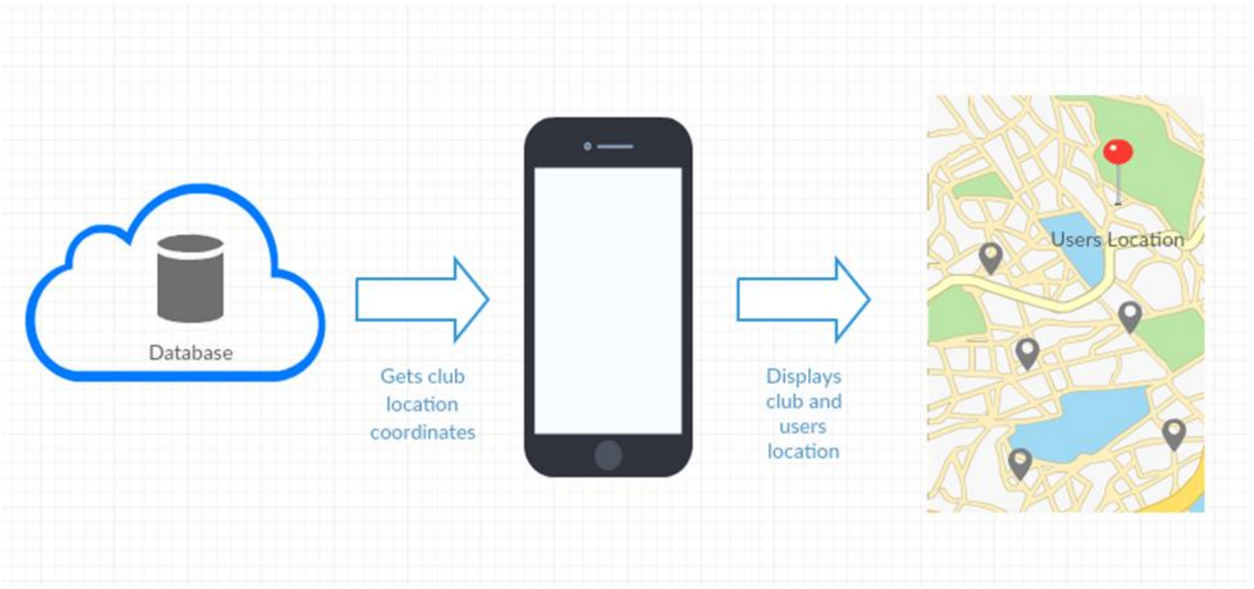
Controller: Acts as a mediator between the Model and View.

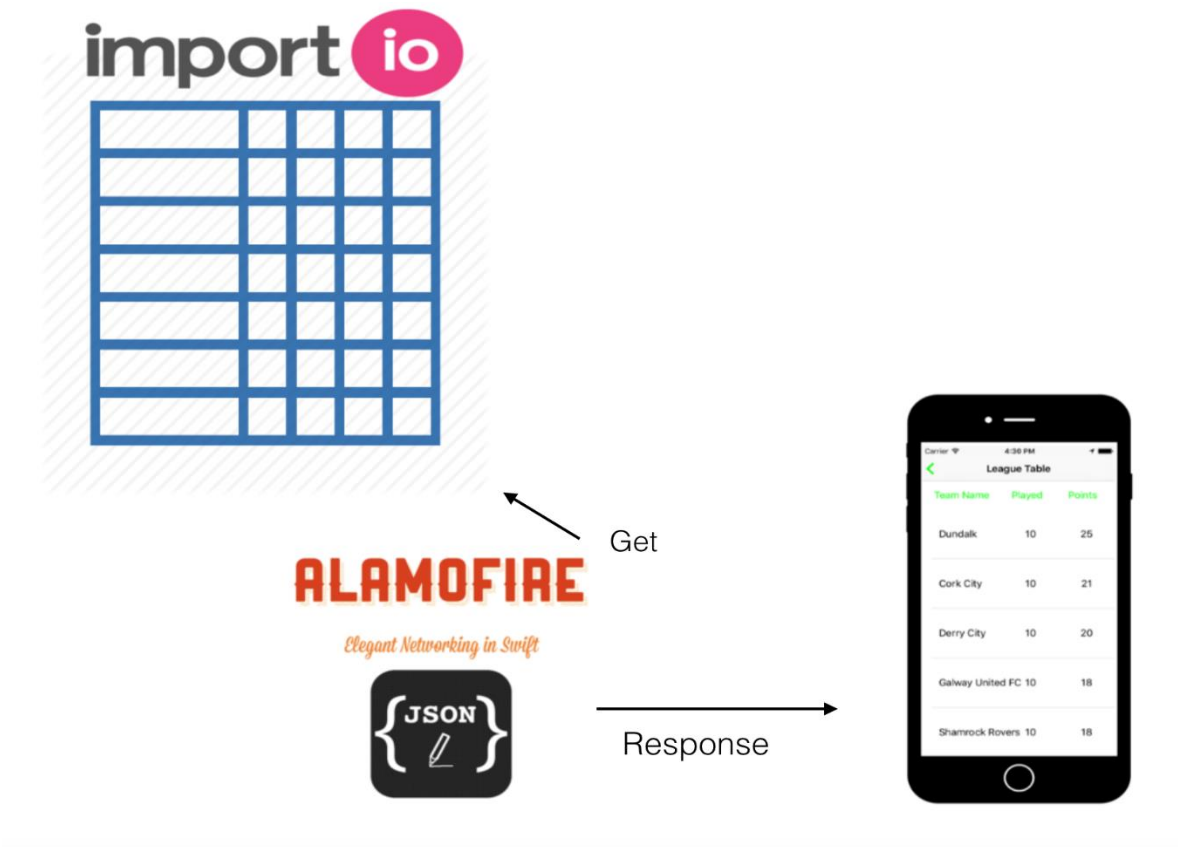


COMMUNICATION ARCHITECTURE

My application will need to communicate with my cloud based database in order to get and display club information to the user as well as authenticating the user at the register/login pages. It will also have to communicate with the Import.io RESTful API in order to retrieve live league table information.


Users
Object_id STRING
Username STRING
Password STRING
Email STRING
Clubs
Object_id STRING
ClubName STRING
Stadium STRING
Division STRING
Location GEOPOINT
Database


Database
Gets club location coordinates
Displays club and users location
Users Location



Team Name	Played	Points
Dundalk	10	25
Cork City	10	21
Derry City	10	20
Galway United FC	10	18
Shamrock Rovers	10	18

2.3. Implementation

The purpose of this section is to describe the technologies used in the implementation of the iPhone application. It was also cover the methods used in the implementation.

2.3.1. Technologies

<u>Xcode</u>

Xcode is Apple’s own IDE. It contains a suite of tools developed by Apple that forms the basis of their developing platform for iOS and MacOS. It also provides an iOS device simulator which can be used to run and test developing applications.

<u>Swift</u>

According to Wikipedia, Swift is a multi-paradigm, compiled programming language created for iOS, OS X, watchOS and tvOS development by Apple Inc. Swift is designed to work with Apple's Cocoa and Cocoa Touch frameworks and the large body of existing

Objective-C code written for Apple products. Swift is intended to be more resilient to erroneous code ("safer") than Objective-C and also more concise. It is a fairly new programming language being originally released in 2014.

<u>Parse.com</u>

Parse is a Backend as a service providing company which was acquired by Facebook in 2013. They provide a cloud based application development platform.

All data in my application will be stored in a cloud database on Parse.com. The Parse SDK is then installed within the application development folder which allows the app to connect to the online database.

<u>Cocoapods:</u>

CocoaPods is a dependency manager for Cocoa projects that provides a standard format for managing external libraries. CocoaPods focuses on source-based distribution of third party code and automatic integration into Xcode projects. CocoaPods runs from the command line and I used this to install the Alamofire framework on my application.

<u>Alamofire:</u>

Alamofire is an HTTP networking library written in Swift. It provides an interface on top of Apple's Foundation networking framework that simplifies a number of common networking tasks. It provides chainable response/request methods, JSON parameter and response serialization, authentication, as well as other features. For this project I will be using it to perform the basic networking task of requesting data from my Import.io RESTful API.

<u>Import.io</u>

Import.io is a web-based platform for extracting data from websites without writing any code. The tool allows you to create an API using their point and click interface. The data that users collect is stored on Import.io's cloud server. You can also generate an API from the data collected and easily integrate live web data into your own applications.

JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. As well as being easy for humans to read and write, it’s also easy for machines to parse and generate. It is completely language independent but uses conventions that are familiar to programmers of the C-family of languages. Together these properties make JSON an ideal data-interchange language.

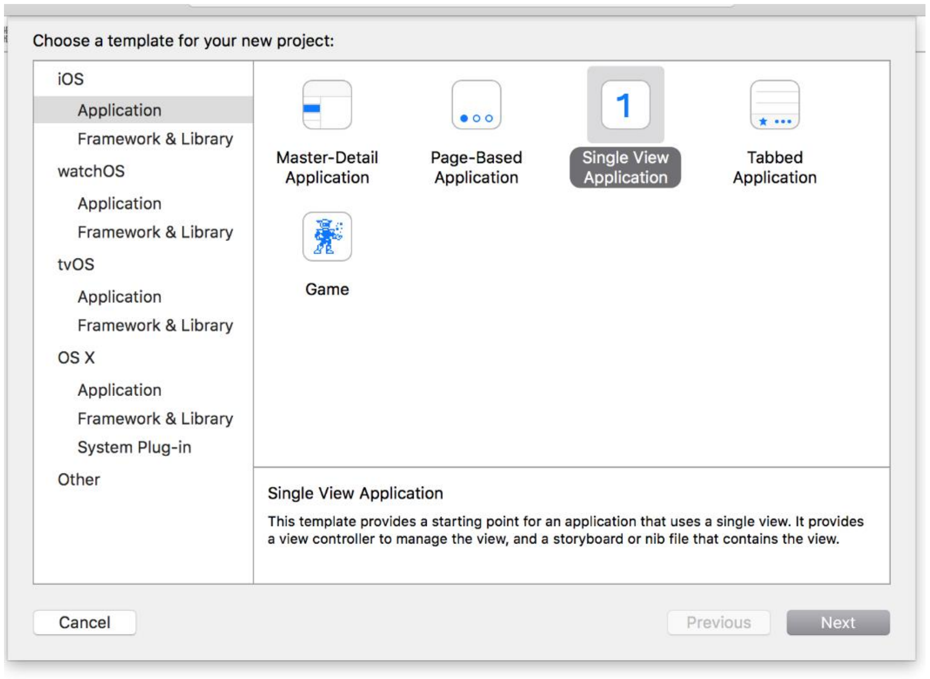
Gimp

Gimp is a free and open-source graphics and image editor used for image retouching, editing more specialized tasks. I used this software to develop all graphics and icons used in my application.

