**CMPT 275**

**Assignment 2B**

Quality Assurance Document

**Project Team 4**

Ximin Zhang

Florence Huang

Shayne He

Nov 09, 2015

Table of Content

Contents

[Revision History 3](#_Toc433328508)

[Software Tools for Testing And Test Case 3](#_Toc433328509)

[Internal Deadlines For Unit/System Testing 4](#_Toc433328510)

[User Acceptance Testing 4](#_Toc433328511)

[Integration Testing 5](#_Toc433328512)

[Software Tools For Measuring Size and Complexity 6](#_Toc433328513)

[Additional Measures For Quality Assurance 8](#_Toc433328514)

[References 8](#_Toc433328515)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Status** | **Publication/Revision**  **Date** | **By** |
| 1.0 | Created | Oct 19, 2015 | Florence |
| 2.0 | Add first four parts | Oct 21, 2015 | Shayne |
| 3.0 | Added Complexity part and completed the whole document | Oct 22, 2015 | Florence |
| 4.0 | Revised and add references | Oct 23, 2015 | Ximin Zhang |
| 5.0 | Revised | Nov 08, 2015 | Shayne |

# Software Tools for Testing And Test Case

In order to perform effective testing on our system, the IOS Xcode tools’ simulator will be introduced for general UI and functional testing.

We will use Xcode simulator to analyze the UI links to see if each button on the UI is connected properly to its desired function and display correctly on the simulation window. In terms of health kit data, the health kit Application Programming Interface will be used to retrieve data from the iWatch. We will write a small program on Xcode using swifts to see if our system data matches with what it displays on the watch interface.

Unit testing will be done in Xcode as well by adding print line in the end of the function to keep track of whether or not each feature’s support function can calculate the output correctly.

We will also use a program to accurately find all the desired deep sleep and light sleep heart rate. We will use breakpoints set in Xcode to create test cases for our application.

Finally, we will bring it to the real life, install our system to an IOS device (a selection of iPhone6, 5 or 6s) and pair it with iWatch to see the performance of the alarm system.

# Internal Deadlines for Unit/System Testing

(1) All of Version 1’s testing will be done by November 08, 2015.

Unit Testing will be done by October 31, 2015.

System Testing will be done by November 01, 2015.

User Testing will be done by November 08, 2015.

(2) All of Version 2’s testing will be done by November 15, 2015.

Unit Testing will be done by November 10, 2015.

System Testing will be done by November 15, 2015.

User Testing will be done by November 15, 2015.

(3) All of Version 3’s testing will be done by December 1, 2015.

Unit Testing will be done by November 23, 2015.

System Testing will be done by December 1, 2015.

User Testing will be done by December 01, 2015.

# User Acceptance Testing

Version 3 users acceptance testing will be done in the week of November 20th. We will distribute our App copy to friends and classmates who are currently using or have access to iWatch, and ask them to experience the app’s functionality. We will then record the users’ feedback about each feature that they used three days after including whether or not they experienced better wake up in the morning, accuracy of the alarm and sleep report feature and the usefulness of calendar and weather notification etc. The distributed feedback survey should also include what other features people will want in our final version of the application, how existing features could be enhanced/streamlined, and how they would improve.

# Integration Testing

Each person in our group will have his/her access to the project through our shared Github, and every member can have their own branch to work on. Each a member is trying to work on a new branch he/she should test the branch first. When a member completed a section of the implemention, he/she will make a pull request to update the master repository and it will be uploaded to the master branch of Github if all members confirm the request. The resulting integrated program will be built and tested through the simulator in Xcode by all members on a weekly basis, to ensure the application is running properly or generating errors for us to debug in the next development week, before integrating any more sections.

# Software Tools for Measuring Size and Complexity

Our app is built using Xcode, so that we will be using Oclint [1] to measuring the Size and Complexity of our program.

The Size of our Project and the complexity of our Project are both about Medium. The number of line for our code is around 3000. We will use some external libraries. We will reference them. The number of files will be around eight, and the number of classes will be around fifteen.

# Additional Measures for Quality Assurance

Regular meeting will be held every Monday after class and Sunday. During the meeting, we will discuss about our recent progress and what we should accomplish in the coming weeks, also what can be improved in our next build.

Wechat group discussion have been ongoing throughout the course, where we discuss parts of the assignments and brainstorm ideas that we take into the meetings and further discuss whether or not the idea will be doable for the application.

# References

[1] OCLint developer community. “*OClint*.”[Online] Available: http://oclint.org/. Accessed on Oct.23, 2015.