SEM2220 Assignment Two
Report

Assignment Two - iOS Language application

Submitted in partial fulfillment of
the requirements for the award of the degree of

MEng
in
Software Engineering

Submitted by

crh13 Craig Heptinstall

Department of Computer Science
Aberystwyth University

20th April 2016

Chapter 1

1.1 Introduction

This report describes the stages of implementation and testing of an iOS application written in swift that helps a user learn and revise welsh. The app the brief asked for, required the user to be able to:

1. Enter, edit and delete word phrase pairs (To be stored persistently)
2. Search the list of phrases in the app
3. Import phrases from an online url (json format)
4. Revise a number of words based on the list the user has entered

To break down the meaning of a phrase, this needed to describe a term in both welsh and English, alongside the type of word (noun, verb or unknown), a note to help explain the phrase, and a set of tags to describe what the phrase would be used for.

Alongside these main requirements, the user interface had to be as friendly as possible, making sure users are given a means of navigating between different scenes, and ensuring that where the user inputs or edits information that conforms to IOS suggestions.

1.2 Design aspects considered

Before beginning the implementation of the app, two major decisions had to be made. Firstly, designing the application to be as easy to use and self explanatory in terms of what options and inputs to be shown on certain pages across the app.

To help decide, a few tutorials and apple guides were consulted, such as the iOS human interface guidelines site [1]. The guideline firstly gave good insight into the three main categories of structuring an app: hierarchical, flat and content driven. In this case, flat was chosen because of its description as being good for smaller sets of requirements where major content can be accessed directly and therefore quicker. Because the app being created would not be too complex, a flat, tab bar navigation style was selected.

Alongside the reasoning already provided, choosing the tab based navigation means meant that the user should always know where they are in the app. For instance, they would find it easier to access the list of phrases easier if they decide the revision quiz has pairs in it that are not correct. As for any inputs, search facilities, or table management of phrases, it was decided that these should all be implemented with the provided tools in XCode and swift. Even though many libraries across the internet offer extra UI elements, taking into account the time available to complete the application, and the desired simplicity of functionality, these were not considered. This goes the same for colour schemes and fonts, where these were designed to be kept to default standards to give the users a feel of remaining in the usual iOS environment.

1.3 Creating the application

Upon starting development, creating the key navigation views was first. By placing views for the phrase list, revision view and import views, the tab bar navigation was able to be created. The tab bar navigation was one of the simplest steps of the implementation process, and by

again using online tutorials based from the apple developer guide [2], this was done in little time.

Core data was selected following this, and was chosen over SQLite simply due to the usefulness (as mentioned in a software development article [3]) that database tables could then be connected to objects, and stored easier. By making use of the NSManagedObject class, it meant that after designing the database (see figure 1.1) to the needs of the brief, object classes could be directly implemented.

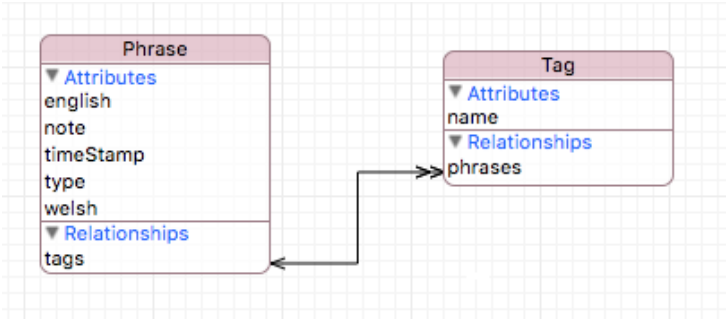

Figure 1.1: The one to many relationship between phrases and tags used. This allowed each phrase to have tags assigned when creating or modifying them.

Once the database was correct and connected to objects in the application, creating the functionality to add phrases and display them was undertaken. Again, as this quite a new topic to learn, the Apple developer guides were great help again [4]. The table view was created, and the add, edit and delete functionality was created. Although it worked, at this stage, it was not using core data, but a simple array. Once the functionality was working, core data then replaced the temporary array to make data more static.

