iOS Report

Carlos Roldan
car23@aber.ac.uk

December 1, 2018

Abstract

This document is an assignment for the module CS39820 of the computer science department in Aberystwyth University. This paper is written in a subjective format. The aim of this document is to back up the code presented in the assignment; design and build an app in swift to help the user learn vocabulary in the foreign language of her/his choice.

Content

Introduction

This assignment contains different elements that enhance valuable development experiences such as UI design, backend development, extensions, and unit tests. The main focus on this assignment has been producing a professional experience prioritising simplicity, therefore I provided less than 200 lines of code per each file while having the minimum screens for the user. Whenever I have the opportunity for reducing complexity, I made use of refactoring or re-implementation.

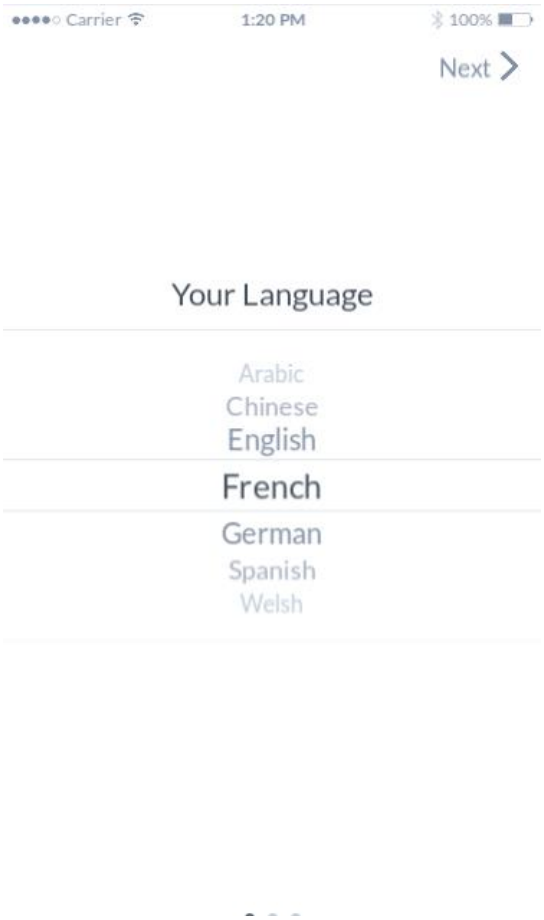

Figure 1 - Mock-up - Inital screen 1/3

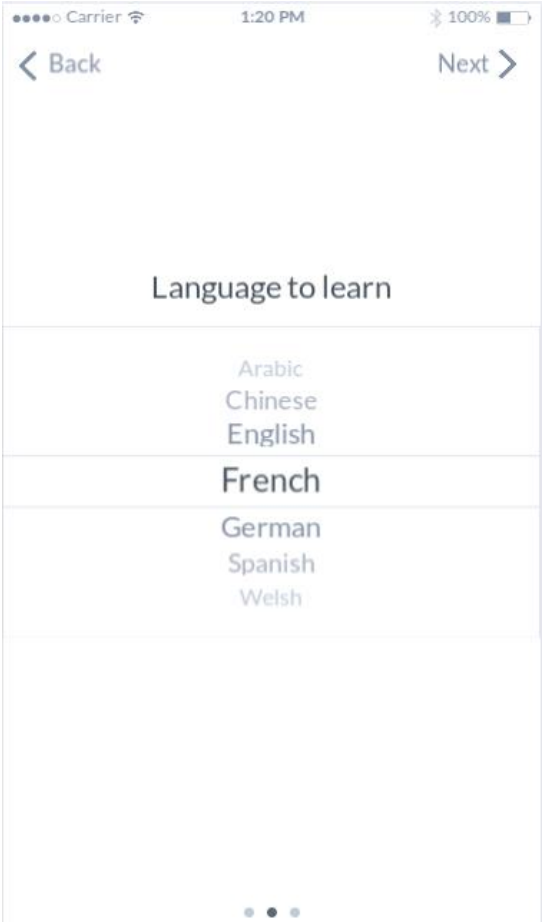

Figure 1 - Mock-up - Inital screen 2/3

Design

When I read the assignment requirement, I visualised the app in my mind, however, this was not enough, so I decided to use marvelapp.com in order to create mocks up to follow an initial design pattern.