2.3.2. Procedures

Project Creation

In order to create this project I chose the single-view application template created within Xcode.

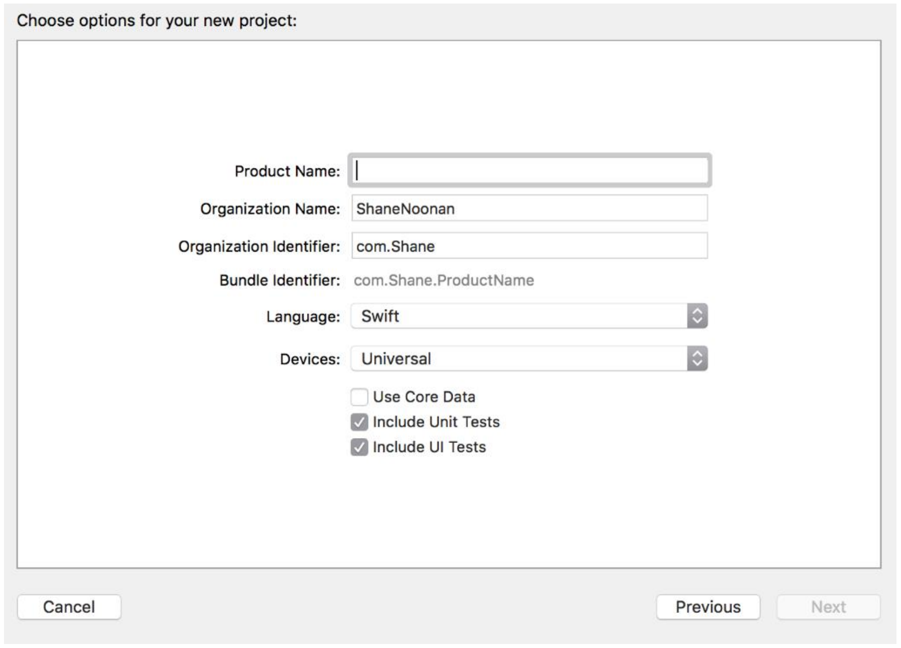


Single View Application

This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a storyboard or nib file that contains the view.

I found using the Single-View Application setup suited my applications interface and it allowed me to easily add and customize scenes and their controllers.

I then chose my projects name, the language used and which devices the application will run on. I then chose to include Unit Tests, this allowed for a quicker setup when unit testing my app later on.


Choose options for your new project:

Product Name:

Organization Name: ShaneNoonan

Organization Identifier: com.Shane

Bundle Identifier: com.Shane.ProductName

Language: Swift

Devices: Universal

☐ Use Core Data

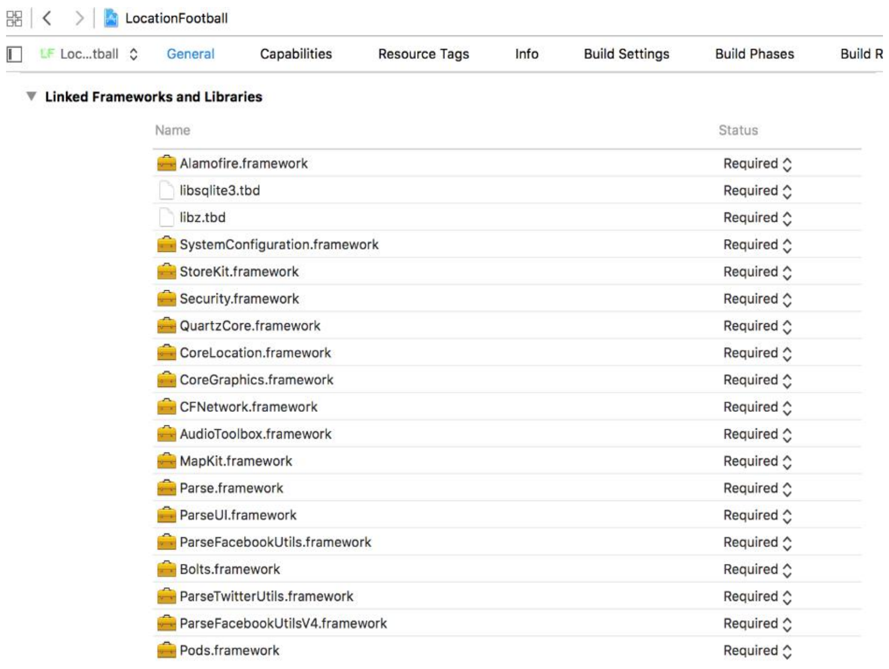
☑ Include Unit Tests

☑ Include UI Tests

(Note: For devices I chose iPhone not universal and also did not include UITests)

<u>Adding Parse SDK Framework</u>

In order for my application to connect to my online database I had to install and add the Parse SDK Framework and libraries to my project in Xcode. I did this by downloading all the Parse frameworks from their official GitHub account and then adding them and the other required frameworks to my project as seen in the screenshot below.



Linked Frameworks and Libraries

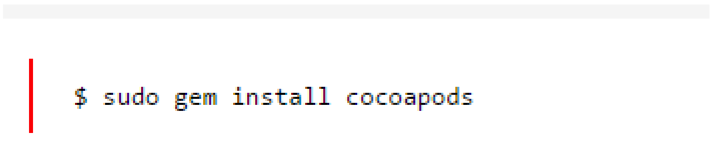
Name	Status
Alamofire.framework	Required
libsqlite3.tbd	Required
libz.tbd	Required
SystemConfiguration.framework	Required
StoreKit.framework	Required
Security.framework	Required
QuartzCore.framework	Required
CoreLocation.framework	Required
CoreGraphics.framework	Required
CFNetwork.framework	Required
AudioToolbox.framework	Required
MapKit.framework	Required
Parse.framework	Required
ParseUI.framework	Required
ParseFacebookUtils.framework	Required
Bolts.framework	Required
ParseTwitterUtils.framework	Required
ParseFacebookUtilsV4.framework	Required
Pods.framework	Required

Also my Parse API client Key and application ID had to be added to the app delegate file of my project in order for my application to communicate with my Parse database.

```
import UIKit
import Parse
import Bolts

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject:
        AnyObject]?) -> Bool {
        // Override point for customization after application launch.

        Parse.setApplicationId("80cgsCxer6PR6HleMktDr5UstZ2flE7YOmBghxBE",
            clientKey: "gyX2XfPrYVjcAg5GFZM6nCAfTdlpZa1g713fLITB")

        return true
    }
```

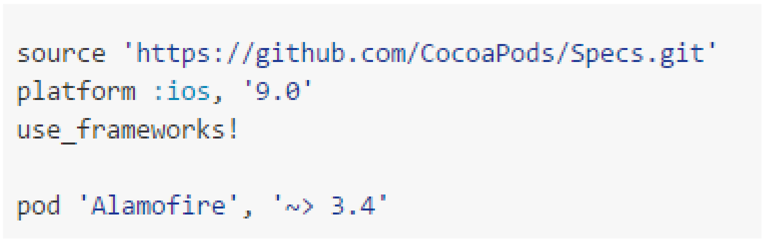
Adding Alamofire Framework via Cocoapods

In order for my application to make http calls to my Import.io API I decided to use the popular swift networking framework Alamofire. To install this framework I decided to use the very popular dependency manager Cocoapods. It is installed via Terminal (Command Line) as seen below

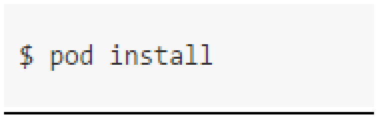


```
$ sudo gem install cocoapods
```

Once Cocoapods was installed the latest version of Alamofire was integrated into my Xcode project using CocoaPods in Terminal



```
source 'https://github.com/CocoaPods/Specs.git'
platform :ios, '9.0'
use_frameworks!

pod 'Alamofire', '~> 3.4'
```

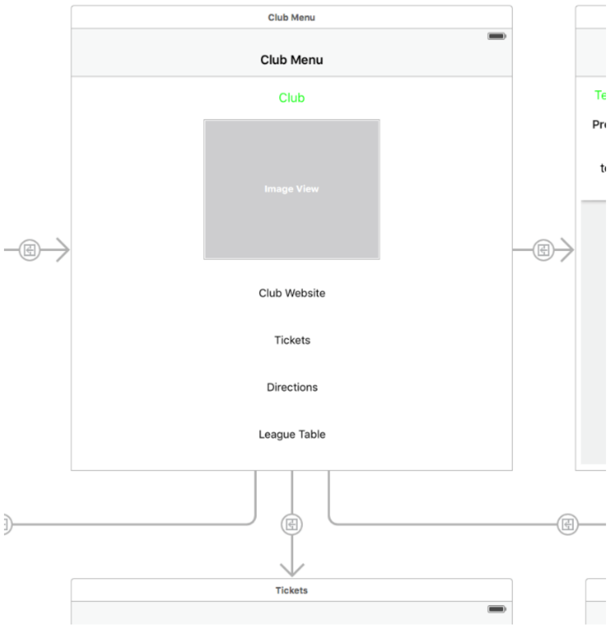


```
$ pod install
```

Creating Storyboard

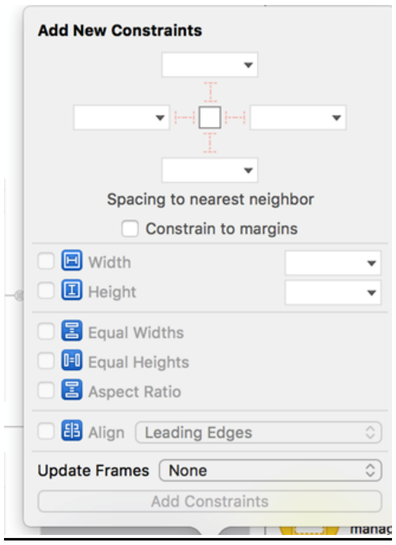
In Xcode the UI of the application you are building is designed through storyboard. This is a graphical representation of what your application will look like on an iOS mobile device. Storyboard is made up of scenes which are graphical representations of each interface of your application. Here you can use Xcodes drag and drop option to add visuals representations of various object features and views to each scene such as textfields, imageviews, tableviews, buttons and many more. I chose to add a navigation controller into my application, this allowed each of the scenes I created to be connected to each other via a navigation bar which contains a title and also a back button to make it easier for the user to navigate throughout the application.

