A wearable Device Software to Support Presentations

Final Report

Master of Information Technology

Student: Zicong Li

Student ID: 807298

Supervisor: Jianzhong Qi

School of Computing and Information Systems

The University of Melbourne, Australia

Abstract

When a speaker wants to show something in presentation, a presentation program may help the speaker illustrates the point smoothly. The best example is Microsoft PowerPoint, people can prepare their slides in this program with diagram, multi-media etc.

Presentation always coming with many audiences, the speaker feel nervous is a very common thing. This project aim to build a wearable device software to help speakers feel more confident during presentation. This system could display the words on the smart watch and control the progress of speech. It has a great practical value. Practically, the project has a variety of applications especially in university, for example, it could help students prepare presentation of project.

I certify that
- this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- where necessary I have received clearance for this research from the University's Ethics Committee (Approval Number ....) and have submitted all required data to the Department
- the thesis is 3324 words in length (excluding text in images, table, bibliographies and appendices).

[illegible]

Acknowledgement

This report would not exist if not for the support and guidance of a special group of people who will always have my deepest appreciation.

I would like to thank Dr. Jianzhong Qi, my project supervisor, who has offered his technical supports and guided me into this project.

I would like to thank my parents, they gave me spiritual encouragement and financial support.

Context

Figures

Tables

1. Introduction

As a student, presentations are not strange during the study period. Presentation could help people illustrate their thoughts to others. Also, it is a good approach to show the work to audiences. A good slide always be defined by some rules. For example, keeping text to a minimum, that means leave the key points on the slides not the whole sentence in presentation draft.

1.1. Motivation

Presentation always coming with many audiences, the speaker feel nervous is a very common thing. The worst case happening in presentation is forget words. In the research at the University of Chicago shows worries undermines our working memory, and working memory is a short-term memory system that could deal the problems at hand immediately. [1] In other words, there exists a vicious spiral model, the people feel nervous then they forget the words, because they forget the words, they feel more nervous.

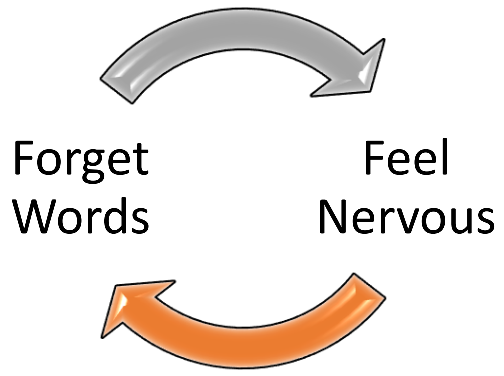


Figure 1-1-1 Vicious Spiral Model

Smart watch is the state-of-the-art technological product. Some famous technology company, such as Apple and Samsung, release smart watch in recent years. In the meantime, as the technology gets mature, accordingly, more and more wearable device applications were developed for all aspects of life. In this project, all software development is based on iPhone and iWatch which was programmed in swift language.

Figure 1-1-2 Apple Watch Series 1 and Samsung Gear S3 Smartwatch [2] [3]

This project introduces the implementation of a wearable device software which can help the speaker performance better during presentation. This introduction provides a brief description, the objectives and requirements analysis of this project.

1.2. Objectives

The goal of this project is to implement a smart watch application to support presentations.

Specifically, the objectives of this project are defined as follows:

- Learning wearable device programming: Once the latest swift version releasing, there are many different places comparing with old version. As the beginner, the student need to learn a new programming language, and adapt new integrated development environment (IDE) which is Xcode.
- Architectural Design: This section describes the structure, behavior and more views of the system. It links all the elements comprising the system such as software and hardware components, by using the relations among those elements.
- Physical Design: Physical design contains two parts, iOS and watchOS. In each part, user interface (UI) design and process design are required. UI design in concerned on the communication between user and system. And process design focused on the data moving through the system.

- Logical Design: It contains data flows, inputs and outputs of the system. In this project, Core Data is an Object-oriented database, which is built in Xcode
- Implement Model View Controller (MVC) framework: The student need divide the system into three interconnected parts. In this project, student could use MVC to make the system a lot smarter.
- Program Implementation: The most significant section in the project, the student starts to write code for all function based on the requirement.
- Program testing and debug: A unit testing is performed after implement a function of the system. An integration testing is performed after the whole system is accomplished.
- Demo: A oral presentation was required to show the work in this semester. In the demo part, the system should work as a presentation assistant.
- Finish a report: A report with all the details of the project is to be finished.

1.3. Risk Assessment

As mentioned above, this project is to apply a state-of-the-art technology in real problems. There are several potential risks that probably affects the progress of the project. These risks are listed in Table 1-1, and further explained as the following:

- Theoretical Challenge: There are two aspects of theoretical challenges from programming and design. This project requires a thorough understanding of Swift and Core Data. The project poses a non-trivial challenge to the student. Without hard study and investigation into a new programming language, the project easily fails.
- Financial Problem: Xcode is an IDE only for macOS. The project requires an Apple computer to implement the system. The project need spend a lot of money on devices include Mac, iPhone and iWatch. This is again a non-trivial requirement.
- Time Limitation: The duration of the project is only one semester. The student has to spend two months on developing a relatively mature software.

Priority	Risks
1 (highest)	Theoretical Challenge
2	Time Limitation
2	Financial Problem

Table 1-1 Table of Prioritized Risks

These risks have different likelihood and impact on the project, which are shown in Table 1-2

	Low Likelihood	Mid Likelihood	High Likelihood
Low Impact			Time Limitation
Mid Impact	Theoretical Challenge		
High Impact		Financial Problem	

Table 1-2 Impact and Likelihood Analysis of Risks

1.4 Summary

Corresponding to the objectives and tacks of the project, this report is organized as follows:

- Chapter 1 gives an illustration of the project, which includes the objectives and the tasks. This chapter also analyze the relevant potential risks and tries to give the contingency plans for these risks.
- Chapter 2 introduces the background and motivation of this project. This project is relevant with a number of state-of-the-art wearable and mobile technologies. These technologies are to be explained before moving to the design and implementation details of this project.
- Chapter 3 briefly explains the requirements of this project and gives analysis for each one of them in detail. Student should identify and elicit design requirements. In addition to the background in Chapter 2, student is able to get additional information from the real life
- Chapter 4 presents the design approach, which contains the system architecture, MVC build, and platform choice.
- Chapter 5 introduces how to implement the system including user interface and ER-diagram. In this chapter, swift is used as the programming language.
- Chapter 6 gives the results of testing. In the chapter, I find the problems of each function, analyze the reason and give solutions.
- Chapter 7 makes a conclusion and predicts the future work on this project.

2. Background

This chapter explains the relative tools in wearable and mobile software development. Finally, it introduces several similar applications in APP Store and analyze their pro and con.

2.1. Wearable and Mobile

When I start to implement this wearable software, I have to face to a new programming language, Swift. For the beginner, there are two tutorials very valuable, the first one is the Apple's official developer website [4], the second one is Developing iOS 11 Apps with Swift [5].