In diagram 1 and 2, I implemented choose pickers to select between a set of languages. From the beginning, I wanted to start with a selection of languages which will be displayed later on in the main screen. Nevertheless, in development, I decided to implement the two chooser pickers together to improve the UX. Additionally, in the app, the initial mock-up for screen 1, 2, three are compressed in the first screen.

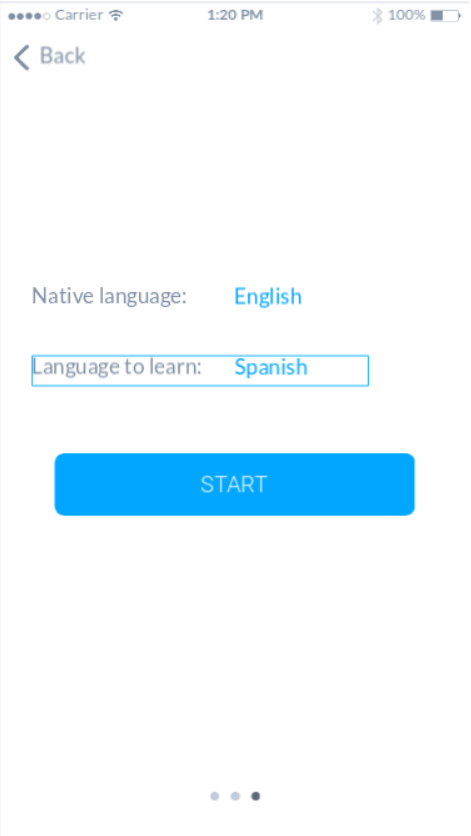

Figure 3 - Mock-up - Inital screen 3/3

Figure 4 - Mock-up - Home screen 1/2

In the main screen, I first designed a table view with words typed by the user with their respective definitions as well as a bar menu in the bottom. As an initial design, it was interesting. However, in development, I realised

I had to implement an external library for the dictionary or dictionary of words for each selected language. Moreover, creating the bar menu would also represent an additional high time-consuming feature. By doing that, it would increase the complexity, and I wanted to keep it as simple as possible. Hence, the current design does not involve any definitions, and each cell from the table view contains only the words typed by the user together with their translation, both separated by a '='.