In the screenshot below you can see one of the scenes I have created for the club menu interface of my application. The arrows going and coming from the club menu view are connections. This type of connection is known as a segue (pronounced: seg-way) and represents a transition from one scene to another. Segues are triggered by taps on buttons, table view cells, gestures, etc. You can also pass data between scenes



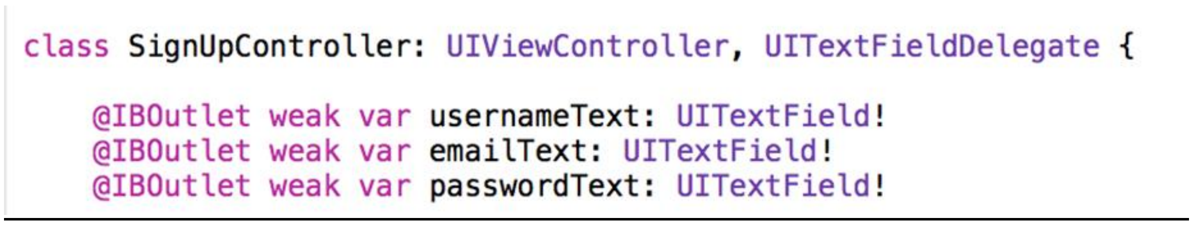
<u>Auto Layout and Adding Constraints:</u>

According to Apple’s developer website, Auto Layout dynamically calculates the size and position of all the views in your view hierarchy based on constraints placed on those views. This basically means that when you position a button under a under an imageview so that it is horizontally centered. If the images size changes the button size will change automatically to match. This was hugely beneficial when building this application as it made it much easier to adjust the layout of the application to support different screen sizes and resolutions. I added constraints manually in some cases but also used Xcode's powerful suggested constraints tool as well to add the required constraints to my apps objects and views.



Creating outlets:

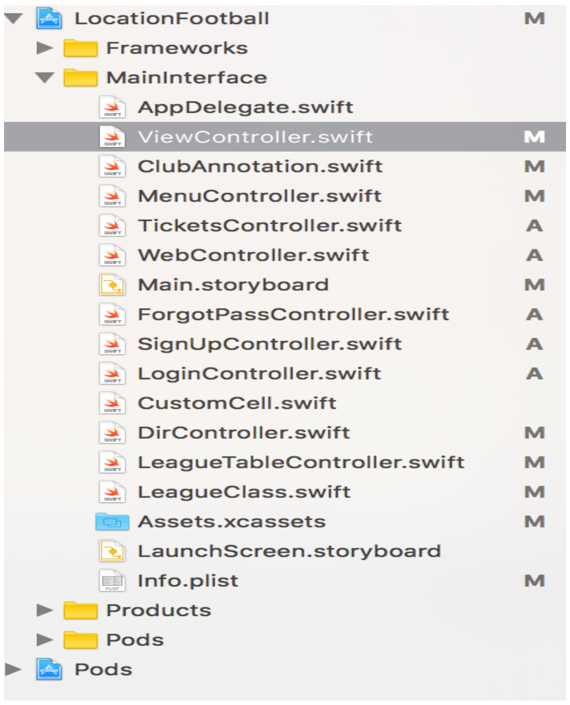
In order for the objects created via the storyboard to communicate with the code in the corresponding Controller, Outlets must be created to connect them. This is done through a drag and drop approach were you select them object and drag it onto the corresponding controller, then giving it a name and a type.



```
class SignUpController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var usernameText: UITextField!
    @IBOutlet weak var emailText: UITextField!
    @IBOutlet weak var passwordText: UITextField!
```

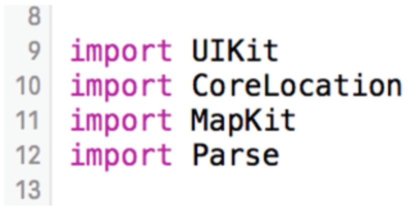
Creating ViewControllers

In Xcode ViewControllers are swift code files used to control each scene (storyboard) of your application. Within my project I created separate ViewControllers for each of the scenes of my application.


```
LocationFootball                    M
  Frameworks
  MainInterface
    AppDelegate.swift
    ViewController.swift            M
    ClubAnnotation.swift            M
    MenuController.swift            M
    TicketsController.swift         A
    WebController.swift             A
    Main.storyboard                 M
    ForgotPassController.swift      A
    SignUpController.swift          A
    LoginController.swift           A
    CustomCell.swift
    DirController.swift             M
    LeagueTableController.swift     M
    LeagueClass.swift               M
    Assets.xcassets                 M
    LaunchScreen.storyboard
    Info.plist                      M
  Products
  Pods
Pods
```

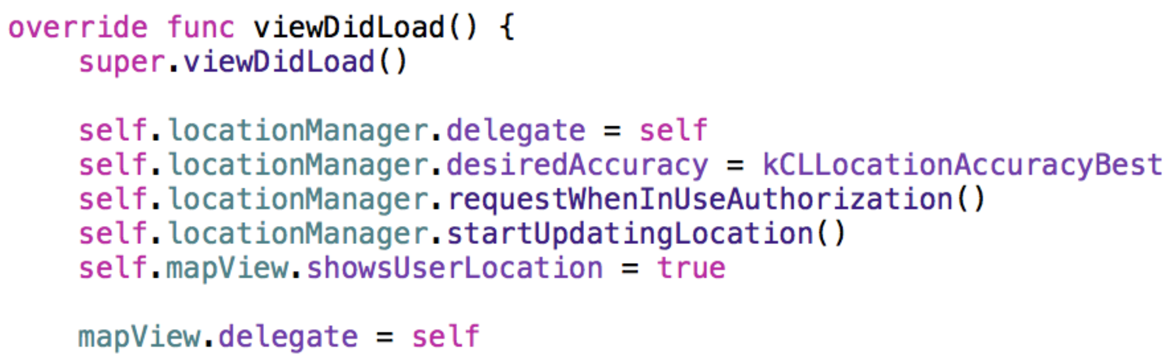
<u>Importing Frameworks:</u>

In order for each scene to use its required frameworks you must import them in the scenes corresponding controller as seen below


```swift

import UIKit
import CoreLocation
import MapKit
import Parse

```

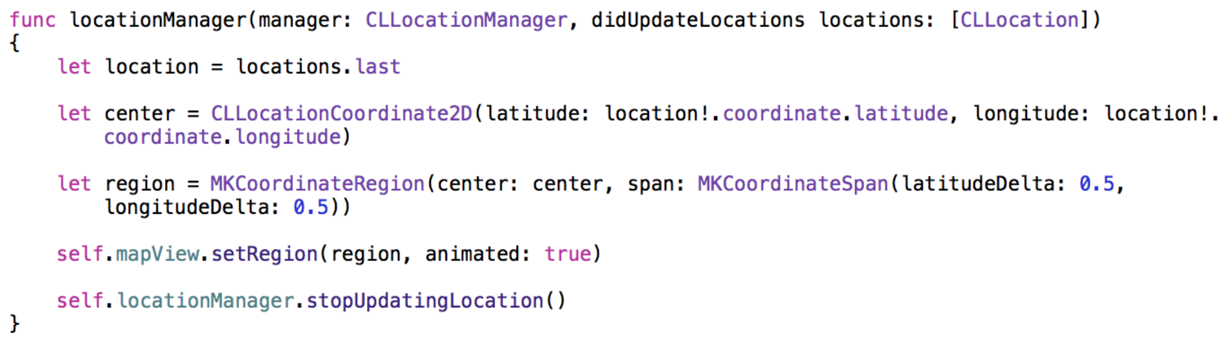
<u>Getting the users location</u>

Getting the users location is an essential part of this application, as the whole app is built around this key objective. To achieve this I first had to import Apple's CoreLocation Framework as well as conforming the ViewController to the CLLocationManagerDelegate Protocol. I then used the created location manager object to ask the user to enable location tracking when using the app and enable the user's location to be shown on the map interface.



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

    self.locationManager.delegate = self
    self.locationManager.desiredAccuracy = kCLLocationAccuracyBest
    self.locationManager.requestWhenInUseAuthorization()
    self.locationManager.startUpdatingLocation()
    self.mapView.showsUserLocation = true

    mapView.delegate = self
```

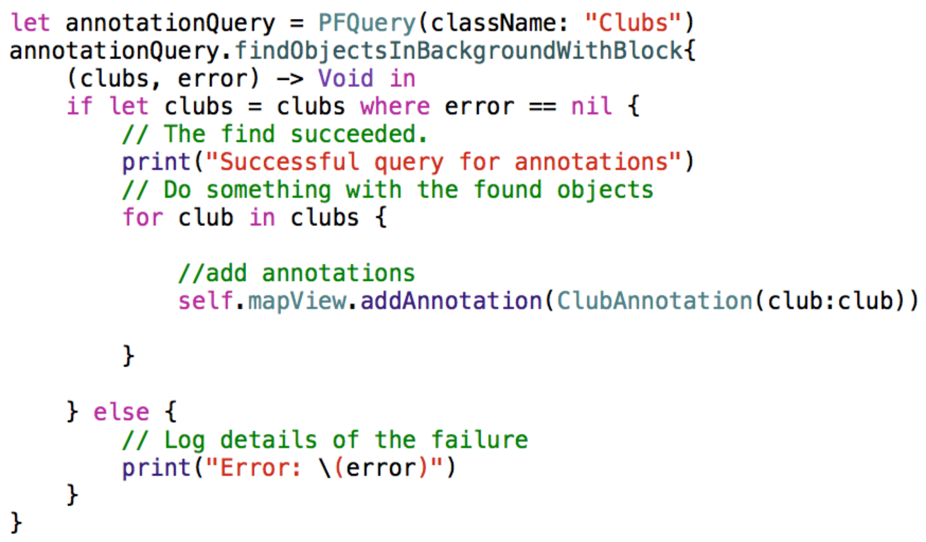
I then created a function to center the map around the user's location and zoom to the level I deemed most appropriate. The user's location is then stopped from updating.