Next was designing the storyboard to allow users to navigate editing and adding using the same view. By doing this, the amount of code and classes was reduced, and it also meant it followed guidelines again. The figure below (figure 1.2) shows how the same segue was used for both editing and adding new phrases.

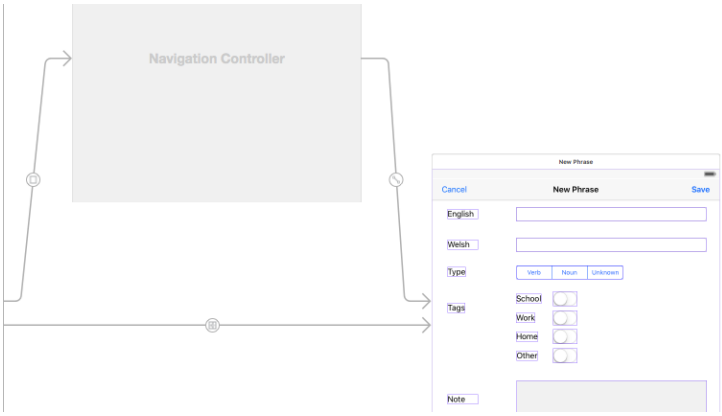

Figure 1.2: Storyboard showing the two connecting means to the Phrase details view. Via the navigation controller allowed editing, whilst the shorter path opened the same window for new phrases.

The inputs used for editing and adding included simple text views, with a text field for the

note, and a switch for the type of word. The switch proved useful in restricting users to the finite options that should have been shown. All of the elements in the view were easy to pull data from, and also to set (when editing a phrase).

Following the views allowing the user to add and edit phrases, the function to delete was then added. Once again, by using the guide available online from Apple, this was done in very little lines of code. Once phrases were available to the user, the quiz functionality was then added. In the quiz tab, by getting all of the latest top 10 added phrases (whenever a phrase was added in the phrase view, the current time was assigned within the core data table), and then shuffling these, a scrollable picker was added to allow the user to choose from 5 random answers for the opposite language. In order to shuffle the questions randomly, a built in game library was used as part of the code, specifically the GamePlayKit [5].

Import and search facilities were added following this, the second of which required the use of a tutorial. Alongside a plain search box, options were added below the search bar in order to allow users to search for one type of language (English or Welsh words), or by all terms.

For parts of the application that required users to see when an action was complete (such as importing words), notification pop ups were added.

In order to put to use what was learnt over the past course, and to follow good coding standards, the life cycle methods in swift were used to ensure data was always loaded and up to date in the veiws, from the viewDidLoad method, to the viewWillAppear (used for checking the amount of phrases available before generating a quiz).

As suggested by the brief, and in good practice, xCode was used for this application, and the code was well commented and structured following swift coding practises.

1.3.1 Issues encountered and solutions

As with any project, there were some unexpected difficulties having used some of the coding concepts for the first time. The first of these was getting core data hooked up to a table view. In previous examples worked on prior to this project, local arrays had been used to get data for table cells, whilst doing this using core data proved more difficult. The main issue was on editing phrases. Before reading a tutorial on how to fix this issue, the controller which looked after populating the table had been sending the selected managed object to the edit controller, which then attempted to save the phrase there. The problem here though meant the context on which it was being saved under was the not the original one from the table controller, therefore the changes were lost.

To fix this, a variable is stored in the table controller with the managed object to be edited, and any changes are first sent back from the edit controller before the variable is edited and saved to the correct context.

The same problem occurred when searching and attempting to edit. Because search results were being presented as a separate array of filtered results, on attempt to edit these, only the filtered results array would have been edited on save, therefore core data was not able to see the changes. Again, a similar fix was applied.