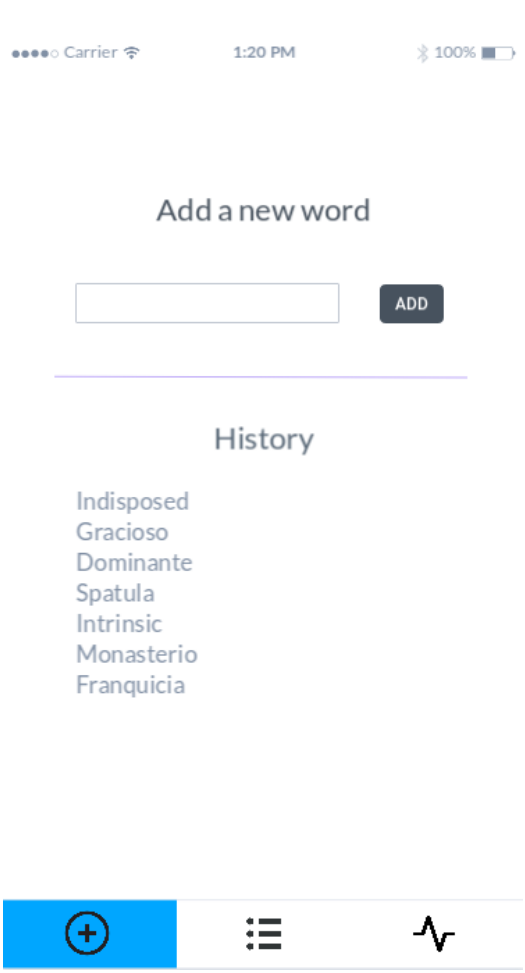

Figure 5 - Mock-up – Home screen 2/2

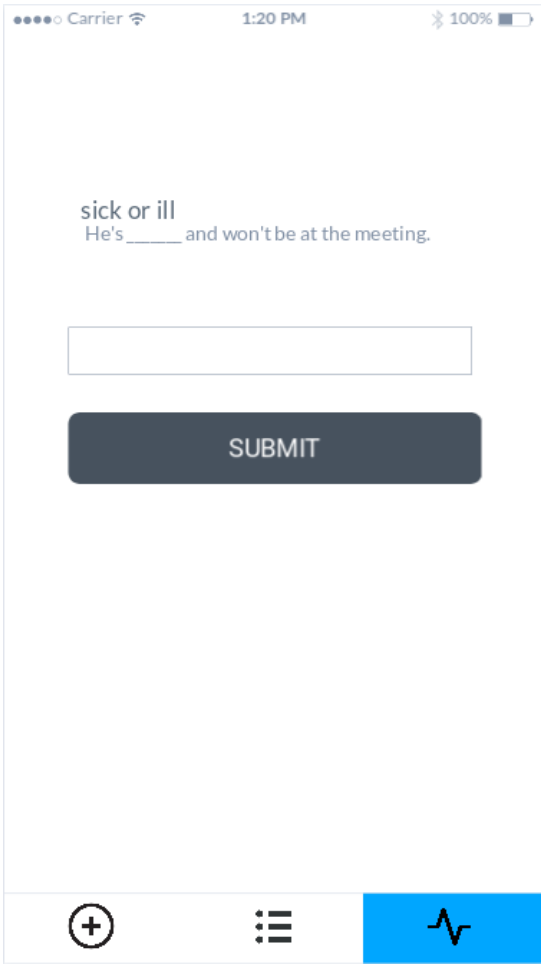

Figure 6 - Mock-up – Game/Testing screen 1/3

I wanted to keep track of each word entered most easily. Although, I did not implement any history or additional screen for such a feature. I decided to list every word in the table view, and when a new word is entered, an alert displays asking for its translation. Once the word with its translation is introduced, they are stored in the device and displayed in the table view which is instantly reloaded.

The game/test screen has an entirely different design than the initial designed, as the way for testing users was initially conceived based on definitions. Now the game/test screen is based on spelling or character hints to reduce complexity and improve the user experience.

