Specification

Your team will develop a desktop program which will pull and display data about an authenticated user from Fitbit.

This document outlines the specification for the programming portion of the project.

There are project-organization, documentation and other deliverables that make up the grade at each stage.

Please refer to the Stage specifications to see these additional requirements.

Specification Contents

This document is subject to change, ensure that you are always working from the latest version.

Version	Date	Comment
Current Version (v. 7)	**Mar 18, 2016 16:05**	**Marjorie Elizabeth Osborn Locke**: Daily goals can pull - apologies - sample URL added
v. 6	Mar 17, 2016 16:49	**Marjorie Elizabeth Osborn Locke**: Cannot pull daily goals from fitbit - part of unreleased scope
v. 5	Mar 11, 2016 22:10	**Marjorie Elizabeth Osborn Locke**: Accolade 2.a and Goals 2.a removed
v. 4	Feb 23, 2016 18:09	**Marjorie Elizabeth Osborn Locke**
v. 3	Jan 18, 2016 23:32	**Marjorie Elizabeth Osborn Locke**
v. 2	Jan 15, 2016 17:13	**Marjorie Elizabeth Osborn Locke**
v. 1	Jan 13, 2016 15:00	**Marjorie Elizabeth Osborn Locke**

Base Functional Specifications

- All teams must complete the specifications in this section.
- They are worth 60% of the marks allocated to the programming portion of the team project.

Start up and Exit

1. The application name is up to you, but the runnable jar **must** have "team" and your team number as a prefix
   a. Ex. Team 4 **must** use the name team4_FitbitViewer.jar
2. It **must** be possible to run the application by executing the command:
   a. java –jar target/*team#_JARname.jar*
   b. Ex. team4_ FitbitViewer.jar
   c. where team#_JARname.jar is the name of your team's jar file
3. The user must be able to quit the application at any time
4. On initial startup, the user should be able to authenticate through the Fitbit website using OAuth2 authentication (We will assist with this part)

5. User information (including configuration preferences, and authentication) must persist between runs of the program when appropriate

Daily Dashboard

1. The user must be able to view today's data for:
    a. Calories burned (out)
    b. Total distance
    c. Floors climbed
    d. Steps
    e. Active minutes
    f. Sedentary minutes
2. The user must be able to switch the dashboard to another day
3. The user must be shown the time at which the information was last updated
4. The user must be able to refresh the data

Best Days

1. The user must be able to view their best days for:
    a. distance
    b. floors
    c. steps

Lifetime Totals

1. The user must be able to view their lifetime totals for:
    a. distance
    b. floors
    c. steps

Fitbit API

1. The data requests to the Fitbit API may not succeed (e.g. access not permitted, rate limit exceeded); Your program should handle API errors internally, in a way that is not visible to the user unless appropriate
    - Ex. rate limit exceeded: If the user is actively trying to refresh data, but the app has exceeded the rate limit, you should tell the user to try later, but they should not see a crash message or anything to do with HTTP requests
2. Your application (and your use of the website) must adhere to the Terms of Service and Privacy statements which can be found on the Fitbit website
3. You must give credit to Fitbit for the data somewhere on the dashboard(s)
    a. Any use of the Fitbit logo or formatted name must conform to their graphical standards
4. Your project must include a way to test your application that does not hammer the Fitbit API (making hundreds upon hundreds of requests), and being able to test without access to the Fitbit API at all (Ex. with canned or stored data)

Additional Components Specifications

Combining additional components will make the team eligible for 100% of the marks allocated to the programming portion on the team project.

- The team should add the following components based on their interests.
- Each component listed is given a grade value which indicates how much the implementation is worth at maximum in the marks allocated to the programming portion of the team project.

Upon submission, teams will indicate which components they would like to be graded.