In Apple Developer website, I am able to check the relevant document about API reference, articles and sample. Besides that, I can also download the latest version IDE, Xcode 9, which is a developer tools for macOS, iOS, watchOS and tvOS. In this project, I used Xcode 9 for developing a watchOS and iOS apps. The features of Xcode IDE are shown as following [4].

• Source Editor	• Asset Catalog
• Assistant Editor	• Open Quickly
• Version Editor	• OpenGL Frame Capture
• Interface Builder Built In	• Complete Documentation
• Simulator	• Live Issues
• Integrated Build Systme	• Fix-it
• Compilers	• Quick Help
• Graphical Debugger	• XCTest Framework
• Continuous Integration	• Static Analysis

Table 2-1 Features of Xcode IDE

For the course of Developing iOS 11 Apps with Swift, I systematically studied Swift from 4 aspects [5].

- Tools and APIs required to build applications for the iPhone platforms using iOS SDK.
- User interface design for mobile devices
- Object-oriented design using MVC paradigm and Swift programming language.
- Object-oriented database API.

2.2. Similar Applications in App Store

1. Remote controller

Presentation remote control app: The client could change slides from iWatch.

However, this kind of applications also has some disadvantages:

- It cannot support speaker notes.
- It supports weak progress controlling.
- The client can only change next slide by clicked the button.
- The timer only worked as stopwatch function.

The best example is Microsoft PowerPoint on watchOS. The following figure is the screen shot of Microsoft PowerPoint in Apple Watch App Store [6].

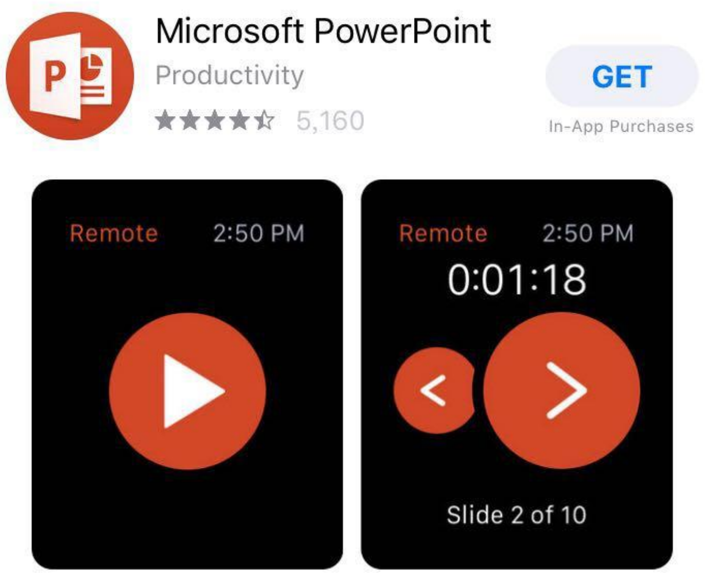


Figure 2-1 Microsoft PowerPoint of iWatch

2. Presentation Timer++ [7]

   This app shows three different colors green, yellow and red for reminding speaking time of presentation. The application shows 3 notifications when timer turns yellow, red and once it's done. When the time is up, the alarm is ringing.

   Nevertheless, this app also has some drawbacks:

   - It cannot support speaker notes.
   - It has a poor UI design.
   - It cannot support a specific topic.

   The following figure is the screen shot of Presentation Timer++ in Apple Watch App Store.

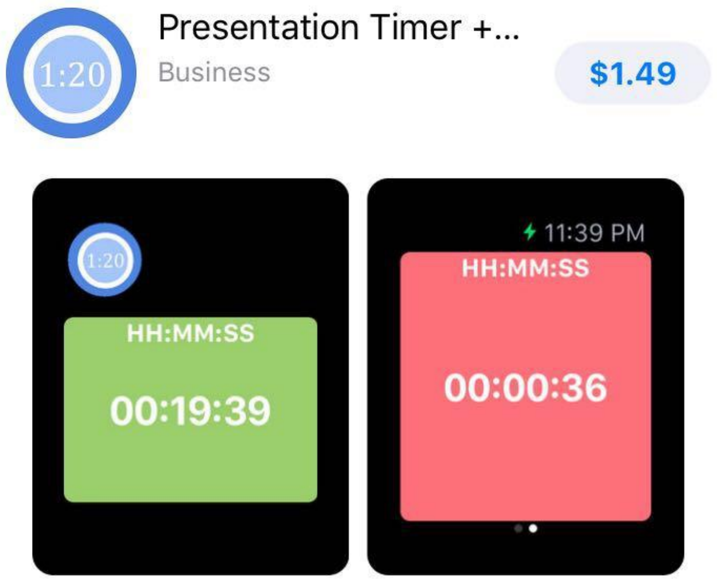


Figure 2-2 Presentation Timer++

3. Requirements Analysis

This project focuses on facilitation the mobile and wearable device software. Most of the functional requirements are about recording the topic details and displaying on iWatch and iPhone. In this chapter, 20 functional requirements are broken into 3 blocks (e.g. A, B, C) and individual requirements are named alphanumerically (e.g. A1, A2 …)

iOS

The front page of the app on iPhone side shows a topic list.

A1. The user may add a new topic by click a button in topic list page.

A2. The user may browse titles in a list of topics. The list shows a basic information of topics, titles. Each title belongs to one of the pre-defined topics.

A3. The user can look for details by clicking the title in the list.

The topic detail page shows detailed information about one topic

A4. The user may select a title in the topic list to go to the topic detail page. The topic detail page shows information for one topic, which includes the presentation's name, the estimated time, word list and time list. In addition, the user may delete or go back to the topic list page in detail page.

A5. The user may insert new word by click a button in the topic detail page. The input field should contain word and corresponding time.

A6. The user may modify or delete word and time in the list. No matter what operation, the list is sorted by time.

A7. The user may cast the whole topic to iWatch.

To build any topics, a user must add topic to his/her topic list. The user can input general information in the "title" page. This page can accept input of title and estimated time.

B1. The user adds a new topic by clicking a button in the topic list page. The text field of title is assumed to be empty. The estimated time is assumed be 1 minute. In addition, the user may go back to the topic list page or proceed to the next step.

B2. The user may input presentation title in the text field of title.

B3. The user may select an estimated time of presentation in a time picker.

To build an integral topic, a user must add new words to his/her topic. The user can input intricate information in the "words" page. The page can accept input of words and times list.

B4. The user adds word and time list by clicking a button in the "title" page. The "words" page shows general information which was submitted in "title" page. In addition, the user may go back to the "title" page or save all information.

B5. The user may insert new word by clicking a button in "words" page. The input field should contain word and corresponding time.

B6. The user may modify or delete word and time in the list.

To get the summary of inputs, the user can check general and detailed information in the summary page.

B7. The user saves all information by clicking a button in the "words" page. The summary page shows them including presentation title, estimated time and word list.