```
func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
{
    let location = locations.last

    let center = CLLocationCoordinate2D(latitude: location!.coordinate.latitude, longitude: location!.
        coordinate.longitude)

    let region = MKCoordinateRegion(center: center, span: MKCoordinateSpan(latitudeDelta: 0.5,
        longitudeDelta: 0.5))

    self.mapView.setRegion(region, animated: true)

    self.locationManager.stopUpdatingLocation()
}
```

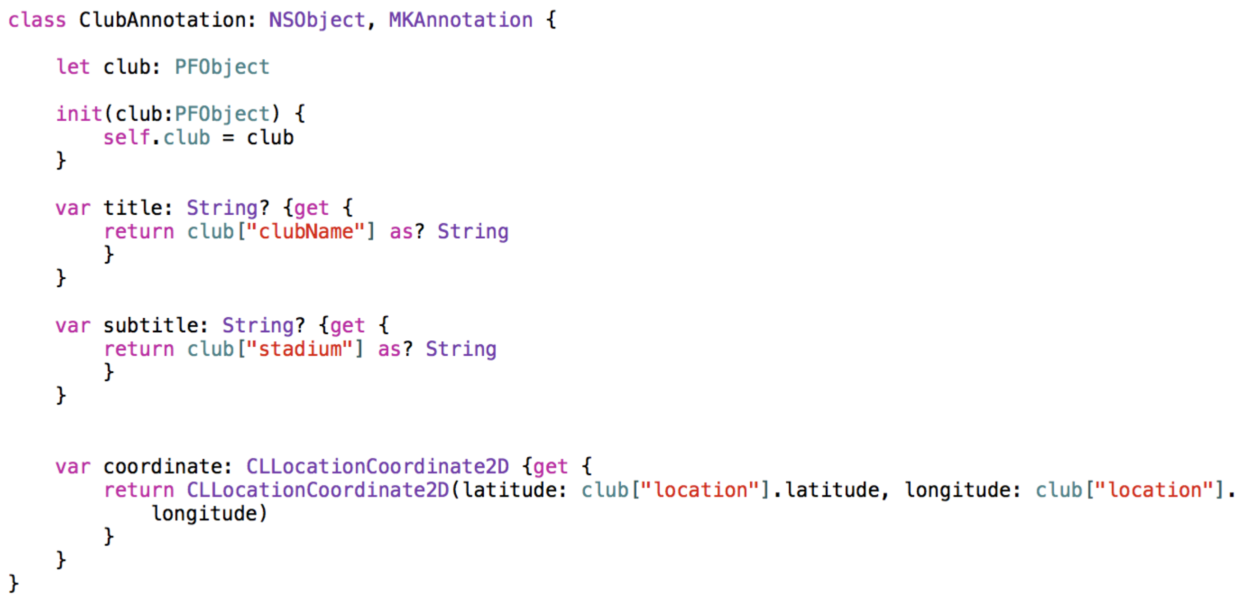
Query Parse and create map annotations for clubs:

In order to load and display all clubs from my parse database onto the map I first query my clubs database on Parse using the Parse frameworks PFQuery function. Once successful it then takes the clubs found, conforms them to my created class and then put them on the mapView as pin annotations.



```
let annotationQuery = PFQuery(className: "Clubs")
annotationQuery.findObjectsInBackgroundWithBlock{
    (clubs, error) -> Void in
    if let clubs = clubs where error == nil {
        // The find succeeded.
        print("Successful query for annotations")
        // Do something with the found objects
        for club in clubs {

            //add annotations
            self.mapView.addAnnotation(ClubAnnotation(club:club))

        }

    } else {
        // Log details of the failure
        print("Error: \(error)")
    }
}
```

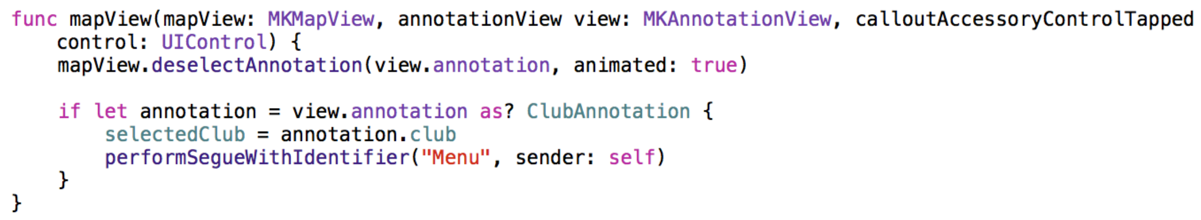
I then created a custom class (ClubAnnotation.swift). This is used to create an object conforming to Apples Mapkit MkAnnotation protocol to hold the queried clubs from my Parse database and display them onto the mapView interface created.



```
class ClubAnnotation: NSObject, MKAnnotation {

    let club: PFObject

    init(club:PFObject) {
        self.club = club
    }

    var title: String? {get {
        return club["clubName"] as? String
        }
    }

    var subtitle: String? {get {
        return club["stadium"] as? String
        }
    }


    var coordinate: CLLocationCoordinate2D {get {
        return CLLocationCoordinate2D(latitude: club["location"].latitude, longitude: club["location"].
            longitude)
        }
    }
}
```

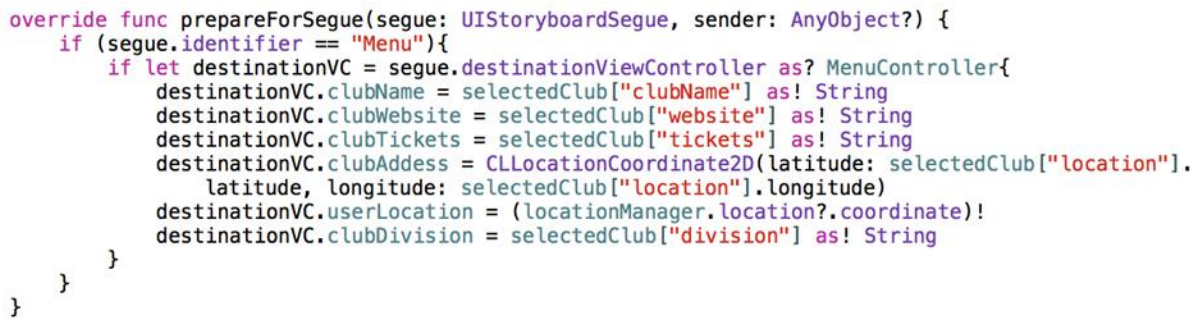
<u>Passing club and data to next view:</u>

An important aspect of my application was to allow the user to select a club on the map interface and select a more info button to bring them to the next menu page allowing the user to choose options relevant to the club selected. In order to do this I had to figure out a way to pass the selected club and its data onto the next view. I first created a button within the annotation view, once tapped this would then open the next scene. I also created a selected club object to which I made it equal whichever club was selected via the club annotation.



```swift
func mapView(mapView: MKMapView, annotationView view: MKAnnotationView, calloutAccessoryControlTapped
    control: UIControl) {
    mapView.deselectAnnotation(view.annotation, animated: true)

    if let annotation = view.annotation as? ClubAnnotation {
        selectedClub = annotation.club
        performSegueWithIdentifier("Menu", sender: self)
    }
}
```

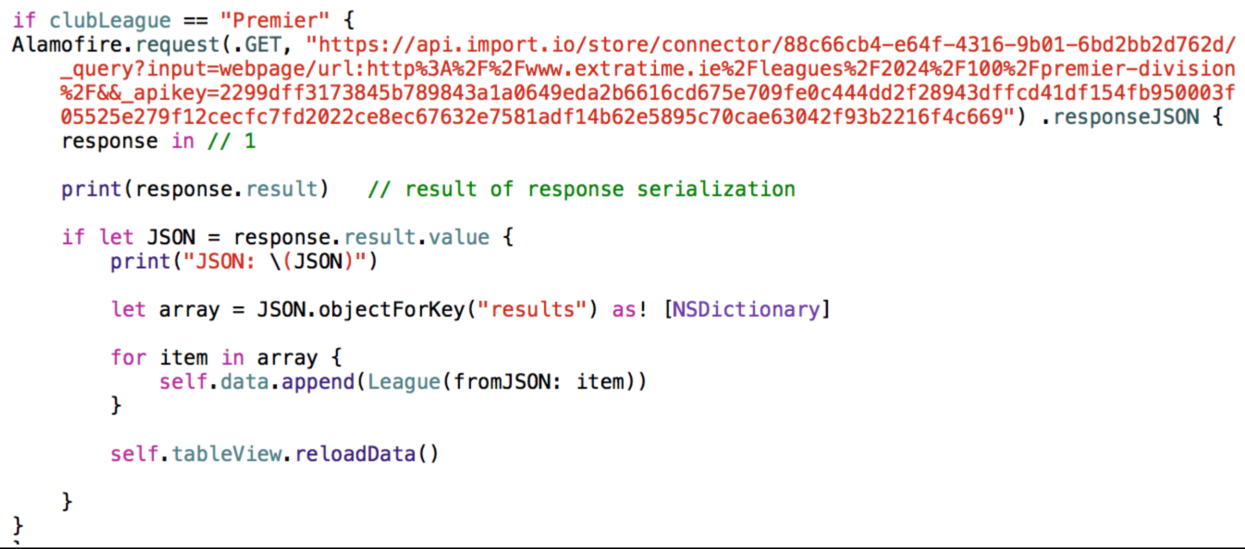
The next step was to use swift's prepareForSegue function in order to pass the selected clubs relevant data onto the next view. In order to pass each data object, I had to make sure to create a relevant data object in the next view controller to hold the incoming data. This function was used throughout the app to pass relevant data from scene to scene.



