EasyGlucose

CMPT 276 – Group 01 – Glucinators
Assignment 5 – Quality Assurance Plan

Website: https://sites.google.com/view/cmpt276-summer2018/

Anmol Bajaj

Faisal Atif

Zhixin Huang

Tony Liu

Henry Yip

August 1st, 2018

1. Table of Contents

2. Revision History

Revision	**Status**	**Publication/Revision Date**	**By**
0.0	Rough draft created with requirement for each category	July 11, 2018	Henry Yip
1.0	Design document created	July 11, 2018	Everyone
1.1	Added rough writing with each part	July 12, 2018	Henry Yip
1.2	Assigned sections of plan for each group members	July 13, 2018	Everyone
2.0	Added “Software Tools”	July 13, 2018	Zhixin Huang
2.1	Added “Internal Deadline”	July 14, 2018	Tony Liu
3.0	Added “User Acceptance Testing” & “Integration Testing”	July 14, 2018	Anmol Bajaj
3.1	Added “Measure Size and Complexity”	July 15, 2018	Henry Yip
3.2	Added “Ensure Quality of Project”	July 16, 2018	Everyone
4.0	Modified the report	July 17, 2018	Everyone
4.1	Formatted the file	July 17, 2018	Zhixin Huang
5.0	Changed to fit version 3 QA.	July 30, 2018	Henry Yip

3. Software Tools

As of the updated Iteration 2, we have changed the software for our local database. We planned to use SQLite wrapper for our database. However, we have used a more stable software called Realm.io. Everything else remains mostly the same. Using the latest software bundles may introduce bugs to our application as they have not been well-used by the developer community. In addition, priority is given to open-source software when deciding which software to use.

Listed below are the software tools we plan to use in the development of our EasyGlucose. This is a comprehensive list that includes software to setup the best possible programming environment to the best available tools to perform Unit Testing and Component Testing on our code.

1.CocoaPods: https://github.com/CocoaPods/CocoaPods

Cocoa is Apple's Application Programming Interface (API) for their operating system iOS and macOS. iOS applications rely on many Cocoa libraries to be fully functional. We plan to use the most popular "Cocoa Dependency Manager" called CocoaPods. It's a powerful tool that "resolves dependencies between libraries, fetches source code for all dependencies, and creates and maintains an Xcode workspace." [1]. CocoaPods will help us in integrating the libraries and testing tools listed below.

2. Quick or Nimble

a) Quick: https://github.com/Quick/Quick
b) Nimble: https://github.com/Quick/Nimble

Both tools will be an important aspect of our testing development framework. We simply need to import Quick and Nimble into our Xcode project and start programming. The test function in Quick and Nimble will verify our code's behavior and compare the outputs of the code from our expected outputs. It will return whether our code behaves as we expect.

3. Balsamiq mockup and Sketch

We plan to design the User Interface (UI) concurrently with the development of the application. Balsamiq will be used to try and experiment with sample "rough" sketches of the app. Whereas, Sketch is a professional design tool that will be used to create the final design files. The developer's will be able to use the design files to start programming the app.

4. DateTools: https://github.com/MatthewYork/DateTools

It is an extremely cumbersome to process to write date and time zone libraries in iOS. Most professional iOS developers rely on external Date and Time Zone management tools to help them integrate those functions into their applications. We have decided to use DateTools since it is extremely important for us to associate the user glucose data with date and time. DateTools is a free, open-source, and the most widely used of such libraries.

5. Realm.io Mobile Database: https://realm.io

Realm.io Mobile Database a popular database tool for data-persistence used in the iOS. It is easier to use than the in-build alternative called CoreData. It allows us to initialize new data as objects and manipulate that data within a class.

Software Testing

After extensively researching the many testing methodologies, we have decided to use Black Box Testing while developing for EasyGlucose. Black Box Testing involves hiding the implementation details from the Tester. The Tester has no access to the code. They are expected to submit inputs into our app and compare the outputs with expected outputs. We believe that Black Box testing will create a less biased testing environment and lead to a more stable application.

In addition to the final component testing by the Tester in our group, every developer will be required to follow a protocol of having their code Unit Tested by another developer before it is pushed to the main project file. Developers will be required to manually generate test cases and test them individually. We believe that this protocol, in addition to the software tools, will lead to the software of the highest quality.