B8. The user can go back to the topic list page by clicking a button in the summary page.

watchOS

The first page of the app on iWatch side shows a general topic information.

C1. The user may browse a general topic information in the first page by clicking a button of iPhone side (A7). Without any cast from iPhone side, the presentation title and estimated time are shown as “nil”.

C2. The user may the presentation by clicking a button in the first page.

The second page shows timer and key words of presentation.

C3. The timer should automatically run after clicking a button in the first page.

C4. The key words should automatically change in the corresponding time. The user may change to next word by clicking a button in the second page.

C5. If all key words have been already displayed, the app will display a label to tell user the presentation is finished. The user may go back to the first page by clicking a button.

4. Design Approach and Methodology

This chapter presents the design approaches, focusing on the system architecture, which is the conceptual model that defines the structure, behavior, and move views of the system [8]. Beyond that, platform of system is also introduced in this section.

4.1 System Architecture

Three major system components are comprised in system architecture, including the externally visible properties and relationships between them. The following figures is the architecture of this project.

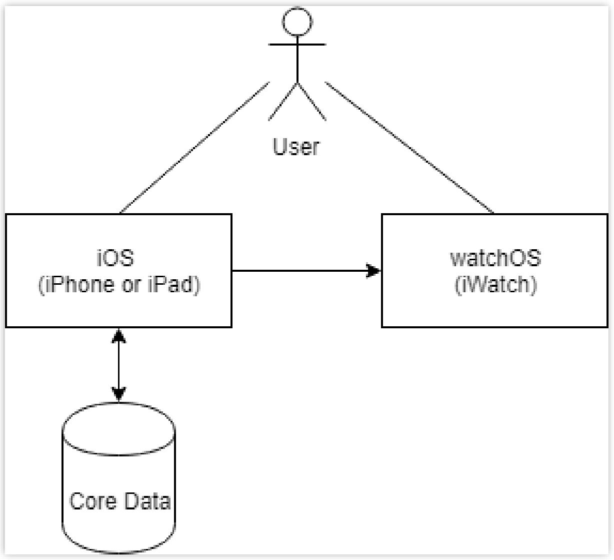


Figure 4-1 System Architecture

- iOS (iPhone part):

  According to A1 – A7 & B1 – B8 in chapter 3, user can insert, delete and modify data by using iPhone.
- Core Data:

  All the data was stored in Core Data. Core Data is an Object-oriented database built in Xcode.
- watchOS (iWatch):

  According to C1 -C5 in chapter 3, iWatch could receive data from iPhone.

To explain the system architecture more detailly, the following figure shows the software design of this project.

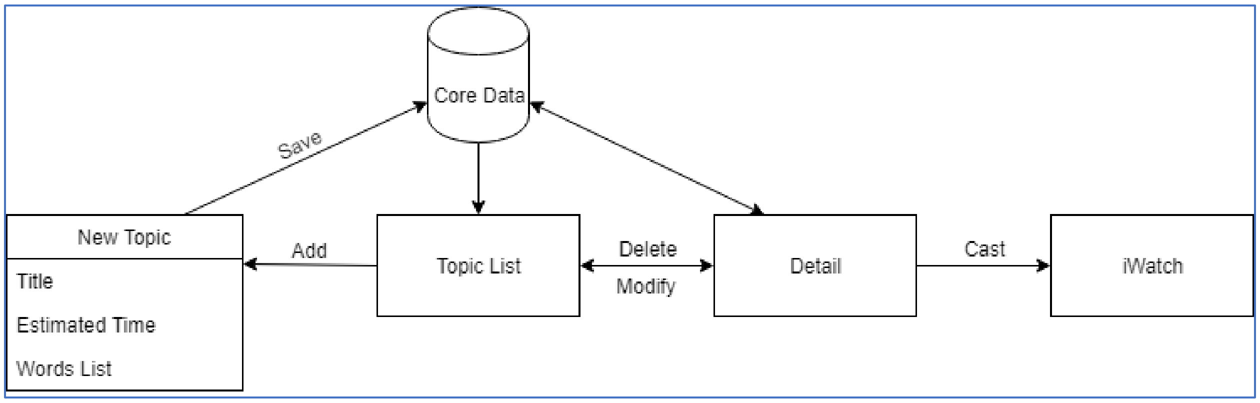


Figure 4-2 Software Design

In Figure 4-2 Software Design, the iOS part is broken into several components.

On the iPhone side, the home page is a topic list, user can add new topic in this page. The attributes of topic include title, estimated time and word list. All of these saved in core data. And user can see the detail of existed topic in the topic list. User can delete or modify the record. In this section, user can cast the detail to iWatch.

4.2 Model-View-Controller

The Model-View-Controller (MVC) design pattern in this project are considered to by 3 types of objects: model objects, view objects, and controllers objects. In this project, I chose the Cocoa version of MVC as a compound pattern [9]. The following figure shows the structure of Cocoa MVC.

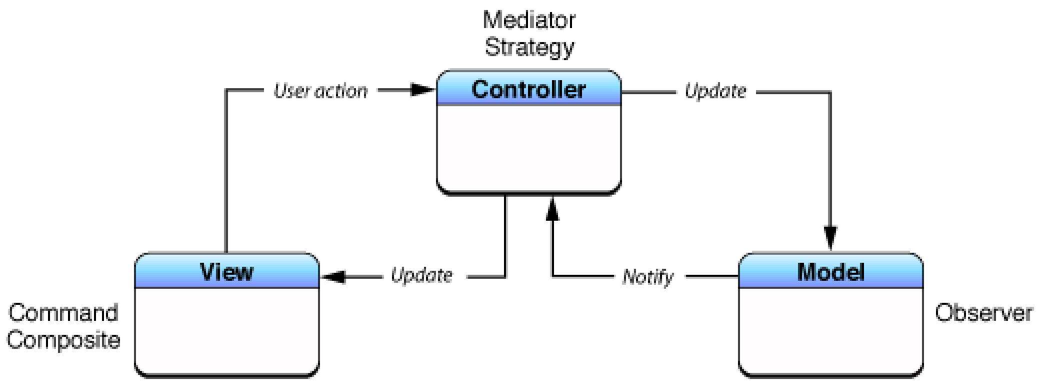


Figure 4-3 Cocoa version of MVC as a compound design pattern

For the model objects, it encapsulates and manipulates the data associated with the problem domain. In this project, I created several custom subclasses, for example, the Topic class is used to save the title name and estimated time; the Detail class is used to save a pair of words and corresponding time.

For the view objects, it represents the outlooking of an operating system and the applications which system supports. Based on Xcode inherent functions, it is implemented in the UI design.

For the controller objects, it mediates the flow of data between model and view objects in both directions. It requires the most code to write. In this project, each view object has a corresponding controller object, for example, I created DetailViewController class to connect the data and display them on the detail page.

4.3 Platform Choice

After discussing this project with my supervisor Dr. Qi, I bought a Mac Mini for system implementation, because the Xcode and Swift can only run on a masOS computer. And the software run on the simulator of iPhone 7 and iWatch series 1 with the latest operating system.