```swift
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if (segue.identifier == "Menu"){
        if let destinationVC = segue.destinationViewController as? MenuController{
            destinationVC.clubName = selectedClub["clubName"] as! String
            destinationVC.clubWebsite = selectedClub["website"] as! String
            destinationVC.clubTickets = selectedClub["tickets"] as! String
            destinationVC.clubAddess = CLLocationCoordinate2D(latitude: selectedClub["location"].
                latitude, longitude: selectedClub["location"].longitude)
            destinationVC.userLocation = (locationManager.location?.coordinate)!
            destinationVC.clubDivision = selectedClub["division"] as! String
        }
    }
}
```

<u>Using Alamofire:</u>

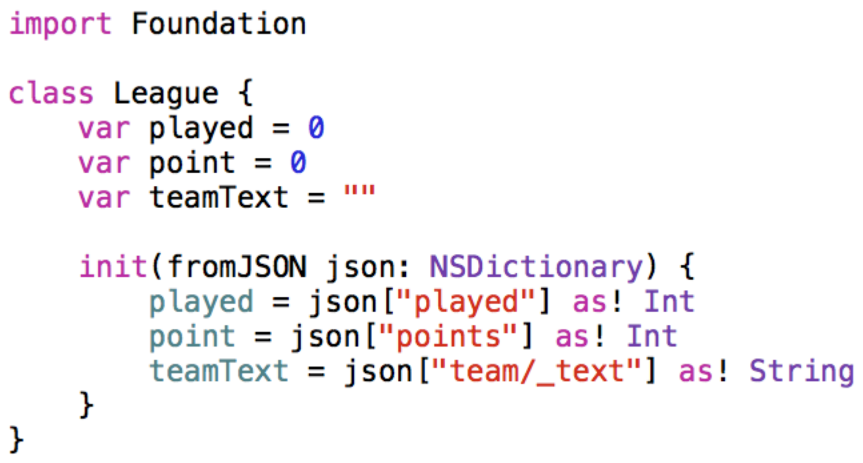
In order to get the relevant league table for the selected club, I used the Alamofire framework to make a HTTP Get request on my import.io API and return the response in JSON. I also placed the request in an if statements to take into consideration whatever league the selected club is in. This means the app will only get the league data for the league corresponding to the selected club rather than every league in my import.io API.



```
if clubLeague == "Premier" {
Alamofire.request(.GET, "https://api.import.io/store/connector/88c66cb4-e64f-4316-9b01-6bd2bb2d762d/
    _query?input=webpage/url:http%3A%2F%2Fwww.extratime.ie%2Fleagues%2F2024%2F100%2Fpremier-division
    %2F&&_apikey=2299dff3173845b789843a1a0649eda2b6616cd675e709fe0c444dd2f28943dffcd41df154fb950003f
    05525e279f12cecfc7fd2022ce8ec67632e7581adf14b62e5895c70cae63042f93b2216f4c669") .responseJSON {
    response in // 1

    print(response.result)   // result of response serialization

    if let JSON = response.result.value {
        print("JSON: \(JSON)")

        let array = JSON.objectForKey("results") as! [NSDictionary]

        for item in array {
            self.data.append(League(fromJSON: item))
        }

        self.tableView.reloadData()

    }
}
```

<u>Creating League table</u>

To create a league table for the received JSON league data, I firstly created a UITableView onto the league table scene. I then created a custom class for a league object to hold the relevant data received in JSON as a dictionary.



```
import Foundation

class League {
    var played = 0
    var point = 0
    var teamText = ""

    init(fromJSON json: NSDictionary) {
        played = json["played"] as! Int
        point = json["points"] as! Int
        teamText = json["team/_text"] as! String
    }
}
```

I then created a custom tableView cell in which I put three labels each of which corresponded to a club detail (name, played, points).

Finally I used each label in the custom cell to hold each clubs league details from the League class dictionary.

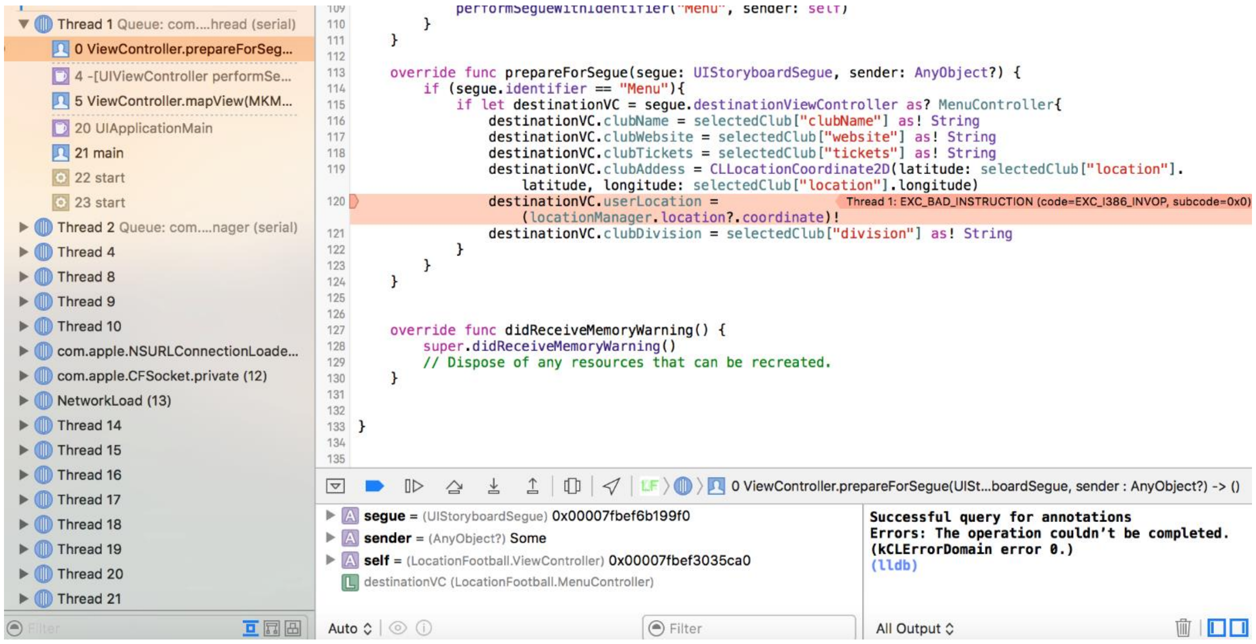
```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell
    {
    let cell = tableView.dequeueReusableCellWithIdentifier("Cell", forIndexPath: indexPath) as!
        CustomCell
    let row = indexPath.row
    cell.teamLabel.text = data[row].teamText
    cell.playedLabel.text = "\(data[row].played)"
    cell.pointsLabel.text = "\(data[row].point)"
    return cell
}
```

2.4. *Testing*

Software testing is a vitally important aspect in the software development lifecycle. In order to test effectively and efficiently, I dedicated a lot of time to it. These are the steps I took:

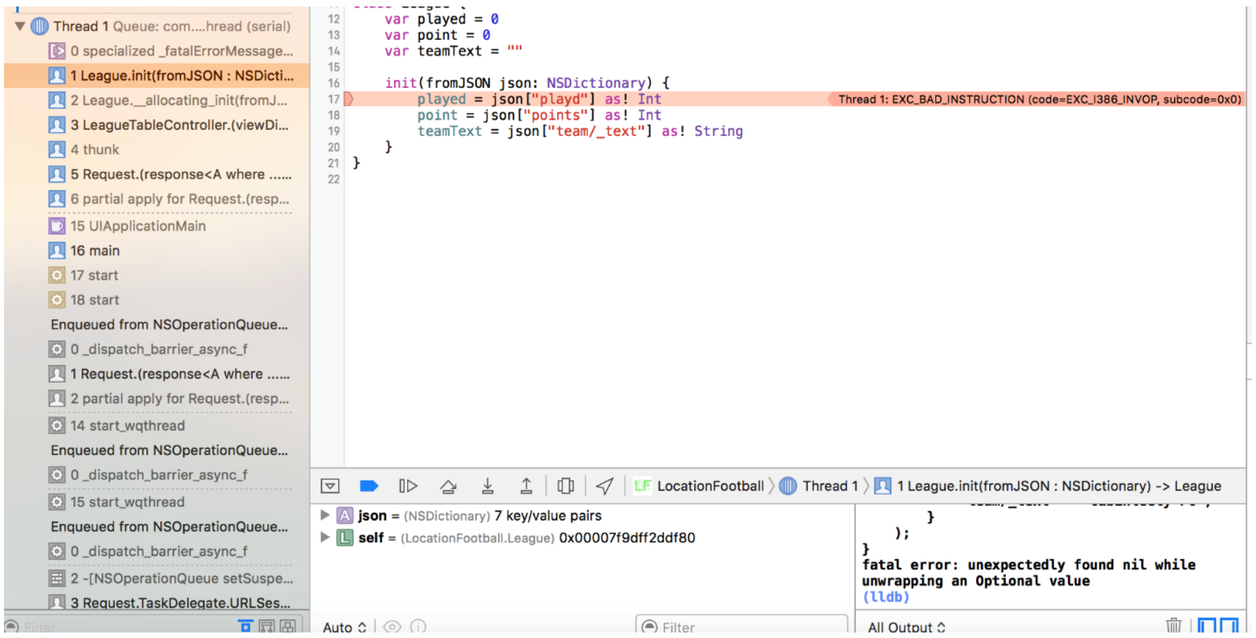
2.4.1. Unit Testing

Unit testing ensures that the functionality of each core component is correct. As I created each function or requirement, I carried out unit testing. This involved running the application on the Simulator and later on my iPhone and monitoring to see if any errors are triggered or any problems occur in Xcode's debug area. This was a hugely beneficial way of unit testing as it allowed me to correct mistakes early on in development and stop them from recurring later.