One final issue was that when changing or importing words, if a user left the quiz tab halfway through a quiz, words may have been out of date or old. To fix this, the quiz was reloaded, including reloading phrases to avoid old or incorrect revision data.

1.4 Testing and running the app

Throughout the creation of the application, the iOS emulator was sufficient for checking the layout and functionality. The logging within xCode was also a great tool for finding out any issues, and helped identify the causes. xCode helped a great deal with layout issues, by providing options for adding constraints to storyboard views to ensure thy fit in the device screen.

1.4.1 Emulating the application

The images following show the application running in the emulator. Alongside these (and in detail of all features), the narrated video can be found inside the source folder of this project.

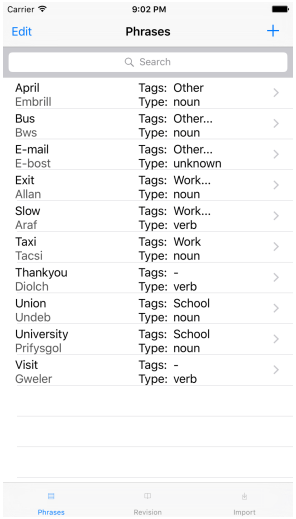

Figure 1.3: A list of phrases shown in the first tab.

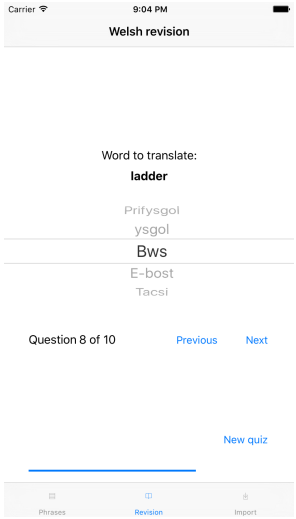

Figure 1.4: The revision quiz, showing the picker, and progress bar of the quiz.

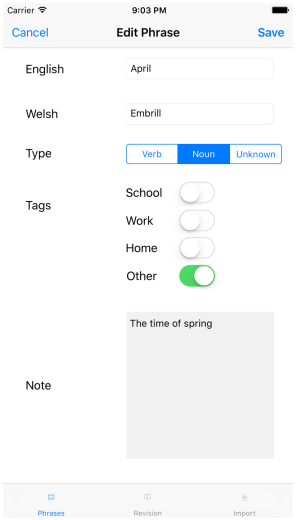

Figure 1.5: Showing the addition of a new phrase, with GUI controls.

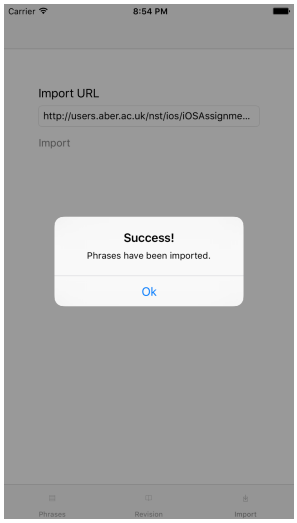

Figure 1.6: Success message and import tab.

1.4.2 Testing

In addition to the emulating and debugging performed, usability tests have been performed using the GUI, and has been compiled into the below table 1.1. The test, expected result, actual result and status of the test have been recorded.