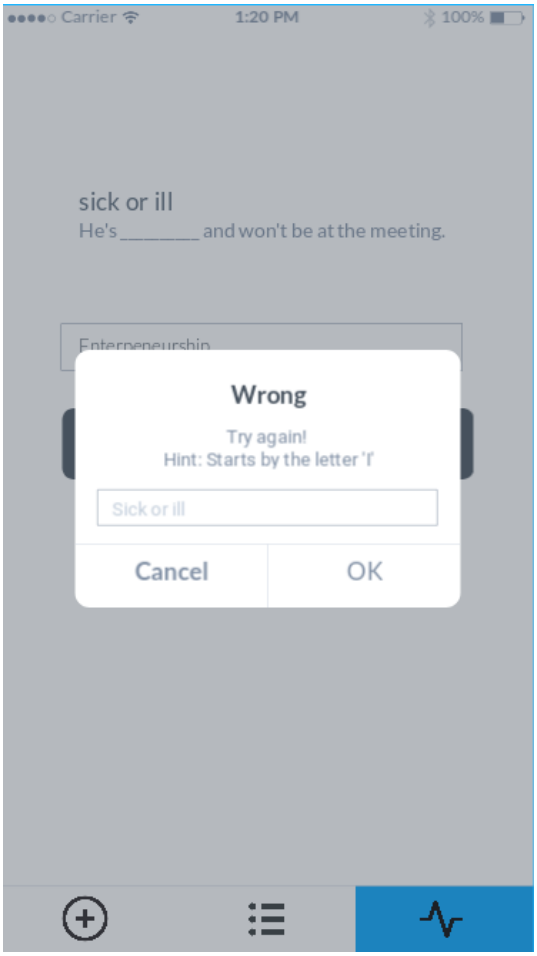

Figure 7 - Mock-up – Game/Testing screen 2/3

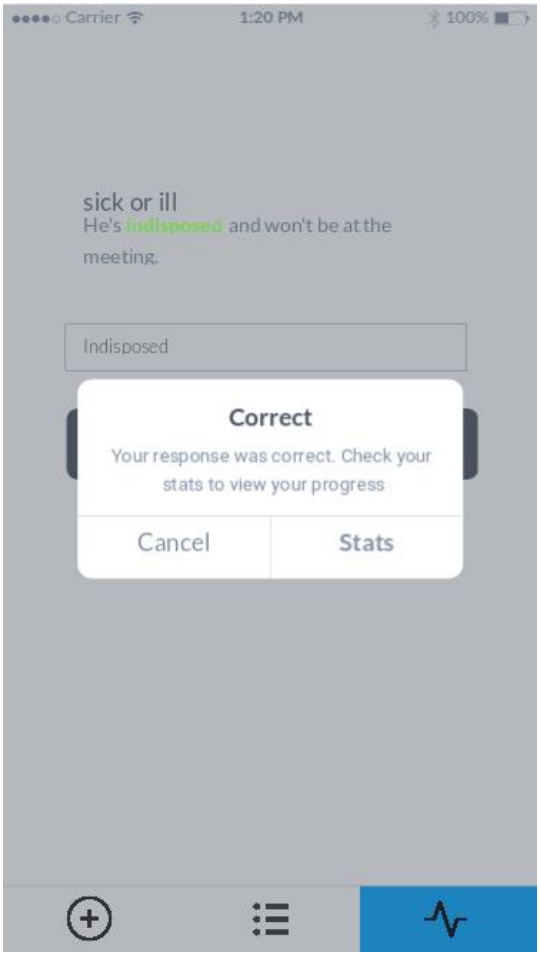

Figure 8 - Mock-up – Game/Testing screen 3/3

I also implement a point reward schema to calculate how well the user does with its vocabulary. If the user uses many hints, he receives fewer or no points. The same happens if the user types an incorrect translation. Now, if the user types the correct translation, he gets the total amount of points per round.

There is a maximum of 10 rounds to test the user's vocabulary. After it is finished, the user returns to his main screen in which the user can see how many tests have completed.

