**Assignment 2 - Breathify**

Quality Assurance Plan

**CMPT 276 - E100**

Group 9

**Group Members:**

Jason Xu

Princess Macanlalay

Keith Chan

Hans Kim

Table of Contents

[Revision History 3](#_Toc475721792)

[Table 1 – Revision History 3](#_Toc475721793)

[Software Tools for Automatic Unit Testing 3](#_Toc475721794)

[Internal Deadlines 3](#_Toc475721795)

[Final User Testing 5](#_Toc475721796)

[Integration Testing 5](#_Toc475721797)

[Exercise Trainer 5](#_Toc475721798)

[Updating UI and Database 6](#_Toc475721799)

[Menus 6](#_Toc475721800)

[Software Tools for Size and Complexity 6](#_Toc475721801)

[Other Quality Assurance Items 6](#_Toc475721802)

# Revision History

## Table 1 – Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Status | Publication/ Revision Date | By |
| 1.0 | **Created**  Added  Software tools for Automatic Unit Testing,  Internal Deadlines,  Final User Tesing | **February 23, 2017** | **Jason Xu, Hans Kim, Princess Macanlalay, Keith Chan** |
| 1.1 | **Final**  Added  Software tools for software size and complexity,  Other QA items,  Final Formatting | **February 24, 2017** | **Jason Xu, Hans Kim, Princess Macanlalay, Keith Chan** |

# Software Tools for Automatic Unit Testing

We will be using the built-in functions of Xcode, in particular the XCTest framework, to perform software testing. XCTest allows users to write tests for Xcode projects that ensure certain conditions are met during test execution and records failures. Some of the functions XCTest provides for testers are collecting information about test runs, checking for errors, and testing user interface elements. All tests will be written by developers before coding individual components.

# Internal Deadlines

* Mar 2nd Deadline:
  + Timer based exercise component
  + Game based exercise component
  + 'Results Screen' for time based trainer:
    - Show the exercise, duration, and heart rate data results
  + Camera and flash heart rate monitor
* Mar 6th Deadline (End of Version 1):
  + Pause menu:
    - Mute/unmute
    - Restart
    - Pause
    - Quit (back)
  + 'About Breathify' page for the 'Resources' menu
  + 'Scoring Screen' for the game based app completion
  + Components that have failed in unit testing during previous deadline will be tested again
* Mar 16th Deadline:
  + Database of user profile and exercise records
  + API for accessing the database
  + Create local and online user profile functionality
  + Authentication for online user login
  + Exercise filter for the "Exercise Library"
  + Exercise history
  + Unfinished components from Version 1
* Mar 20th Deadline (End of Version 2):
  + 'Friend List' section:
    - View friend profiles
    - View friend exercise results and log history
    - Add/remove friends
  + Password changing/recovering for user
  + 'View Profile' section:
    - View/edit profile
    - View user exercise results and log history
    - Change account information/settings and app settings for user
  + Navigational UI elements should be functional—except the 'View Resources' menu
* Mar 30th Deadline:
  + 'Contact List' section:
    - Display contact information
    - Display e-mail forms
    - Dial contacts using system dialer
  + 'Resource Library' section:
    - Anxiety, stress, and breathing sourced articles
  + Rating and feedback section:
    - Feedback form for Breathify
    - Feedback and rating for timer based trainer
  + Tutorials for the breathing exercises
* Apr 3rd Deadline (End of Version 3):
  + Create personalized breathe patterns to add to exercise library
  + (optional) App optimization and extra features:
    - Creating profile using social media login method
    - Reminder functionality for exercises
    - Graphical improvements

# Final User Testing

User acceptance testing for the final version of Breathify will be done 4:30pm on March 30, 2016 at CSIL. We will be asking users to:

* Set up a local profile, then make it an online profile
* Go through the tutorials
* Browse the exercise library
* Make changes to and apply a filter to the list of exercises
* Do one timer-based exercise
* Play one game-based exercise
* Rate and comment on the exercise at the results screen
* View user’s exercise history
* Add a friend which will be a dummy account
* View a friend’s profile
* Delete the dummy friend account
* Access an article in the resource library
* Use call/email function to contact a placeholder contact
* Use developer feedback form
* Sign out of the user profile

At the end of each step, the user will be asked to comment on the quality of the experience. It should be noted how many errors the user makes as well as how many times the user is confused by what they are being asked to do.

After the initial test, the user will be asked to play three more timer-based exercises and three more game-based exercises. The user will then be asked to comment on the effectiveness of both. At the end of the session, the user will be asked to comment on their overall impression of the application.

# Integration Testing

Testers will perform unit testing on individual classes and external libraries. Once these have been completed, integration testing will be done. These are the integration tests that we expect to do:

## Exercise Trainer

Both the timer-based and game-based trainers are composed of many different classes. They both have the Heart Rate Sensor, Timer, Exercise Data and Exercise Results. The game-based version includes a Microphone class and a Scoring System class. Manual testing will need to be used here because of the data input from the Heart Rate Sensor and Microphone classes.

## Updating UI and Database

Comments, Ratings, Syncing, Viewing Exercise List and Viewing Profile

The functions listed above take input from the UI and update the database, or vice versa. Automatic testing will be performed to make sure that the database updates correctly from user input, or the UI updates correctly to changes in the database.

## Menus

When the three core features (exercises, user account, and resources) have been implemented, the User Interface will be used to “link” them. Testers will use the built-in UI testing features of Xcode for this.

# Software Tools for Size and Complexity

To measure the size of the project, an executable called 'CLOC' can be used to count the number of lines of code (from GitHub, created by Al Danial <https://github.com/AlDanial/cloc>). It can also be used to count the number of commented lines, blank lines, and files used. We'll also use 'Lizard', a Python script that counts the number of tokens and parameters of functions. It also contains a Cyclomatic Complexity Analyzer, which estimates the risk complexity of the project (from <https://github.com/terryyin/lizard>).

# Other Quality Assurance Items

The development team will hold a QA review twice a week during the Monday and Thursday meetings. This review session will go over challenges the developers faced since the previous review. An action plan will be put in place to be completed before the next QA review.