- The components added to the base specifications to be graded cannot total more than 100%
    - Your group cannot complete 110% worth of components and ask us to grade them all out of 100%.
- The components added to the base specifications to be graded do not have to total 100%
    - Your group could decide to aim for a 90% completion instead of a 100%.
- Teams can also propose additional components of their own creation in Stages 1 or 2 of submission.
    - These will be evaluated for difficulty, approved and assigned a grade value by the teaching staff at that time.
    - Unapproved additional components may not be submitted for grading in Stage 3

Custom Dashboard (10%)

1. The user must be able to hide or remove information in the dashboard
2. The user must be able to reveal information in the dashboard that had been hidden/removed
3. The user's dashboard preferences must persist between program runs
4. The dashboard layout should stand alone and look good no matter the content
    a. Just hiding the info isn't enough
5. If additional elements are added to the main dashboard, these must also be configurable

Time Series Data (15%)

1. Create a view for the user to see a time series of the following items for the day at 15 minute resolution at minimum
    a. Steps
    b. Calories
    c. Distance
    d. Heart rate
2. The layout and organization are up to you
3. They can be combined or separate
4. The user must be able to "zoom" into, at minimum, hourly intervals, within which the resolution is 1 minute at minimum

Heart Rate Zones (5%)

1. The user must be able to view daily heart rate zone information:
    a. Minutes in each heart zone
    b. Description of zones must be available
2. The user must be able to view their resting heart rate

Accolades (5%)

1. Create an award system and custom badges or reward system with at least 20 significantly different accolades
    a. You CANNOT use the badges that are given by Fitbit
    b. Come up with the system levels and calculations on your own
    c. They cannot be too similar to each other, or all based on the same measure (Ex. cannot all be step counts)
    d. They can be similar to other systems as long as those system are not copyrighted or trademarked
    e. They must have some kind of graphics (not just listed as text)
2. The user should be able to see their daily badges/rewards (depending on what kind of accolades you have)
    a. ~~The daily dashboard should display this information~~
    b. The user should be able to review whether they met their badges on any day they browse to
3. The badges must persist between runs of the program
4. The user should be able to see their lifetime badges/rewards (depending on what kind of accolades you have)

Daily Goals (5%)

1. The user must be able to set daily goals, or the program can pull the goals from the users fitbit account (found in: https://api.fitbitcom/1/user/[user-id]/activities/date/[date].json)
2. The program should inform the user if they are below, at or above their goals
    a. ~~The daily dashboard should display this information~~
    b. The user should be able to review whether they met their goals on any day they browse to
3. The goals must persist between runs of the program
4. There must be at least 4 goals
5. The goals must be different from each other, and cannot all be based on the same measure

---

Technical Specifications

1. The application must run on the Linux systems in MC 10
    a. You can develop on any system, but you must ensure that the final product will run within the lab environment (test early and often)

    b. The platform your program runs on will affect how your GUI looks. We can grade the GUI based on your main development environment, but the GUI must still be usable on the lab machine for testing.

2. The application must be developed in Java
3. The application must have a Swing graphical user interface
4. Your project must use git and Bitbucket for version control and repository storage respectively
5. It must be possible to both compile and package the application into an executable JAR file using the Maven command mvn package
6. All dependencies must be contained within your JAR file
7. Data persistence between runs must be stored using object serialization
8. The data must be gathered from the web service api.fitbit.com in JSON format using OAuth2 verification (NOT OAuth aka OAuth1)
9. The system must be easy to use. For example, a user must be able to navigate easily between screens, and must have the option of quitting or going back to a previous screen at all times
    a. Messages, as well as errors, must be reported correctly and consistently
    b. Keep in mind that popup messages are jarring to a user and interrupt his/her workflow. They should typically be used only in exceptional circumstances. When considering using a popup message, ask yourself whether or not there is another way to convey the same information
10. You must make sure that your code is commented with javadoc-style comments so that it can be modified and reused by others
11. The code implementing your project must be written by members of your team
    a. You may use libraries you find on the internet, please have them approved by the teaching staff before integration