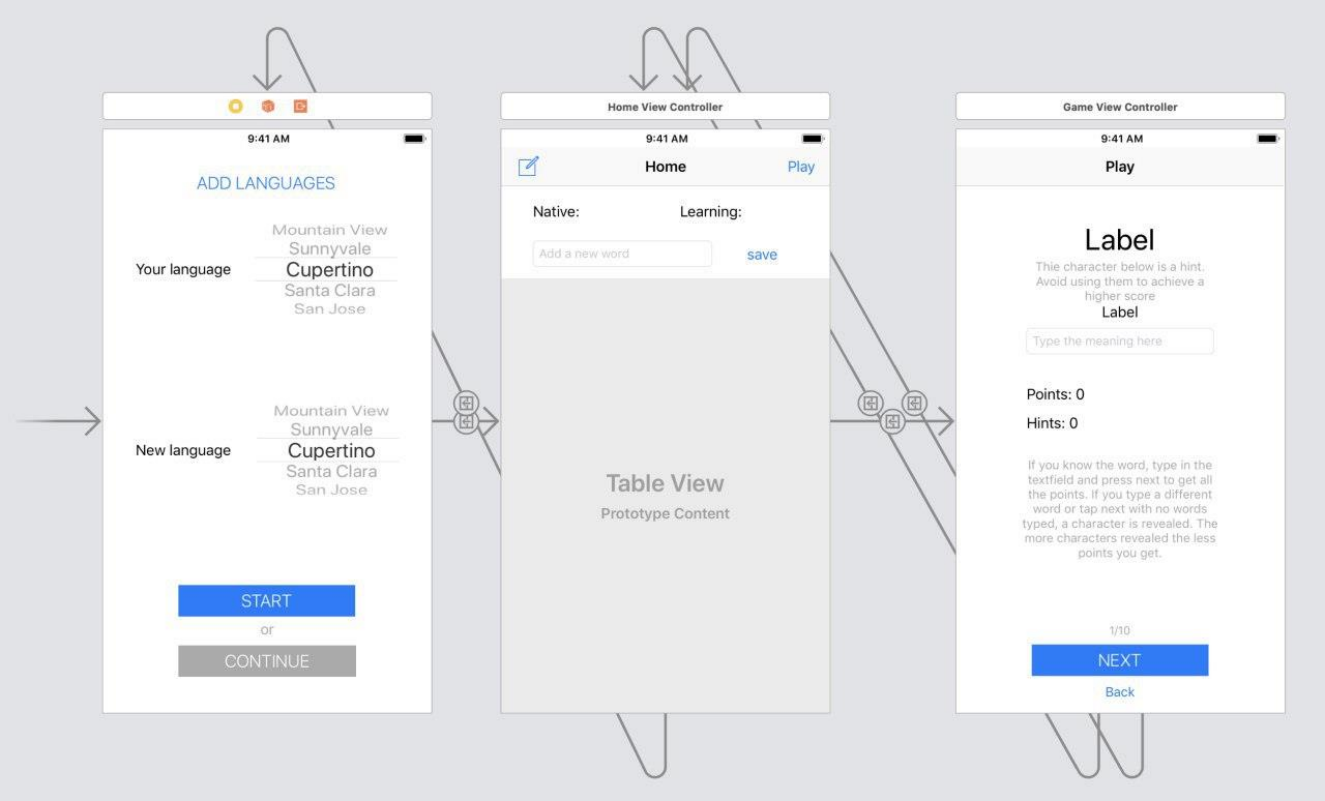

Figure 9 – Design – Complete design stack

Technical Details

Initially, I designed the mocks up with a different logic, based on simple arrays. Where a single array had all the words entered separated by a =. The problem presented with this logic was the complexity on selecting single words within the array, as I first had to investigate each element in the array, and the into each character to find the character = and delete the rest of the string and append that single word into an array.

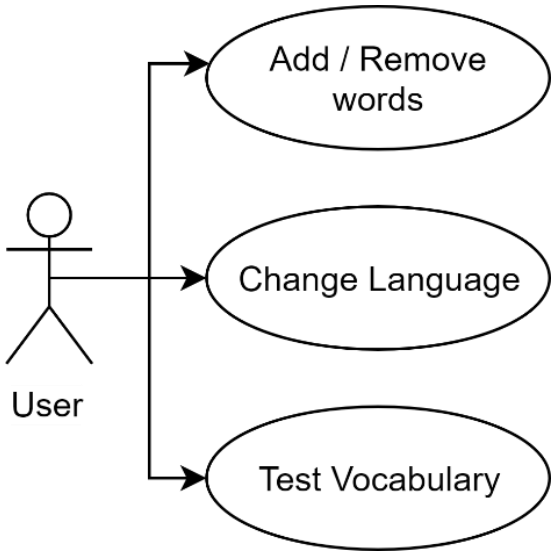

Figure 10 - Design – UML Use Case diagram

I initially achieved it, but obtaining the second word from the same element was overwhelming as seen in the screenshot below.

```
func getCleanArray(){
    //  instantiate counter in -1 to coun with the space before the "="
    var counter = -1

    //  instantiate variables to store single words
    var cleanWord   = ""
    var foreignWord = ""

    //  for each word in the array
    for word in wordsArray {
        //  for each character in the word
        for character in word {

            //  start the counter
            counter = counter+1

            //  if the character selected is "=" stop
            if (character == "=") {

                //  make use of the extension utils to delete the rest of characters except the word
                cleanWord = word.substring(toIndex: word.length - counter)

                //  make use of the extension utils to delete the rest of characters except the meaning word
                foreignWord = word.substring(fromIndex: word.length - counter)

                //  append word without spaces to the clean array
                cleanArray.append(cleanWord)

                //  append the meaning word to the foreign array
                foreignArray.append(foreignWord)

                //  stabilize the counter for next iteration
                counter = cleanWord.length - word.length

                //  print array for testing
                print(cleanArray)
                print(foreignArray)
        }
    }
}
```