Figure 4-4 Mac mini [10]

CPU	2.6GHz dual-core Intel Core i5
Memory	8 GB
Hard drive	1 TB
Graphics Card	Intel Iris Graphics
Operating System	macOS 10.13 High Sierra, iOS 11.3, watchOS 4.3
Programming Language	Swift 4
IDE	Xcode 9.3

Table 4-1 Configuration Information

5. Implementation

This system uses Xcode for implementation, in this section, user interface design, ER-diagram and other implementation details are briefly introduced.

5.1 User Interface Design

User interface design is developed by Xcode. It contains 2 parts: iPhone and iWatch.

5.1.1. UI for iPhone

In this part, view controllers are shown as following figure, and then this report explains each of them with related functional requirements.

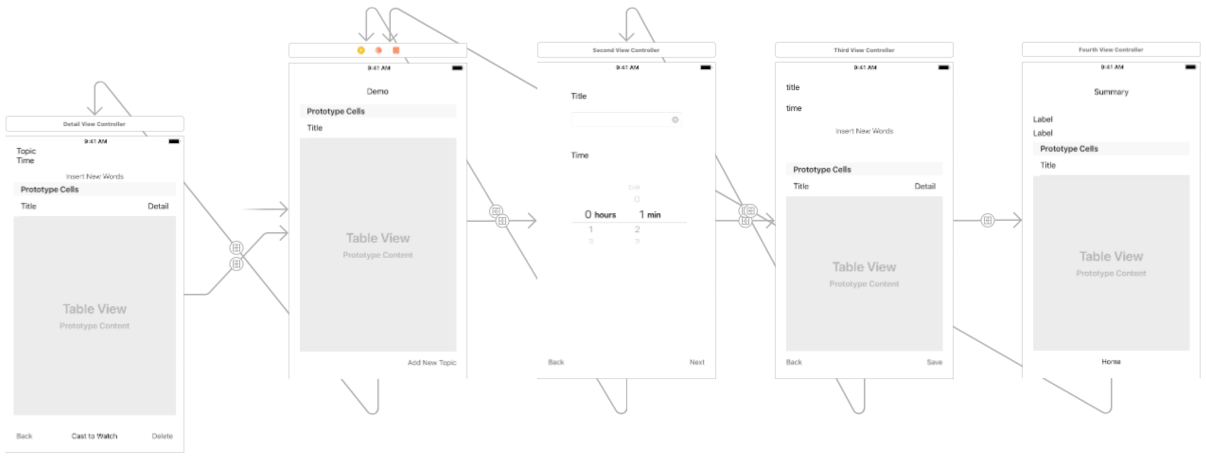


Figure 5-1 UI for iPhone

The following figure shows the topic list page design and result on simulator of iPhone 7, *ViewController* is the corresponding class of this part. The relevant requirements are A1, A2 and A3.

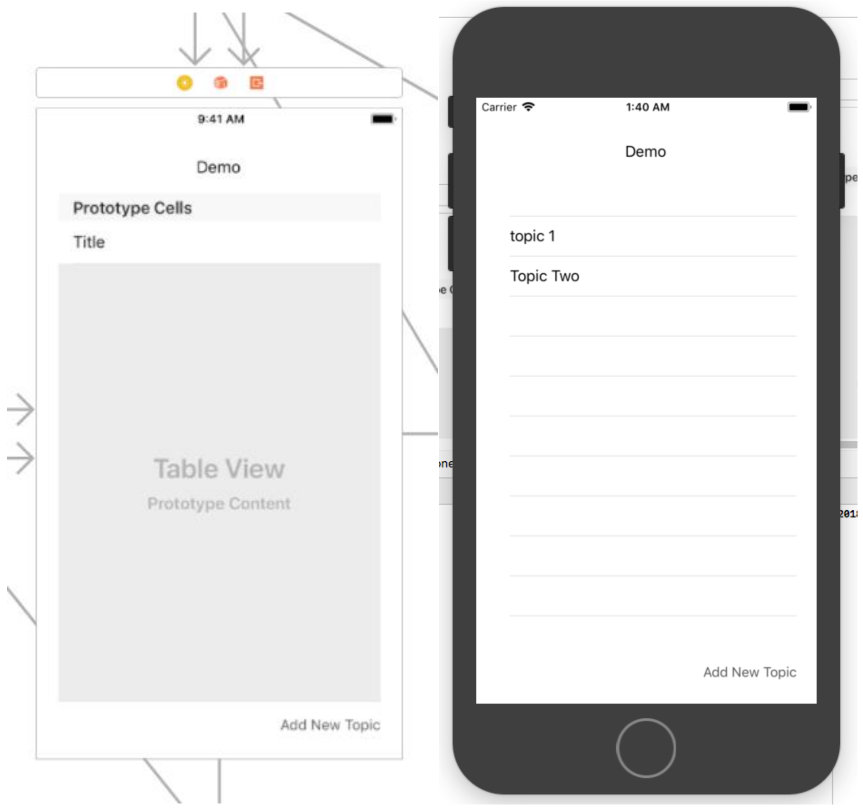


Figure 5-2 View Controller

The following figure shows the topic detail page design and result on simulator of iPhone 7, *DetailViewController* is the corresponding class of this part. The relevant requirements are A4, A5, A6 and A7.

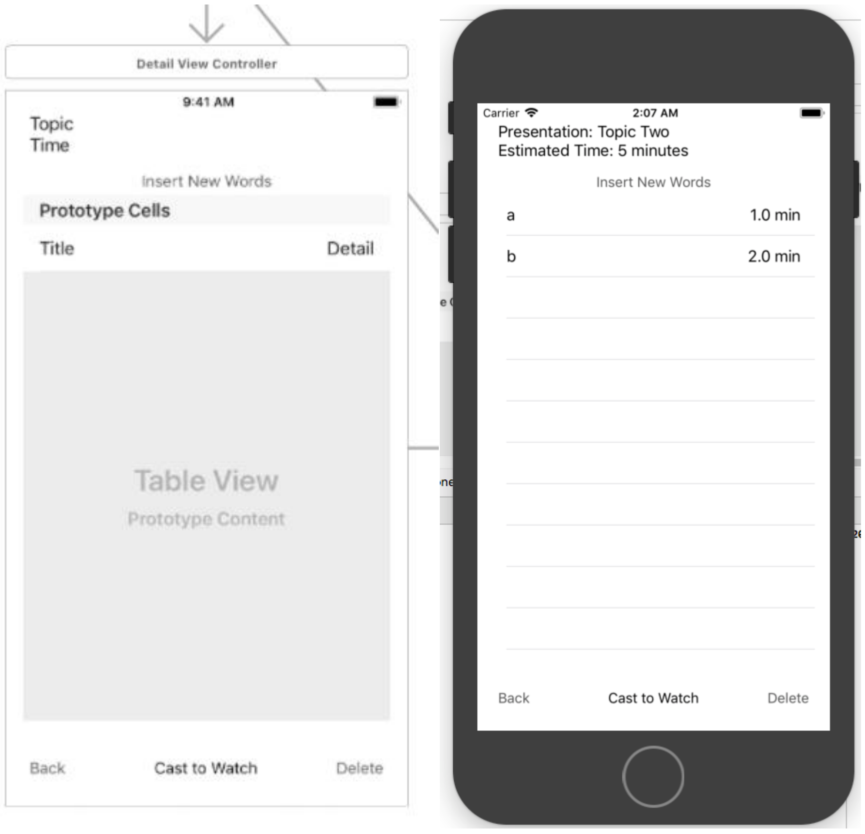


Figure 5-3 Detail View Controller

The following figure shows the “title” page design and result on simulator of iPhone 7, *SecondViewController* is the corresponding class of this part. The relevant requirements are B1, B2 and B3.