4. Internal Deadlines

The following list of deliverables are to be completed and tested by developers or users on their corresponding deadlines. Adequate time is allotted after each system deadline for bug fixing and system integration.

Unit:	**Description:**	**Deadline:**
Keyboard input	User should be able to input a decimal number via keyboard.	June 25
Database	Inputs should be stored and sorted by time; analyzed for mean, median, and outliers.	June 28
Graphical interface	The application should exhibit a line graph of blood glucose levels.	June 29
Personal information	The application should ask for age, gender, and type of diabetes and allow users to input them when they first start up the app.	June 29
Version 1 system test	**Prepare application for presentation.**	**July 1 [Passed]**
UI design	The app should run smoothly with the new UI and have extra pages for upcoming features.	July 4
Photo input	Users should be able to take pictures and save them in the app.	July 6
Photo log	Users should be able to log notes and tags onto a diet diary with the pictures.	July 9
Voice input	Users should be able to input decimal numbers via voice command.	July 15
Version 2 system test		**July 16 [Passed]**
User profile	A separate user profile page with their information.	July 21
Settings page	Personal information, language, and preferences can be changed within the settings.	July 24
Secondary language	Include language options for Mandarin	July 24
iPad compatibility	The application should scale to suit iPads UI elements in disorder.	July 28
Version 3 system test	**Prepare application for presentation**	**July 29 [Passed]**

5. User acceptance testing

Instead of having our friends or colleagues' user test our application, we believe that we must user test our application with our primary target audience: the elderly. It would be valuable to gain user feedback from the population segment that would potentially be using our application the most. It will help us avoid any technical pitfalls that make the application confusing or hard to navigate. This will immensely help us in fulfilling our goal to design the application to be so intuitive to use that our user's feel like they have been using it for years.

The selected location for the User Testing of Version 3 is Surrey Seniors Center. It is located at 13775 70 Ave, Surrey, BC V3W 0E1. This is the most accessible location for our group members to meet. More importantly, this is a popular place for the seniors to come and enjoy many different activities. We will be conducting the User Testing at 2:00 PM on July 22, 2018. This date was chosen because it is over the weekend and a week before our Version 3 is due. The weekend date will allow us to approach more seniors to ask for their feedback. A week from Version 3 due date also gives us adequate amount of time to take our user's feedback in consideration and implement the changes to our application.

We plan to email and/or call Newton Seniors Center and notify them of our plan to visit them. We will follow all protocols or policies advised by them. It is important to be compliant with the rules at their establishment. Furthermore, we plan to purchase cheap individual flowers for $50 and offer a flower to each senior that to agrees to give us feedback.

Here is the process that we will follow during our User Testing phase:

1. Ask the user if they are interested in helping us create a Blood Glucose tracking app. Only proceed if they agree.
2. Show them the app icon on an iPhone screen and ask to click it.
3. This will lead them to the Onboarding phase. Ask them to navigate to the main screen.
4. Ask them if they are confused about anything on the main screen.
5. Ask to tap the most prominent button on the main screen. Take note if they did not tap the "Log Glucose" button.
6. Use the app to record a sample glucose level (if they are diabetic patients)
7. Ask them if they are clearly able to identify the log button and try to add one of pictures, tags, events, or personal notes.
8. Introduce them to the timeline and glucose level graph. Ask them if they can understand the information presented. If they are not, ask what was particularly confusing.
9. Tell them about the ability to create reports and email it to dietician and/or physician. Request them to navigate the app and create the report.
10. Introduce them to the Profile page. Ask them if they can edit details such as type of diabetes, glucose target range, or change the language of app to Chinese (simplified).

6. Integration testing

Prior to integration testing, individual components must be tested, deliverable, and verified by the unit testing process. Each component is divided into three "levels"

High	Buttons	Graphs	UI	Display
Medium	Data sorting	Data manipulation	Data tables	
Low	Data input	Database	Picture importing	Adding tags/notes

First, high level features are implemented into the system and tested thoroughly. During this phase, the main priority is to establish enough screen space, working buttons, screen redirections, and dropdown boxes. Completing this level first allows the other components to be tested on a working visual interface, rather than a placeholder.

Prior to integration testing, individual components must be unit tested and working. Once they are available, these critical individual units in the lower level will be combined in increments. Interacting components such as blood glucose input and data storage will be tested together first. More relating components will be added as increments until lower level components are completed and grouped into modules. The data transfer between components is the primary aspect of this level that is tested and observed.