Figure 11 - Screenshot – Challenging algorithm for obtaining a single word ussing a String extension

I solved this issue by using dictionaries as my primary data model. Therefore, I have one dictionary from the primary language as key and the new language as value, and another one, the opposite order.

At the beginning, I decided to implement core data to store values in the device. However, due to complexity on soring arrays and dictionaries natively, I decided to go for a more efficient and straightforward solution to store this data model. I stored my data models in

a JSON file from a dictionary directly using a JSON serializer. There was no complexity during this process.

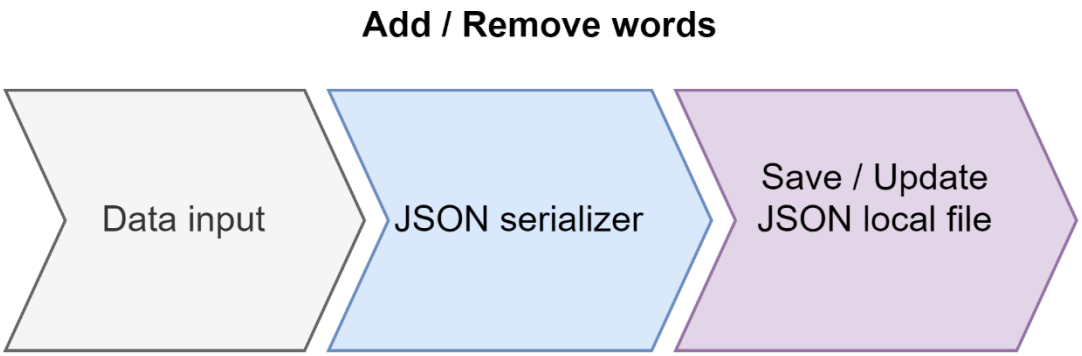

Figure 12 – Design– Add-Remove words logic

I also added an extension to improve my Strings class within my xcode project. I added custom methods to improve the relation between String – Char. This implementation was very handy and useful in many occasions, as seen in the screenshot below

```
//
//  Created by Carlos Roldan on 20/11/2018.
//  Copyright © 2018 Carlos Roldan. All rights reserved.
//

import Foundation

extension String {
    var length: Int {
        return count
    }

    subscript (i: Int) -> String {
        return self[i ..< i + 1]
    }

    func substring(fromIndex: Int) -> String {
        return self[min(fromIndex, length) ..< length]
    }

    func substring(toIndex: Int) -> String {
        return self[0 ..< max(0, toIndex)]
    }

    subscript (r: Range<Int>) -> String {
        let range = Range(uncheckedBounds: (lower: max(0, min(length, r.lowerBound)),
                                            upper: min(length, max(0, r.upperBound))))
        let start = index(startIndex, offsetBy: range.lowerBound)
        let end = index(start, offsetBy: range.upperBound - range.lowerBound)
        return String(self[start ..< end])
    }

    func replace(target: String, withString: String) -> String {
        return self.replacingOccurrences(of: target, with: withString)
    }
}
```

Figure 13 – Screenshot– String extension

UI Test

My main concern about UI testing was asserting some UI elements were existing, when previously required UI elements where tapped. Therefore, I carried two main tests:

- The first UI test was focused on the first screen. When the user is required to select two languages to start, I needed to make a set of UI elements existed based on the user input. As you can see in the screenshot, when the user taps the same language, there is an alert displaying that explains the user that has to pick two different languages. However, when the user chooses two different languages, there is another alert with the buttons to either restart the vocabulary or continue with the same vocabulary.

