FIRST BITE – QUALITY ASSURANCE PLAN

*The quality assurance plans our application **First Bite**. It describes potential issues and in-depth techniques to ensure the quality of the project.*

CMPT 276: HW2

GROUP 07

Leon Trieu

Han Yang

Jeff Wang

Winston Ye

Kelvin Lee

Table of Contents

REVISION CHART

Version	**Primary Author(s)**	**Description of Version**	**Date Completed**
Draft	Kelvin Jeff	Initial draft created for distribution and review comments	June 16th 2018
Preliminary	Kelvin Jeff	Second draft incorporating initial review comments, distributed for final review	June 18th 2018
Final	Everyone	First complete draft, which is placed under change control	June 20th 2018
Revision 1	Winston Han Leon	Revised draft, revised according to the change control process and maintained under change control	June 18th 2018
Revision 2	Winston Han Leon	Revised draft, revised according to the change control process and maintained under change control	June 21st 2018
Revision 3	Kelvin Leon	Revised draft, revised after completion of iteration 1. Changed deadlines for future implementations	Jul 3rd 2018
Revision 4	Jeff Leon	Revised draft, revised after completion of iteration 2. Changed deadlines for future implementations	Jul 17th 2018

1. UNIT TESTING TOOLS

1.1 Software Testing Tool – Apple Xcode Test with XCTest Framework

We will write automated Unit Tests and UI Tests that implements XCTest framework for our Xcode project.

All test cases and result will be uploaded to our product Github repository.

2. Internal Deadlines

1.2 Unit & System Testing

1.2.1 Version 1 : Food Diary & Healthy Eating Guide Implementation (Unit Testing)

Implementation Deadline: July 1st 11:59pm

- Testing Deadline: July 3 11:59pm

  - Food Diary UI testing ensures the it meets the basic specification
  - Guideline information page should display properly

1.2.2 Version 2 : Statistics & Profile Implementation (Unit Testing)

Implementation Deadline: July 16 11:59pm

- Testing Deadline: July 18 3:59pm

  - Ensure all logs are stored in servers correctly and can be easily retrievable and sorted
  - Ensure articles are properly extracted from database and displays accurately
  - Ensure extracting correct information from the food diary function to the overview function to provide accurate statistics and graphs
  - Ensure the profile data stores properly in the database and shares correct data to other features

1.2.3 Version 3 : Sub-Function Implementation & Cleanup (Full System Testing)

Implementation Deadline: July 21 11:59pm

- Testing Deadline: July 24 3:59pm

  - Ensure the inputting hardwares of the device are properly linked with the profile function
  - Ensure the profile function generates correct BMI based on the stored data
  - Ensure sub features (backup, syncing e.g.) work properly
  - Ensure the UI and all the features meet the specification

3. ACCEPTANCE TESTING

1.2.4 Objective

- A group of 15+ people use the 3rd version of the product to test for UI usability and bugs.
- Our team will request our sponsor, Dr. Gerry Kasten, for assistance of getting a group of parent testers from his connection.
- A simple user review survey will be created for testers to send their reviews back to us.
- Provide feedback on the usability and layout for the app UI as well as the ease of use of the app functions.

1.2.5 Planning

- Date & Time

  July 25 (6:00am) – July 28 (11:59pm). Since the deadline for 3rd version is August 1, there will be 4 days to make adjustments based on users' reviews.
- Location

  Testing details will be sent out to testers through email. Parents can conduct the tests anywhere with their devices on.

4. INTEGRATION TESTING

Integration testing will occur simultaneously with the development process. It is different form the unit testing as each module is isolated and tested individually. As each unit completes all possible test cases, they will be slowly combined into one unit and tested as a whole. The main user interface will be created first to ensure that the app can run smoothly and navigate seamlessly between different tabs.

The logging feature will be created separately from the main UI and unit testing will occur on completion. This ensures that the feature can create, edit, and delete logs before it is integrated into the main UI. Upon integration, the app will be tested as a whole; data from the log must store successfully in the database and traverse to the correct location on demand. The food guide will also be worked on simultaneously and integrated after completing its unit testing.

Additional functions will be developed individually and tested before combining it into our larger features. For example, after the food guide has been integrated into our main UI, the search function can be combined and tested while the food guide function is added separately. After ensuring the search function is performing correctly, the entirety of the food guide function will be reintegrated into the main user interface. This way the new functions will be tested multiple times and will combine seamlessly every iteration.

By creating placeholders on the applications home page interfaces, we can slowly integrate each unit into the main program incrementally. By doing so, it will give us enough time to add and test each integration to see how the entire system reacts to the latest unit integration. We will also keep the previous copies. This would ensure that we always have a older version to fall back on if any unexpected error happens right after the most recent unit integration.

5. SCALABILITY

Project size is closely related to the complexity of a project. We will utilize commonly used software sizing methodologies include counting the lines of code of the source code, Function Point Analysis, Use Case Points and so on.

Xcode comes with line numbers and file navigator features that will help us keep track of information such as number of files and lines of codes. We will pay close attention to ensure our code blocks which contain loops and iterations are simple and clean. This will ensure that the code will be executing efficiently and free of memory leaks.

In addition, we will be using GitHub to further count the lines of code and the complexity, as it will allow us to collaboratively upload our progress. We will graph these metrics found in these applications using Google Sheets, which will allow instantaneous graphing, collaboration, and updating. With this, we will be able to create graph metrics such as lines of codes against time to see our progress.

The number of classes can also be monitored by structuring the .swift files properly.

6. ADDITIONAL QUALITY ASSURANCE

In order to ensure that the quality of our product, the group will be implementing additional measures that the group has decided upon:

1. **Early-stage testing**

   We will implement static testing to detect and fix defects right after we add any new code to the software life cycle to yield immediate feedbacks on potential quality issues.

2. **Refactoring**

   We will be continuously improving our code by refactoring and ensuring that our code is simplistic and easy to understand

3. **Live Testing During Meetings**

   In addition to our meetings, we will also test our application together as a group to ensure that the quality of the product will be consistent with our group goals.

4. **Commenting**

   We will be comment upon each commit and update by including information such as: timestamps, editor of the code, and what was changed/updated.