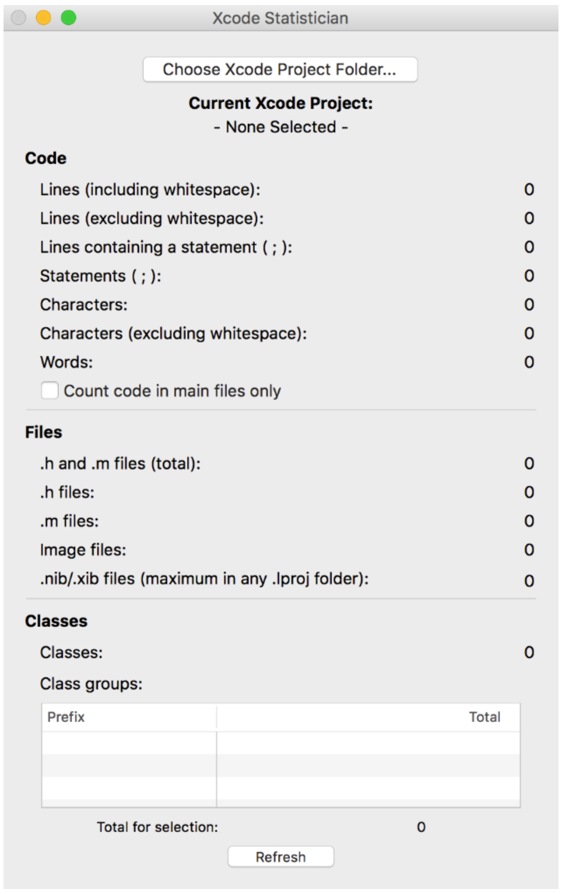
Once lower level components are grouped and tested, they will be integrated into the medium level components that each group interacts with. Correct output and consistent formatting is expected at this stage. Finally, the low and medium modules are implemented into the high-level components to complete the system.

The final system will be undergoing preliminary testing by developers to ensure completeness, and then the application will be handed over to users for user testing.

7. Complexity

1. Spreadsheet
2. Python script for counting lines/scitools
3. Xcode Statistician.

It is important to manage complexity in any software project. We have chosen to use a popular tool, Xcode Statistician, to manage the complexity of our project. Xcode Statistician is a simple tool that will provide us with a lot of raw data about our project. It'll provide us with the total number lines of code, number of total files, number of classes, number of comments etc. We also plan to visually represent out raw data in a summarized comprehensive complexity report using Microsoft Excel spreadsheets



Current Xcode Project:
- None Selected -

Code

Lines (including whitespace):	0
Lines (excluding whitespace):	0
Lines containing a statement (;):	0
Statements (;):	0
Characters:	0
Characters (excluding whitespace):	0
Words:	0

☐ Count code in main files only

Files

.h and .m files (total):	0
.h files:	0
.m files:	0
Image files:	0
.nib/.xib files (maximum in any .lproj folder):	0

Classes

Classes:	0

Class groups:

Prefix	Total

Total for selection: 0

8. Other methods of quality control

The group will take several more measures to streamline the process, work efficiently, and assure quality.

1. Regular peer code review to ensure orderly formatting, clarity, completeness, and modularity.
    a. This will be performed when developers are finished their assigned task.
    b. A last peer code review will be done after each version system test
        i. June 1, July 16, July 29
2. Upon request or observation of struggle, the project manager will assign an extra member to complete a task.
3. All tasks should be assigned based on individual strengths and availability.
4. Before beginning a major task, the group will set the task's essential requirements during a group meeting before the assigned individuals undergo the task.
5. Unexpected requirements changes will be resolved by notifying group members and scheduling a group meeting (online or offline) to validate the change executing the change.
6. Several backup files will be kept on cloud storage, SFU servers, and personal hard drives, especially for files that have been edited (to preserve older versions)
7. The visional control service Git will be constantly updated and pushed to at least once a day.
8. Constant communication via Discord, Google Docs, and group meetings to ensure group management and task preparations.
9. In case of merge conflicts while pushing code to Github, the developer must resolve them while another developer from the group is present.

9. Citation

[1]"CocoaPods/CocoaPods", GitHub, 2018. [Online]. Available: https://github.com/CocoaPods/CocoaPods. [Accessed: 23- Jun- 2018].