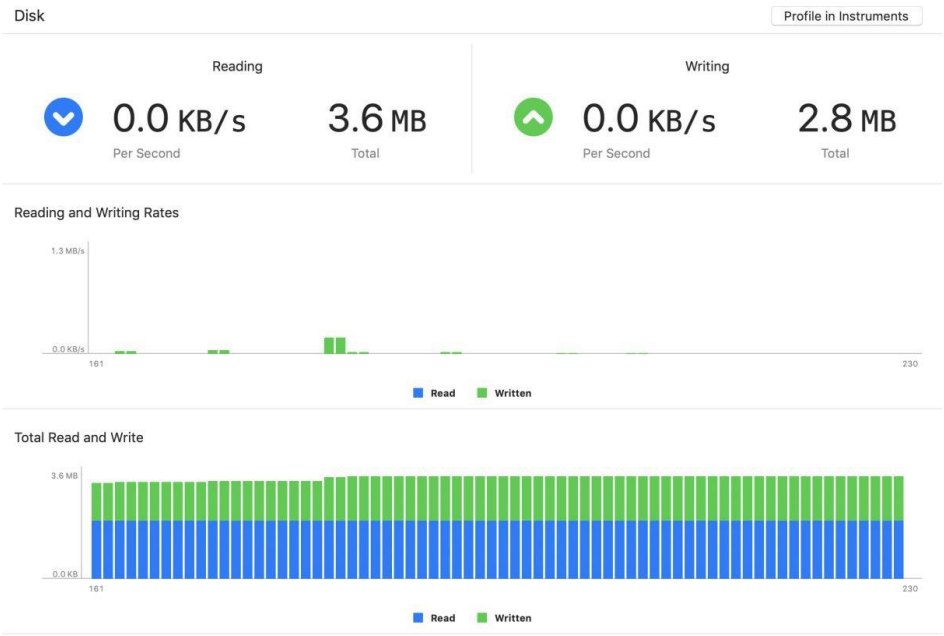

Figure 14 – Performance– Read & Write test

- The second UI test was focused on the second screen, and it was a relevant test as not only was I testing if required UI elements existed when others were selected, but also, I was testing if I could reach the next screen. As a result, if this test successfully concluded, it also meant the access to other screens with segue potentially work.