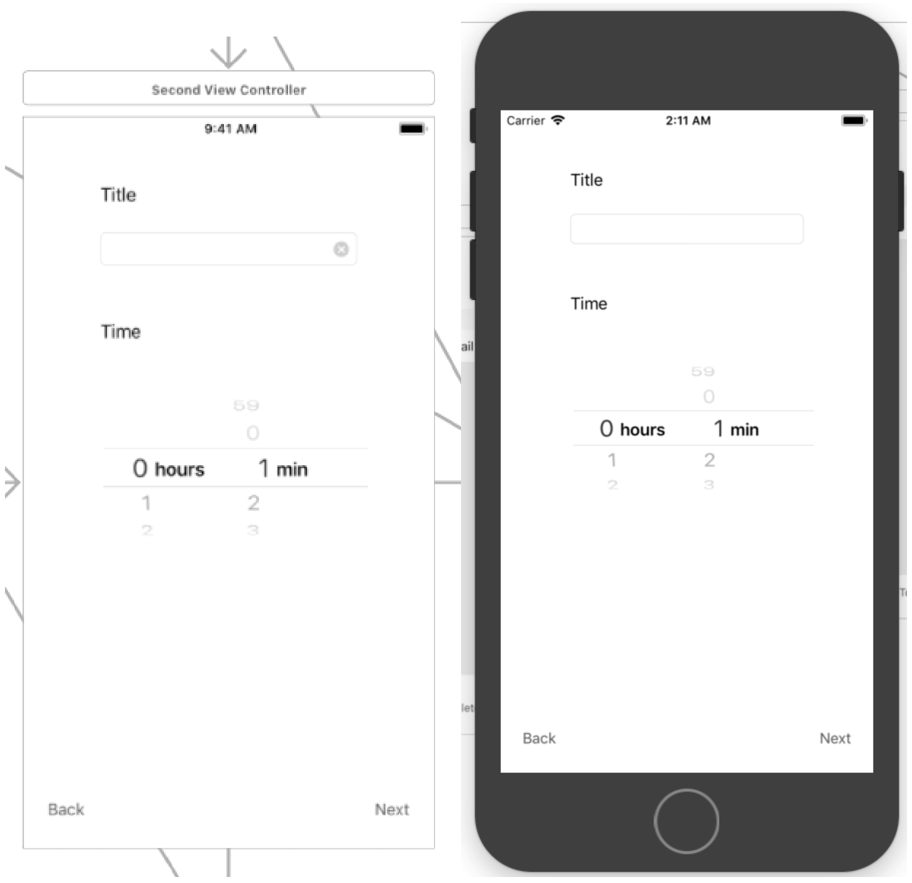


Figure 5-4 Second View Controller

The following figure shows the "words" page design and result on simulator of iPhone 7, *ThirdViewController* is the corresponding class of this part. The relevant requirements are B4, B5 and B6.

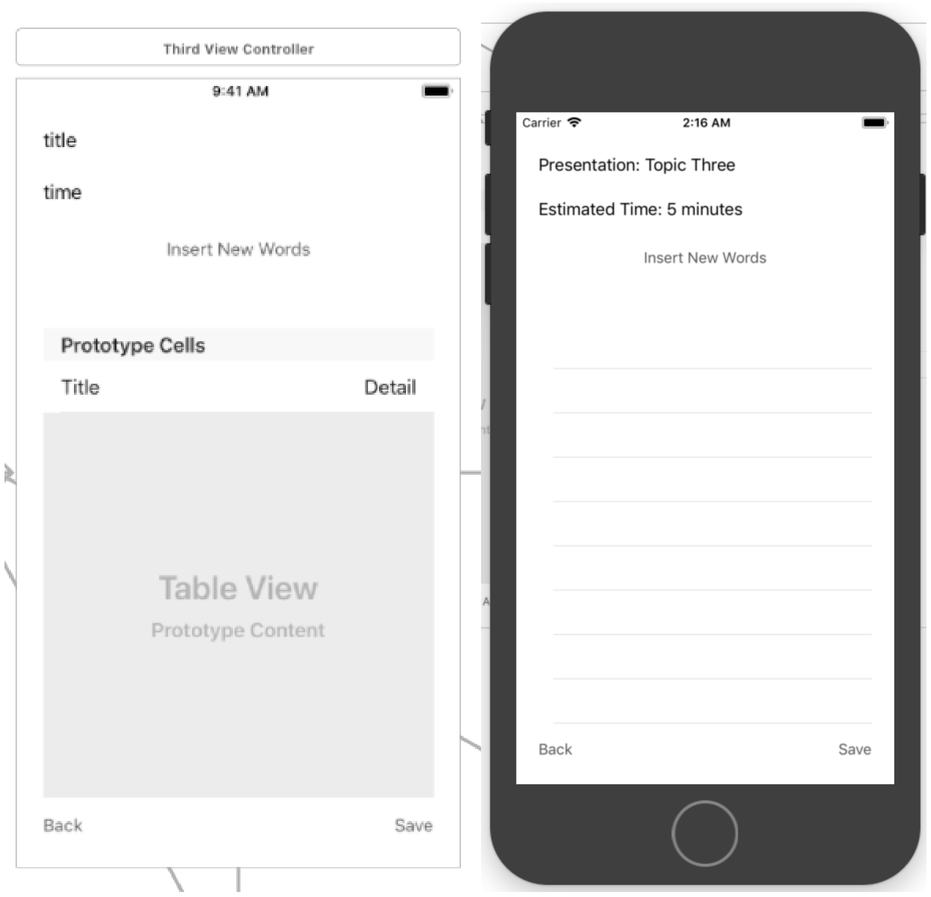


Figure 5-5 Third View Controller

The following figure shows the summary page design and result on simulator of iPhone 7, *FourthViewController* is the corresponding class of this part. The relevant requirements are B7 and B8.