```
            performSegueWithIdentifier("Menu", sender: self)
        }
    }

    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        if (segue.identifier == "Menu"){
            if let destinationVC = segue.destinationViewController as? MenuController{
                destinationVC.clubName = selectedClub["clubName"] as! String
                destinationVC.clubWebsite = selectedClub["website"] as! String
                destinationVC.clubTickets = selectedClub["tickets"] as! String
                destinationVC.clubAddess = CLLocationCoordinate2D(latitude: selectedClub["location"].
                    latitude, longitude: selectedClub["location"].longitude)
                destinationVC.userLocation =          Thread 1: EXC_BAD_INSTRUCTION (code=EXC_I386_INVOP, subcode=0x0)
                    (locationManager.location?.coordinate)!
                destinationVC.clubDivision = selectedClub["division"] as! String
            }
        }
    }


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


}
```

```
0 ViewController.prepareForSegue(UISt...boardSegue, sender : AnyObject?) -> ()

segue = (UIStoryboardSegue) 0x00007fbef6b199f0
sender = (AnyObject?) Some
self = (LocationFootball.ViewController) 0x00007fbef3035ca0
destinationVC (LocationFootball.MenuController)
```

```
Successful query for annotations
Errors: The operation couldn't be completed.
(kCLErrorDomain error 0.)
(lldb)
```
Thread 1 Queue: com....hread (serial)
0 ViewController.prepareForSeg...
4 -[UIViewController performSe...
5 ViewController.mapView(MKM...
20 UIApplicationMain
21 main
22 start
23 start
Thread 2 Queue: com....nager (serial)
Thread 4
Thread 8
Thread 9
Thread 10
com.apple.NSURLConnectionLoade...
com.apple.CFSocket.private (12)
NetworkLoad (13)
Thread 14
Thread 15
Thread 16
Thread 17
Thread 18
Thread 19
Thread 20
Thread 21
Auto
Filter
All Output



```
Thread 1 Queue: com....hread (serial)
  0 specialized _fatalErrorMessage...
  1 League.init(fromJSON : NSDicti...
  2 League.__allocating_init(fromJ...
  3 LeagueTableController.(viewDi...
  4 thunk
  5 Request.(response<A where ......
  6 partial apply for Request.(resp...
  15 UIApplicationMain
  16 main
  17 start
  18 start
  Enqueued from NSOperationQueue...
  0 _dispatch_barrier_async_f
  1 Request.(response<A where ......
  2 partial apply for Request.(resp...
  14 start_wqthread
  Enqueued from NSOperationQueue...
  0 _dispatch_barrier_async_f
  15 start_wqthread
  Enqueued from NSOperationQueue...
  0 _dispatch_barrier_async_f
  2 -[NSOperationQueue setSuspe...
  3 Request.TaskDelegate.URLSes...
```

```swift
    var played = 0
    var point = 0
    var teamText = ""

    init(fromJSON json: NSDictionary) {
        played = json["playd"] as! Int          Thread 1: EXC_BAD_INSTRUCTION (code=EXC_I386_INVOP, subcode=0x0)
        point = json["points"] as! Int
        teamText = json["team/_text"] as! String
    }
}
```

LocationFootball 〉 Thread 1 〉 1 League.init(fromJSON : NSDictionary) -> League

```
▶ json = (NSDictionary) 7 key/value pairs
▶ self = (LocationFootball.League) 0x00007f9dff2ddf80
```