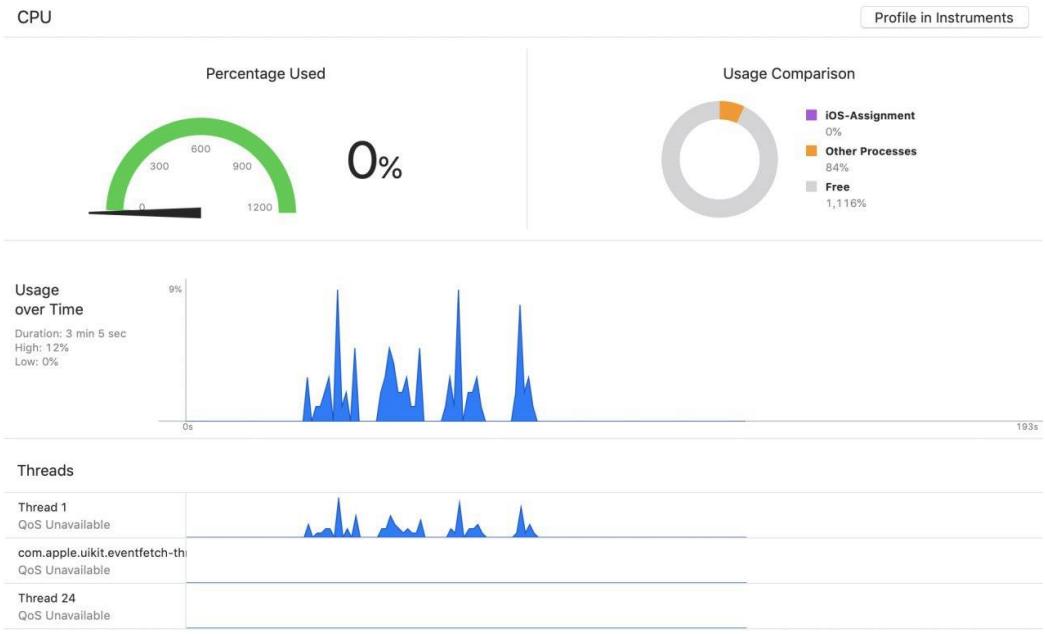

Figure 15 – Performance– CPU usage test

Unit Test

I was not very focused in the unit test of my assignment due to my preference to conduct debugging as a manual unit testing. Nevertheless, in order to show knowledge about unit testing, I tested the function to print a message to the user based on how well he/she has done in the vocabulary test. I did not import any testable class, I just inserted the function in the testing class due to its low complexity. As you can see in the screenshot, based on a given score and an array of messages, the function uses the method assertEqual() to test the message is equal to the given score.

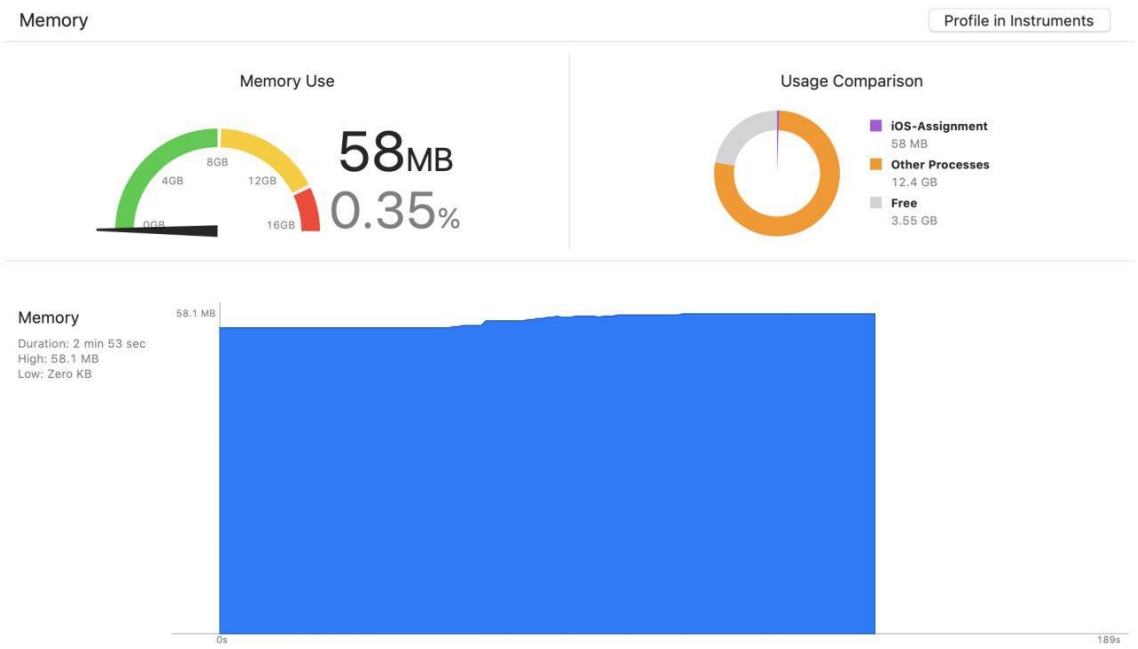

Figure 16 – Performance – Memory usage test

Conclusion

To conclude, the requirements were interesting and challenging. I had so much fun completing this assignment. It has improved my knowledge over the swift development. I aim for a 76%. I have completed the requirements given and commented the code. I have made use of refactoring as good code practice. My unit testing has not been great due to my preference on debugging manually. I believe my report contains detailed documentation about testing, design and technical aspects of my assignment, including and introduction and conclusion. I do not believe this assignment will help for my major projects due to my selection on the networks topic, however, it will be crucial for my professional career in terms of mobile development.