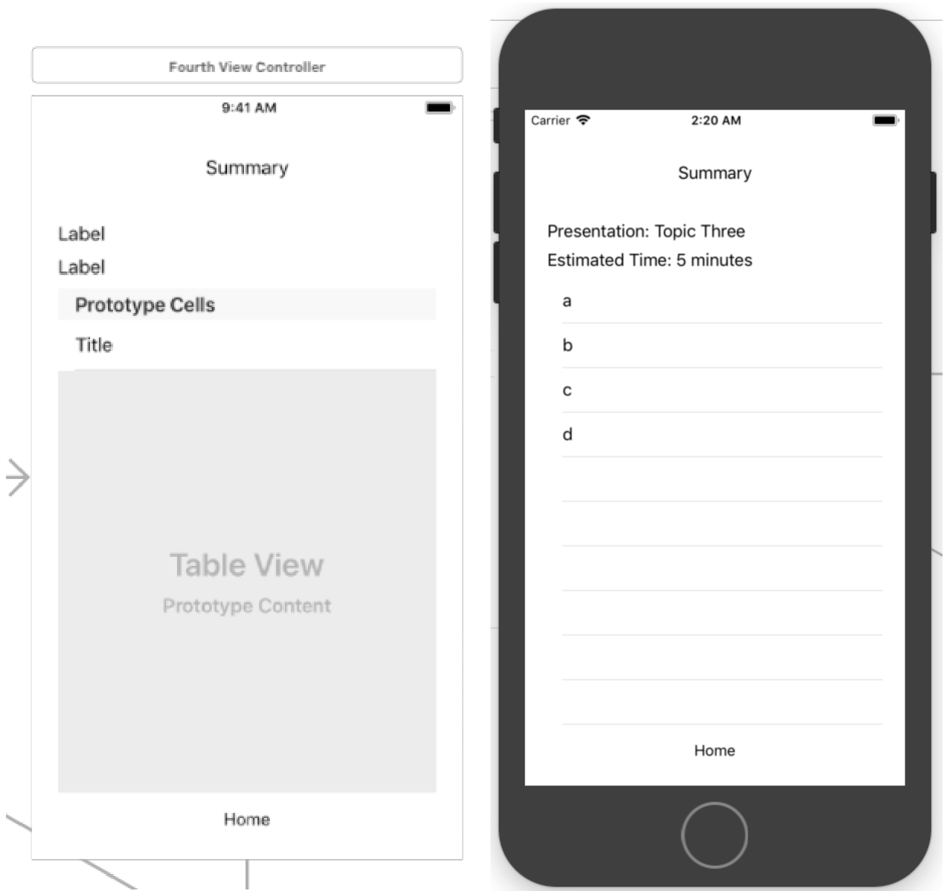


Figure 5-6 Fourth View Controller

5.1.2. UI for iWatch

In this part, interface controllers are shown as following figure, and then this report explains each of them with related functional requirements.

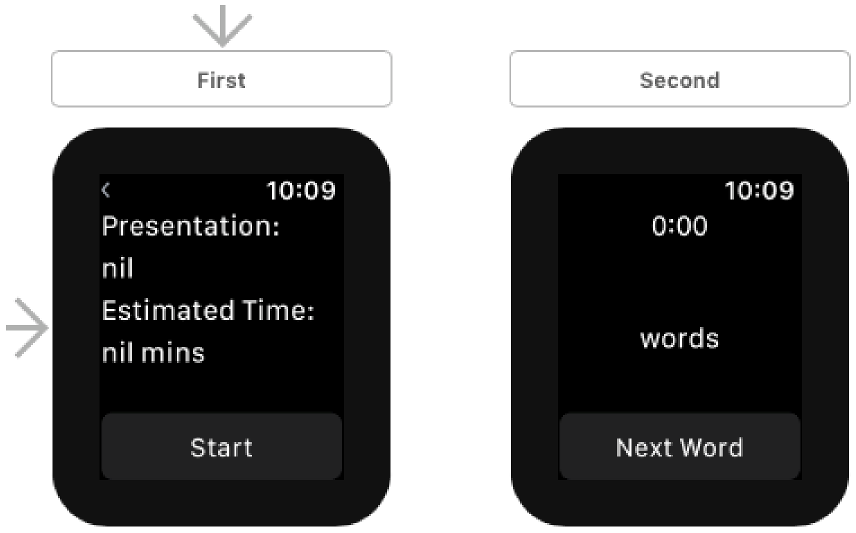


Figure 5-7 UI for iWatch

The following figure shows the first page design and result on simulator of Apple Watch – 42mm, *InterfaceController* is the corresponding class of this part. The relevant requirements are C1 and C2.

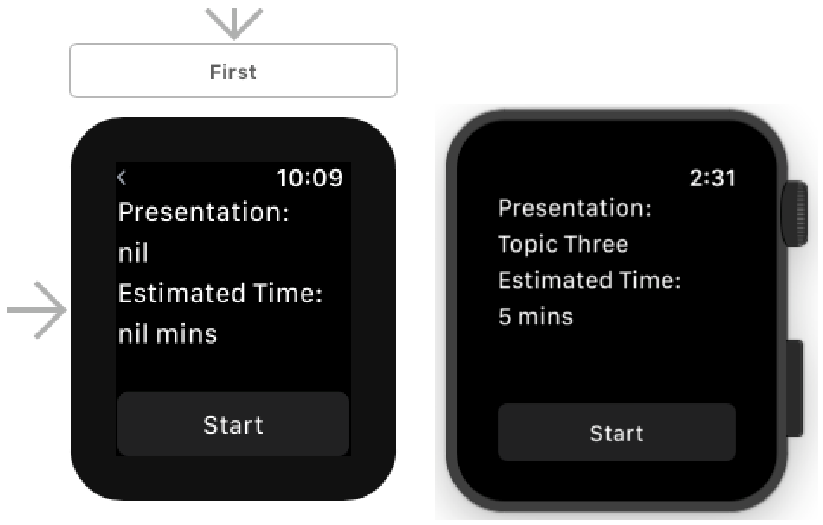


Figure 5-8 Interface Controller

The following figure shows the first page design and result on simulator of Apple Watch – 42mm, Second*InterfaceController* is the corresponding class of this part. The relevant requirements are C3, C4 and C5.