Table 1.1: GUI Test talble

Test #	Test	Expected Result	Actual	Pass?
1	Open app initially	Phase list shows, with no entries	As expected	Pass
2	Attempt to start quiz	Message appears telling user to have 10 phrases	As expected	Pass
3	Add new phrase with correct data	Entry list updates showing data	As expected	Pass
4	Add new phrase without english text	Save button remains grey, cannot add	As expected	Pass
5	Add new phrase without welsh text	Save button remains grey, cannot add	As expected	Pass
6	Add new phrase with more than one tag	Tags show as 'Tag name...' in phrase list	As expected	Pass
7	Add new phrase with no tags	Tag show as '-' in phrase list	As expected	Pass
8	Add new phrase without note	Tag saves fine, and notes are still empty	As expected	Pass
9	Add phrase with same details as another	Phrase adds fine, and shows separately in list	As expected	Pass
10	Search phrase using english scope, when results should be present	Results show relating to search term	As expected	Pass
11	Search phrase using welsh scope, when results should be present	Results show relating to search term	As expected	Pass
12	Search phrase using note scope, when results should be present	Results show relating to search term	As expected	Pass
13	Search phrase using all scope, when results should be present	Results show relating to search term	As expected	Pass
14	Search phrase using all scope, when no results should be present	No results are shown	As expected	Pass
15	Edit item in phrase list, change name	Name change is shown in phrase list	As expected	Pass
16	Edit item in phrase list, change name to empty	Save button is disabled until name is not empty	As expected	Pass
17	Edit item in filtered phrases	Correct phrase is opened and edited	As expected	Pass
18	Delete item in phrase list	Item is removed from phrase list	As expected	Pass
19	Click on quiz when 10 or more phrases are present	Quiz shows, and picker is populated with first question	As expected	Pass

20	Select an answer and click next	Picker answers change, question phrase changes, and status of question number increases	As expected	Pass
21	Click previous when on q 1	Previous q button is grey	As expected	Pass
22	Click on previous when previously answered	Answer picked earlier should be selected	As expected	Pass
23	Click on complete to show score	Score should be shown, including advice, and button to restart	As expected	Pass
24	Click on new quiz button	Quiz restarts, with different/ differently ordered questions	As expected	Pass
25	Change tab and go back to quiz	Quiz restarts with different/ differently ordered questions	As expected	Pass
26	Change tab from quiz, edit phrase and go back to quiz	Quiz restarts with different/ differently ordered questions	As expected	Pass
27	Click import tab and add working address. Import	Import success alert message, phrases are added and list tab updates	As expected	Pass
28	Import invalid address	Alert stating error of no data	App crashes	Fail
29	Import valid address, with invalid json data	Alert stating error of no data	As expected	Pass

1.5 Conclusion and further work

The completed product fulfils the brief, though due to time constraints, there were parts of some functionality which could have been taken further. For instance, adding more unique and dynamic tags, allowing more options within the quiz tab, and also providing more ways to search through phrases. The app is clean and simple, and is very easy to use and navigate, with the use of the tab system. Testing was performed to a good standard, although this can always be bettered, such as introducing swift tests, or automated GUI testing. To fix the error in the above test table, a simple fix was applied to the import controller code to check if data was brought back from the given address. This then allowed the alert to be opened. Following the work performed, I can now provide personal scores and reasons for the amounts:

Topic	**Score**	**Reason**
Documentation	14/20	Both design and implimentation was well explained. Images of app layout and CoreData helped show the structure of the application. The documentation was within the maximum word limit, and explained in good detail the different phases of the project.
Implementation	55/70	Although dynamic tags were not completed, in the time availible the application fulfilled the brief provided. Code was organised and well commented, using guidlines provided on both the apple site, and on various tutorials online. Phrases could be searched various ways, and imported easily.
Testing	8/10	Use of the simulator to test the application, with a comprehensive test table showing all GUI aspects covered.

Overall mark: 77%

References

[1] Apple, *Human interface guidelines*, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/Navigation.html, 2016.

[2] Apple, *Tab bar controllers*, https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/TabBarControllers.html, 2016.

[3] P. Rea, *Getting Started with Core Data Tutorial*, https://www.raywenderlich.com/115695/getting-started-with-core-data-tutorial, 2015.

[4] Apple, *Creating CoreData objects*, https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CoreData/CreatingObjects.html, 2016.

[5] Stackoverflow- R. Roe, *Shuffle an array in swift using GamePlayKit*, http://stackoverflow.com/questions/24026510/how-do-i-shuffle-an-array-in-swift, 2015.