```
        }
    );
}
fatal error: unexpectedly found nil while
unwrapping an Optional value
(lldb)
```

2.4.2. System Testing

To test the system, I gathered a group of people with no programming experience and of the broadest possible customer base, I gave the selected group of people my application on a phone for 24 hours and performed black box testing with them by having them interact with the app. The results showed that all users successfully performed the tasks they were set and navigated efficiently throughout the application. I also asked them to rate the ease of use of each task and also the overall performance of the app.

The Results can been seen it the charts below:

Test Name	Try To Sign In with Wrong / No Credentials			
Before	Access To Internet Application is opened			
Steps	Enter random/no username Enter random/no password Select Sign In Button			
Expected	Alert will respond with error message corresponding to what was entered.			
Result	Pass	Pass	Pass	Pass

Test Name	Sign Up and Login	Ease of use 1-5	Comments
User 1	Pass	5	Very Simple and quick
User 2	Pass	3	Took a minute to navigate back to login
User 3	Pass	4	Good
User 4	Pass	3	A bit annoying having to log in after signing up

Test Name	Map Displays User location and clubs	Ease of use 1-5	Comments
User 1	Pass	4	Showed clubs and my location
User 2	Pass	3	Took a few seconds for the clubs to load
User 3	Pass	2	On first try map was empty but tried later and it worked ok
User 4	Pass	5	Showed both myself and clubs accurately and quickly

Test Name	Navigate to menu of a club	Ease of use 1-5	Comments
User 1	Pass	4	Moved quickly after pressing info button
User 2	Pass	3	Took me a few minutes to realise “I” was a button but work perfect when I pressed it
User 3	Pass	4	Worked well

User 4	Pass	4	Worked fine

Test Name	Navigate back and select different club	Ease of use 1-5	Comments
User 1	Pass	5	Back button responded quickly
User 2	Pass	4	Back button worked well
User 3	Pass	4	Back button worked well and quickly
User 4	Pass	5	Back button was clearly visible and worked perfectly

Test Name	Try each of the menu options for the selected club	Ease of use 1-5	Comments
User 1	Pass	4	All worked and provided the information for the selected club
User 2	Pass	3	All options worked but tickets and league table were slower to respond
User 3	Pass	4	All of them worked well and were in line with the club I selected
User 4	Pass	3	They all worked but the website and tickets options were a bit slow to load.

Test Name	Application Overall Performance	Ease of use 1-5	Comments

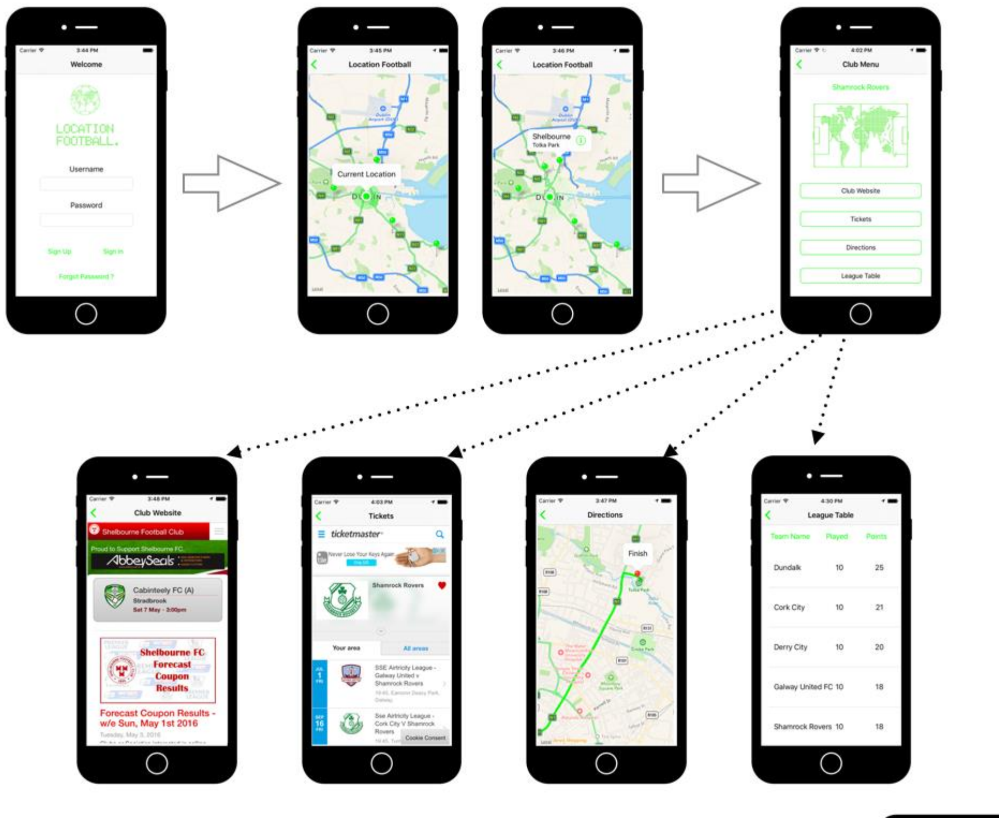
User 1	Pass	5	App worked quite perfectly , did what it was supposed to do and provided me with all the information for the football clubs around me
User 2	Pass	4	The app was good overall, all features worked and was generally quick to respond. Maybe change the info button to make it more obvious it’s a button.
User 3	Pass	3	The app worked fine except for when the clubs and location didn’t load which meant nothing else worked apart from the login at the time, once the clubs showed up it worked ok.
User 4	Pass	3	Everything worked well and good. Improvements could be made on the speed of the website and tickets options responding.

2.4.3. Screen Size Compatibility Testing

In order to test how the application looked on the various screen sizes and resolutions the iPhone comes in. I used the iPhone simulator to run the application on each of the devices released by Apple. Initially this took time to configure the app to look as it should on the various devices but after some constraint and frame altering and adjustments the application displayed on each screen size and resolution as it should have.

2.5. Graphical User Interface (GUI) Layout

On opening the application the user is required to sign in using valid user credentials or else sign up as a new user. Once logged in the user is brought to the main interface which consists of a large map showing the users location as well as showing football clubs as pin annotations. The user can then select these clubs and click the information button to bring them to the next menu interface. Here the user can select various options in relation to the club they selected on the map interface. The options included for the user to find information on the selected club include club website, league table, tickets and directions. There is a consistent green colour scheme throughout the application layout in keeping with the application logo and graphics in order to make the app more visually appealing and vibrant to the user. There is also a navigation bar throughout the app to help the user identify which scene they are currently on and allow the user to navigate back to previous scenes.



Team Name	Played	Points
Dundalk	10	25
Cork City	10	21
Derry City	10	20
Galway United FC	10	18
Shamrock Rovers	10	18

3. Conclusions

Success

Overall I have enjoyed developing this project. It has been very challenging and through my attempts to overcome these challenges, I have learned so much more about mobile application development particularly iOS development. I have also greatly improved my knowledge and understanding of programming languages specifically the Swift programming language. It has given me a huge boost of confidence that I could build an application like this in a short period of time while completing my other studies and balancing other activities.

Problems Faced

The main problem faced at the beginning of this project was my lack of programming skills. I was worried this would deeply affect my project and I would not be able to get a good grade. However after a lot of hard work and practice I continually gained confidence and began to take major leaps in my learning process and my project development.

Throughout my project build I would often get coding errors but I found using the internet as a way of finding out more about what each error means and how to fix them allowed me to get a greater understanding on the most common errors and also how to fix them quickly and efficiently. Other more complex errors took more time to evaluate but I found that taking a break and working on other requirements gave me a chance to take a step back and take a different approach towards the error the next time.

4. Further development

I intend on continuing to develop this project into a practical application that could be used in the real world. I will continue to tweak the application until it is visually and structurally perfect and ready to be introduced onto the AppStore.

I would also like expand the apps resources outside of Ireland and include maybe American or other countries football clubs.

With more resources and time I would be able to further the ability of allowing in app purchasing of tickets for selected clubs and matches. This would involve bringing in a third party API such as Ticketmaster or Eventbright in order to correctly distribute tickets for the respected football clubs.

I would also like to be able to generate a bit of revenue from the application from in-app advertising. To do this I feel the app first needs to first establish a greater user base and then take advantage of Apples iAd Framework which allows developers to set up add banners and pages within their application and gain revenue through user views and interactions.

With even further development I would expand the apps features to expand and provide more club information options such as club fixture lists and news feeds.

Finally if all the above is completed I would then consider releasing the application on other platforms such as Android and Windows.

5. References

App.pluralsight.com, (2015). *Build with Swift Pluralsight*. [online] Available at: https://app.pluralsight.com/library/courses/swift-ios-application-real-world/table-of-contents [Accessed 12 Dec. 2015 - Onwards].

App.pluralsight.com, (2015). *iOS Fundamentals Pluralsight*. [online] Available at: https://app.pluralsight.com/library/courses/ios-9-fundamentals/table-of-contents [Accessed 7 Dec. 2015 - Onwards].

App.pluralsight.com, (2015). *iOS whats new - Pluralsight*. [online] Available at: https://app.pluralsight.com/library/courses/ios9-whats-new/table-of-contents [Accessed 2 Jan. 2016].

Developer.apple.com, (n.d.). *iOS Developer Library*. [online] Available at: https://developer.apple.com/library/ios/navigation/ [Accessed 5 Dec. 2015 - Onwards].

Crew.co. (2016). *Should you build an iOS app or Android app?* | *Crew*. [online] Available at: https://crew.co/how-to-build-an-online-business/build-ios-app-or-android-app/ [Accessed 2 Jan. 2016 - Onwards].

AppCoda. (2016). *AppCoda Community - Learn iOS Programming and Build iPhone App*. [online] Available at: https://www.appcoda.com/ [Accessed 4 Jan. 2016- Onwards].

Raywenderlich.com. (2016). *Ray Wenderlich* | *Tutorials for iPhone / iOS Developers and Gamers*. [online] Available at: https://www.raywenderlich.com/ [Accessed 9 Nov. 2015- Onwards].

YouTube. (2016). *CodeWithChris*. [online] Available at: https://www.youtube.com/user/CodeWithChris [Accessed 7 Feb. 2016- Onwards].

YouTube. (2016). *GeekyLemon*. [online] Available at: https://www.youtube.com/user/GeekyLemon [Accessed 7 Feb. 2016- Onwards].

YouTube. (2016). *Jared Davidson*. [online] Available at: https://www.youtube.com/user/Archetapp [Accessed 5 Feb. 2016- Onwards].

YouTube. (2016). *London App Brewery*. [online] Available at: https://www.youtube.com/channel/UCVD5Vh9LhLBxp3o1vRNyf_w [Accessed 7 Feb. 2016- Onwards].

YouTube. (2016). *Vea Software*. [online] Available at: https://www.youtube.com/user/veasoftware [Accessed 7 Feb. 2016- Onwards].

GIMP. (2016). GIMP. [online] Available at: https://www.gimp.org/ [Accessed 2 Sep. 2015].

GitHub. (2016). Alamofire/Alamofire. [online] Available at: https://github.com/Alamofire/Alamofire [Accessed 2 Mar. 2016].

Import.io. (2016). Import.io | Web Data Platform & Free Web Scraping Tool. [online] Available at: https://www.import.io/ [Accessed 17 Dec. 2015].

Parse.com. (2016). Parse. [online] Available at: https://parse.com/ [Accessed 12 Sep. 2015].

Team, C. (2016). CocoaPods.org. [online] Cocoapods.org. Available at: https://cocoapods.org/ [Accessed 4 Mar. 2016].

6. Appendix

6.1. Project Proposal

Project Proposal

LOCATION FOOTBALL

Shane Noonan

x12435988

x12435988@student.ncirl.ie

BSc (Hons) in Computing

Specialisation: Networking and Mobile Technologies

Date: 28/09/2015

1. Objectives

Location Football will be a fun and easy mobile application that provides club information, league details and ticket booking information on football clubs based around your mobile location.

Objectives:

- To show local clubs on a map interface around your current location.
- To allow the user to select clubs to find out more information.
- To provide league table details on clubs.
- To provide a link to book tickets to see selected club.
- To provide directions to the desired club.

2. Background

I originally came up with this idea for my second year project and worked on a similar web application that showed all the top European clubs on a google map interface that provided their stadium location with a news feed and relevant external links. However I then released that the app was somewhat irrelevant as there are plenty of football apps covering the top clubs in the world. Then when on holiday in Asia and wanting to go see or even find out where and when their local football teams play, I found it messy and scattered to find the information necessary. This is when I thought about altering my second year project to be mobile and location based and to be more focused on match and ticket information and booking.

3. Technical Approach

I will begin by researching similar applications to see what’s already out there and what I could do to be different with my app. I will then research and decide on what approach to take in terms of technologies, libraries and api’s to use. After deciding, I will then begin to draw out storyboards before deciding on the look and feel of my app before beginning my app development. Also I will do two key testing phases mid-point and towards the end of the app completion. I will also upload reflective journals each month to track my progress and development throughout this project.

4. Special resources required

An Apple Macbook is required as it's the only device you can use the Xcode application to build an ios app. An Iphone will also be required to run the application to provide testing.

6. Technical Details

Swift, Xcode IDE. Parse.com SDK

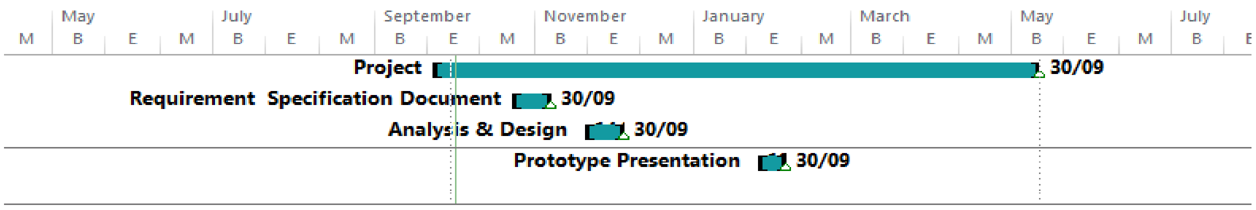
7. Evaluation

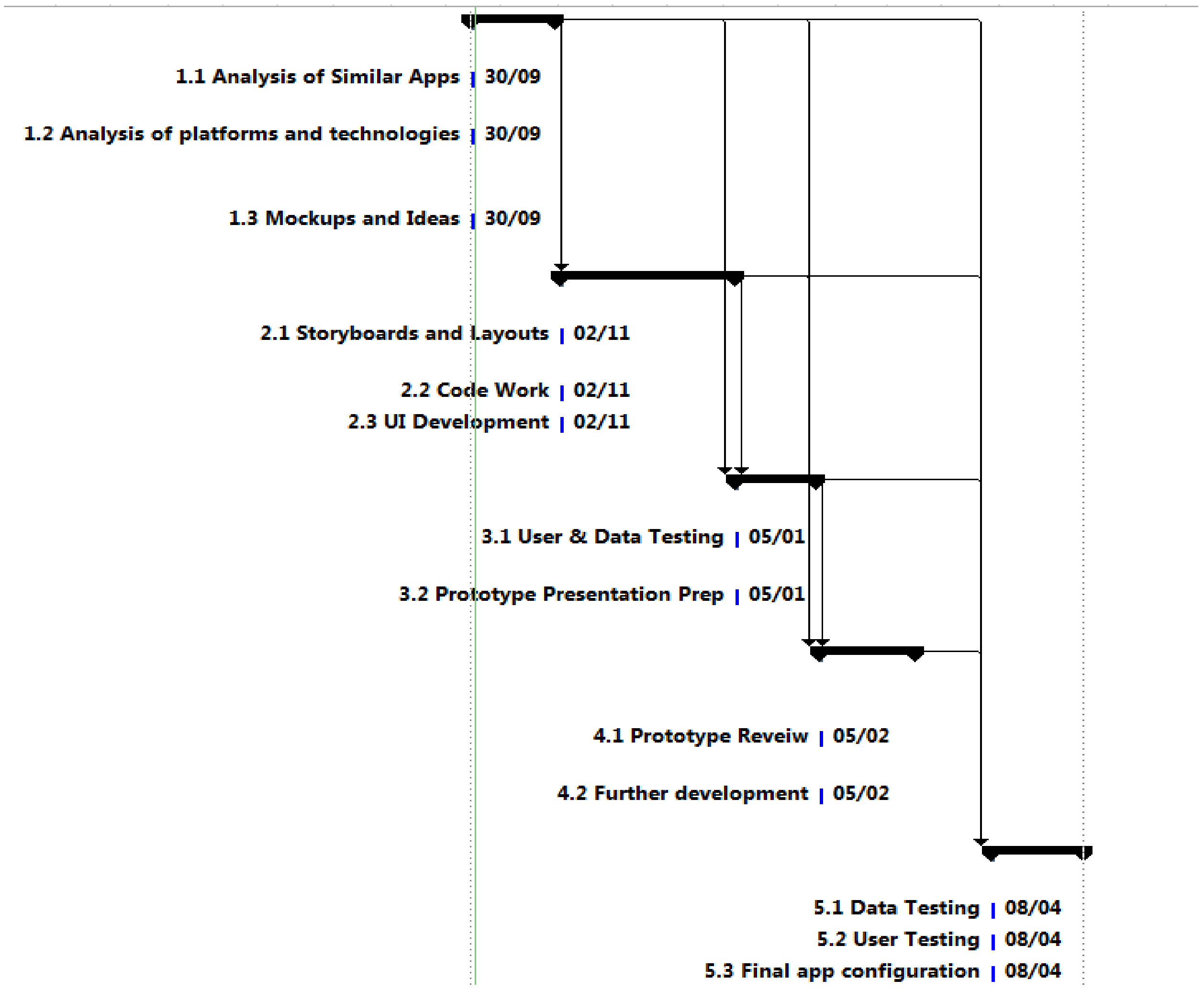
I will evaluate on a small scale using the ionic mobile simulator and my own Iphone by testing each section on the app after completion before moving on to the next section. I will then do a full mid-term evaluation before doing my mid-point presentation by giving the app to my friends and family to test and note any problems or changes they would consider for the app. I will then take these suggestions giving myself time to make any changes before my mid-point presentation.