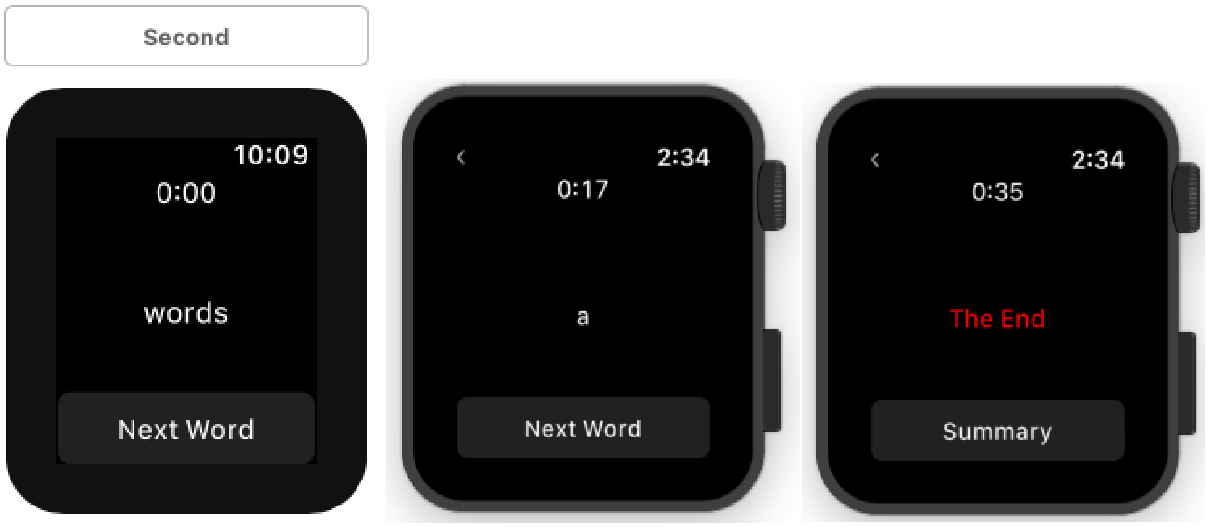


Figure 5-9 Second Interface Controller

5.2 Data Modelling

Core Data manages the model layer objects in this system [11]. Although Core Data is an Object-Oriented database, like a normal relational database, it can be described interrelated things of this project by entity-relationship model. In data modelling of this project, two entities are designed, including Topic and Detail. Figure 5-10 shows the entity relationship diagram of the database of this project.

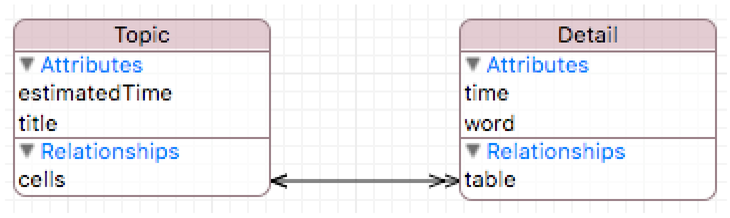


Figure 5-10 ER Diagram

For Topic entity, it contains 2 attributes, *estimated Time* is used for storing the estimated time of presentation; *title* is used for storing the title of presentation.

A presentation can have many words. The Detail entity contains the information of word in the words list, such as a pair of word and corresponding time.

5.3 App Delegate Implementation

In this project, three libraries are imported, "*UIKit*", "*WatchKit*", "*CoreData*" and "*WatchConnectivity*". And related subclass should be declared before programming.

The UIKit framework provides basic structure for iOS apps [12]. In this project, a lot of subclass of UIKit are utilized, such as UILabel, UITableView, UITextField, UIDatedPicker etc.

The WatchKit framework could manipulate the interface of a watchOS app [13]. It provides basic function for watchOS apps. Subclass such as WKInterfaceLabel, WKInterfaceButton and WKInterfaceTimer are used in this project.

The CoreData framework are briefly described in 0

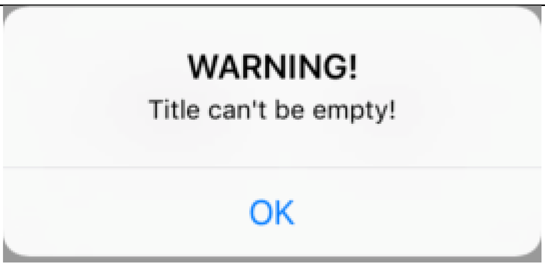
Data Modelling, some subclass of CoreData are used in this system, such as NSManagedObject, NSPersistentContainer, NSFetchRequest etc.

The WatchConnectivity framework could transfer data between iPhone and iWatch [14]. For instance, WCSession initiates communication between a watchOS app and its companion iOS app.

6. Testing and Debugging

In order to test the rationality of my system, a unit test was performed after one function requirement achieved; an integration test was performed after one-page implementation; a system test was performed after iOS or watchOS part finished. Therefore, huge number of tests have been performed during the implementation. Because the user operation is unpredictable, I have list several test cases about user operation as following.

Test Case Number:	1		
Test Case Name:	Empty Title		
Description:	User keep text field empty in "title" page.		
Step	Action	Expected System Response	Pass/Fail
1	Keep text field of title empty		
2	Click Next button	The system displays an alert.	Pass


WARNING!

Title can't be empty!

Figure 6-1 Empty Title

Table 6-1 Empty Title

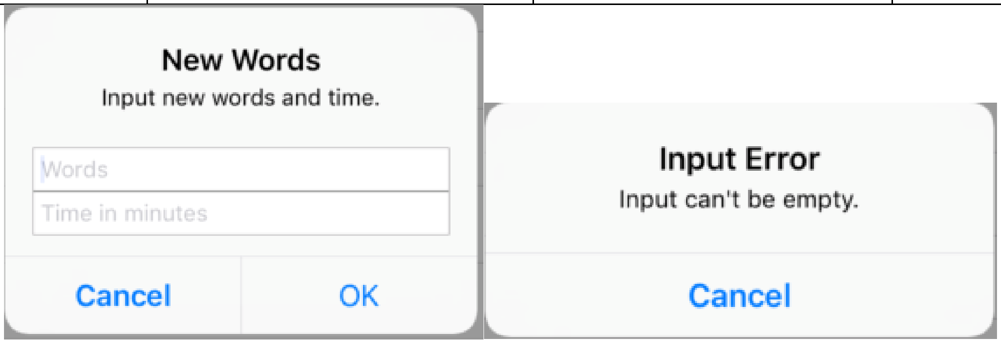
Test Case Number:	2		
Test Case Name:	Empty Input		
Description:	User keep text field empty when insert new words.		
Step	Action	Expected System Response	Pass/Fail
1	Keep text field of input empty		
2	Click OK button	The system displays an alert.	Pass
 New Words Input new words and time. Words Time in minutes Cancel OK Input Error Input can't be empty. Cancel *Figure 6-2 Empty Input*			

Table 6-2 Empty Input

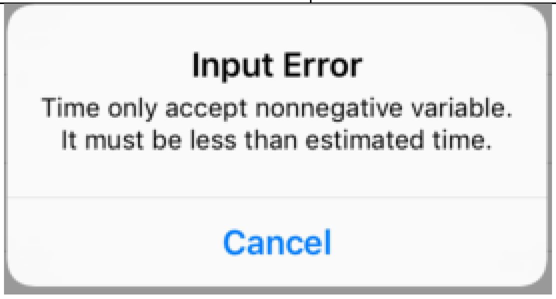
Test Case Number:	3		
Test Case Name:	Wrong Time		
Description:	User input wrong time when insert new words, for example letter, negative number or greater than estimated time.		
Step	Action	Expected System Response	Pass/Fail
1	Input wrong time		
2	Click OK button	The system displays an alert.	Pass
 Input Error Time only accept nonnegative variable. It must be less than estimated time. Cancel *Figure 6-3 Wrong Time*			

Table 6-3 Wrong Time

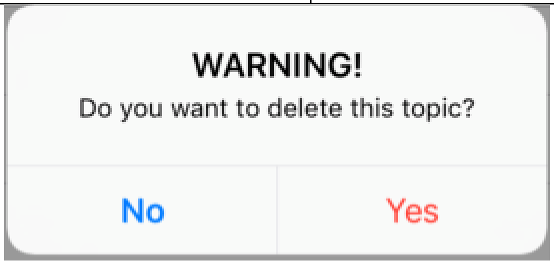
Test Case Number:	4		
Test Case Name:	Topic Deletion		
Description:	User delete the presentation in topic detail page.		
Step	Action	Expected System Response	Pass/Fail
1	Click Delete button	The system displays an alert for asking.	Pass
 WARNING! Do you want to delete this topic? No Yes *Figure 6-4 Topic Deletion*			

Table 6-4 Topic Deletion

7. Conclusion and Further Work

In conclusion, the project involves deep investigation and implementation in wearable device and mobile software development. Thus, much relevant experience was gained in this project. Over 1200 lines of code are written for this project. The system performance in the simulator looks good, and it could use in the real life.

The essential functional requirements are successfully built, the further improvements of the system will be done in the future work. The future work needs to consider the following:

The UI design is not good enough, some work such as icon of this app need be designed unique and concise. The layout of each page should be clearer and better looking, and a nice timeline could be built for the words list.

iWatch could vibrate for notifications, when the time is barely up, for example last 2 minutes, and it can also give haptic feedback to the user when change the word. The function “func play(_ type: WKHapticType)” will be implement in watchOS app [15].

iWatch can record human’s heart rate. Under the user’s agreement, the data of heart rate during the presentation will be recorded and send to server. During the presentation, if iWatch detected high heart rate, that means speaker may feel nervous, iWatch will vibrate and try to shift attention depending on outside stimulate. In the future, an analysis of the data may be concerned, and an improvement of this application may be developed in next version such as predict the time of high probability the speaker feels nervous.

Reference

[1] U. o. Chicago, "Stereotype-induced math anxiety robs women's working memory," 24 May 2007. [Online]. Available: http://www-news.uchicago.edu/releases/07/070524.beilock.shtml.

[2] Apple, "Apple Watch Series 1," Apple, [Online]. Available: https://www.apple.com/au/watch/. [Accessed 24 5 2018].

[3] Samsung, "Gear S3 classic," Samsung, [Online]. Available: http://www.samsung.com/au/wearables/gear-s3-classic-r770/. [Accessed 24 5 2018].

[4] Apple, "Apple Developer," Apple, [Online]. Available: https://developer.apple.com/. [Accessed 25 5 2018].

[5] P. Hegarty, "Developing iOS Apps with Swift," Stanford, 2017.

[6] Microsoft Corparation, "Microsoft PowerPoint," [Online]. Available: https://itunes.apple.com/au/app/microsoft-powerpoint/id586449534?mt=8. [Accessed 25 5 2018].

[7] S. Bhutta, "Presentation Timer++," [Online]. Available: https://itunes.apple.com/au/app/presentation-timer/id1022717427?mt=8. [Accessed 25 5 2018].

[8] H. Jaakkola and B. Thalheim, "Architecture-Driven Modelling Methodologies," 2011.

[9] Apple, "Model-View-Controller," [Online]. Available: https://developer.apple.com/library/content/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html. [Accessed 25 5 2018].

[10] Apple, "Mac mini," [Online]. Available: https://www.apple.com/au/shop/buy-mac/mac-mini. [Accessed 25 5 2018].

[11] Apple, "Core Data Programming Guide," [Online]. Available: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/. [Accessed 25 5 2018].

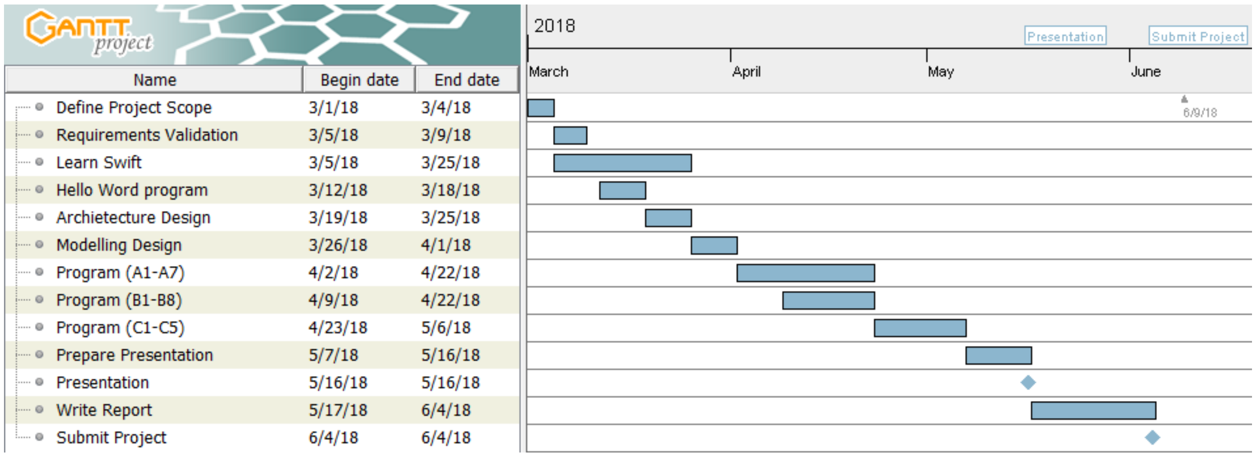
[12] Apple, "UIKit," [Online]. Available: https://developer.apple.com/documentation/uikit. [Accessed 25 5 2018].

[13] Apple, "WatchKit," [Online]. Available: https://developer.apple.com/documentation/watchkit. [Accessed 25 5 2018].

[14] Apple, "WatchConnectivity," [Online]. Available: https://developer.apple.com/documentation/watchconnectivity. [Accessed 25 5 2018].

[15] Apple, "play(_:)," [Online]. Available: https://developer.apple.com/documentation/watchkit/wkinterfacedevice/1628128-play. [Accessed 26 5 2018].

Appendix Project management



Name	Begin date	End date
Define Project Scope	3/1/18	3/4/18
Requirements Validation	3/5/18	3/9/18
Learn Swift	3/5/18	3/25/18
Hello Word program	3/12/18	3/18/18
Archietecture Design	3/19/18	3/25/18
Modelling Design	3/26/18	4/1/18
Program (A1-A7)	4/2/18	4/22/18
Program (B1-B8)	4/9/18	4/22/18
Program (C1-C5)	4/23/18	5/6/18
Prepare Presentation	5/7/18	5/16/18
Presentation	5/16/18	5/16/18
Write Report	5/17/18	6/4/18
Submit Project	6/4/18	6/4/18