I will then do a final evaluation after app completion to which I will again give the app to friends and family to note any problems or any suggestion they might have to improve the app. I will then give myself time to fix/make changes before submitting my project as complete on May 11th.

Shane Noonan. 02/10/2015

6.2. Project Plan





Task Name	Duration	Start	Finish
Project	166 days	Wed 23/09/15	Wed 11/05/16

Requirement Specification Document	11 days	Fri 23/10/15	Fri 06/11/15
Analysis & Design	11 days	Fri 20/11/15	Fri 04/12/15
Prototype Presentation	9 days	Mon 25/01/16	Thu 04/02/16
Stage 1 Preparation & Planning	23 days	Wed 30/09/15	Fri 30/10/15
1.1 Analysis of Similar Apps			
1.2 Analysis of platforms and technologies			
1.3 Mock-ups and Ideas			
Stage 2 Design and Development	46 days	Mon 02/11/15	Mon 04/01/16
2.1 Storyboards and Layouts			
2.2 Code Work			
2.3 UI Development			
Stage 3 Prototype Build & Testing	22 days	Tue 05/01/16	Wed 03/02/16
3.1 User & Data Testing			

3.2 Prototype Presentation Prep			
Stage 4 Final Development & Design	25 days	Fri 05/02/16	Thu 10/03/16
4.1 Prototype Review			
4.2 Further development			
Stage 5 Final Testing & Completion	24 days	Fri 08/04/16	Wed 11/05/16
5.1 Data Testing			
5.2 User Testing			
5.3 Final app configuration			

6.3. Monthly Journals

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: September

My Achievements

This month, I was able to decide on what mobile app I was going to do for my final year project and also decide what platform I would build it on. I also completed the required Project Proposal document.

I firstly spent a lot of time researching app ideas and seeing what similar app are already out there. Once finally deciding on an idea I then spent some time deciding on what platform to build my app on (Ios or Android). I decided to go for ios as I previously had a bad experience using android studio and building an android app and also because I currently use an Apple Macbook which enables me to download and use the Xcode application to build my ios app for free and also as I currently use an Iphone and so do many of my friends and family which I feel will benefit for the testing of my app.

My contributions to the project included beginning to learn how to use the Xcode application to build my ios app. I also began watching and learning from tutorials on building ios apps on various websites including Pluralsight. I also began drawing rough storyboards of how my app would look and work.

My Reflection

I feel, I have a serious amount of work to do to build this project with my current programming skills but that if I put aside time each week dedicated to just my project I will be successfully able to build a complete app. This month I feel I spent a lot of time going back and forth on finding an app I will be able to build with my skillset and also finding an idea that's something different than what's already out there.

I feel good that I have finally decided on an idea to run with and can now focus on learning the skills I need to be able to build it.

Intended Changes

I realised that I need to learn more about the swift and object-C programming languages as I have not worked with them before and my programming skills are very poor.

Next month, I will try and put a lot more hours into the actual app build and I will try to learn more about the swift programming language.

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: October

My Achievements

This month, I was able to get a start on some functions my application and also complete the requirements specifications document.

I decided to build my app through html, css and angularJS using the ionic framework instead of my original idea of building an ios app using swift. Ionic allows you to build cross platform web applications that can run on ios or android devices through html, css and angularJS. My reason for this is that I was struggling deeply with understanding and making any progress with the swift programming language. I am much more comfortable building it through ionic as I understand the code much more and I feel I'll be able to advance the development of my app much more through it.

I created a backend for my app using the Parse core platform. Here I created a database to handle user accounts and also a database to store the football clubs information. With that I managed to create a working register and login page for my application using the Parse JavaScript SDK and ionic.

My Reflection

I feel I now have a clearer picture of what my app will do and also what approach to take in order to be able to do it. I felt, it worked well to complete the requirements specification document as it gave a clear outline of each requirement in my application.

Intended Changes

Next month, I will try to develop my app further. I will start developing the main user interface which uses the google maps API. I realised that I need to find a way to get the football club locations from the database and display them on the google map interface.

I also have to start researching how I will integrate getting ticket information, getting live table stats and also how to integrate directions into my app.

Supervisor Meetings

Date of Meeting: Wednesday 21/10/2015

Items discussed: We discussed my overall project idea and what approach I'm taking in building the project. I felt it helped as I was a bit all over the place in terms of what languages, backend and tools I was going to use to build my app.

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: November

My Achievements

This month, I was able to further develop my app by improving the page routing and further develop the google maps interface.

My contributions to the projects included:

- I completed the first draft off my Analysis Design template.
- Improve the routing of my application.
- Further develop the google maps interface of the app.

My Reflection

I felt, it worked well to improve the routing of my app as I was having problems trying to find where code errors were happening. Now that I have the app nested well and routed properly, it is easier to find where features are going wrong and it's easier to edit and sort the problems.

However, I was not successful in trying to display club locations on the google maps interface. This is an important step in my app as once I can get the club data to display on the map, it should be a lot easier to exchange data between my app and my database.

I felt under a lot of pressure from the other modules this month as I had a lot of deadlines and submissions which resulted in me not focusing or doing as much work as I would have liked on my project this month.

Intended Changes

Next month,

I will try to do a lot more work on my project. I will try to display the club locations stored in my database on the google maps interface.

I will try to further develop my app for the prototype phase. I realised that I need to get at least one major function of my app to work to make a good prototype for the mid-point presentation.

Supervisor Meetings

Date of Meeting: 25/11/2015

Items discussed: We discussed my submitted Requirements Specification document and went through what will have to be changed and improved. I also had some questions with regards to the Analysis Design which we discussed and went over.

Reflective Journal

Student name: Shane Noonan

Programme (e.g., BSc in Computing): BSHC

Month: December

My Achievements

This month, I was able to begin work on the development of my apps code.

I've managed to implement the login page successfully and am now working on the main interface which is working out well so far. I've successfully queried my database so am now working on displaying my data on the main interface and developing my app further.

My Reflection

I felt, it worked well to research similar problems people have faced with their code online and try and see if I could use it to help my problems encountered. It also helped to follow tutorial videos online to further my understanding and learning of the technologies and code I'm using for my app.

However, I was not successful in getting past the initial main interface load and will have to now develop the next stage of a menu and user interaction.

Intended Changes

Next month, I will try to further develop my application in preparation for my mid point presentation.

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: January

My Achievements

Over the past month and a half I took a major leap in the development of my project. Over the Christmas break I decided to change platform and language I will be using to develop my application. I decided I would now build the application using Apples swift programming language and the Xcode IDE. My reasons for this was because I hit a major stumbling block using the Ionic Framework as I felt it wasn't fully developed for Ios development as I got many errors and problems when testing the app. After deciding to change to Apples own ios development suite with swift I began working really hard to get to grips with the swift programming language and xcode.

My contributions to the project after learning the ios development pathway are:

I have built a working prototype of my project application which allows the user to see many features of how the final app will work.

I have completed my mid-point technical report ahead of my mid-point presentation next week.

My Reflection

I felt, it worked well to watch learning videos on the swift programming language and also on ios app development. These really helped me get a start on building my app and how to implement it.

However, I was not successful in getting some features of the application working. I still have a bit of work to do on some features of the application

Intended Changes

Next month, I will try to develop the app even further and also start making it look more visually appealing.

I feel I still need to continue my learning of ios development and the swift programming language.

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: February

My Achievements

This month, I was able to progress my project even further by addressing some of the problems outlined at the mid-point presentation. As well as advance my learning of ios development.

My contributions to the projects included adding a navigation controller to the app, getting two of the main requirements to work correctly, improving the UI and testing the application on actual ios device.

My Reflection

I felt, it worked well to continue learning and researching ios development and swift learning paths and tutorials as they giving me new ideas to improve my project as well as continue my learning.

I feel the mid-point presentation gave me a good indication of what I still have to do.

However, I was not successful in completing the last two requirements of my application.

Intended Changes

Next month, I will try to take a major leap in my project and complete the final two requirements. I feel this is important as I still have lots more work to do including testing, the final report and finalizing the app for the presentation. I also intend to meet with my supervisor more often as I feel it would be beneficial for the run in to the final submission.

Reflective Journal

Student name: Shane Noonan

Programme: BSHC – Networking & Mobile Technologies

Month: March

My Achievements

This month, my progressing with the project has slowed mainly due to the fact we had so many other module projects and assignments due this month. However I did manage to progress my project a little by getting the directions requirement completed which had been troubling me for a while. This means I only have one more requirement to complete before my project is pretty much ready apart from some minor details.

My Reflection

I felt, the massive workload of other modules has slowed my progression and will continue to until after my exams.

Getting the directions requirement to work was a huge weight of my shoulders as it checks one more thing of the list and gave me a boost to continue progressing with this project.

However, I have not yet been successful in completing the last requirement of my application which I am now getting a little worried that it is not possible to achieve. I intend on making a league table that updates itself by scraping the internet or using an API but so far I cannot figure out a) how to create a league table in iOS and b) get the necessary data. I've looked everywhere online for help on creating a league table in iOS development and cannot find anything of the sort. Also I've emailed many football API providers and either got no response or else I was quoted a massive fee for using their API. I intend to discuss what to do with my supervisor after my exams but until then I will continue trying to find a solution.

Supervisor Meetings

Items discussed: We discussed what I had achieved so far and what more I had left to do. I also had some questions with testing which we discussed and went over.

Intended Changes

Next month, I must complete the final requirement in my project. I feel this is vital as I still have a lot work to do including testing and the final report before my project is due. I also intend to meet with my supervisor again in order to get further advice and clarity on what I need to get done.