CSCI-3308: Project Milestone 5
Project group: buffsgitbuff
Team members:

Casey Durham
Aaron Lu
Willem Mirkovich
Daniel Taylor
Tyler Tokumoto
Carlos Salazar

User Acceptance Test Plans

Create a document that describes how at least three features within your finished product will be tested. The test plan should include specific test cases (user acceptance test cases) that describe the data and the user activity that will be executed in order to verify proper functionality of the feature.

1. User login

   A given end user to the website needs to be able to log in again and access their favorite workouts as well as generate new ones personalized to them.

   To test this, the tester will generate many different custom template accounts, log out of each, and log back in to ensure that the settings are saved.

2. Workout generation

   Workouts generated by the website should be reasonably long, with reasonable exercises populating it, customized reasonably for each user depending on their strengths and weaknesses.

   To test this, the tester will access the website itself through the perspective of many different users with different defined strengths and weaknesses, then generate multiple workouts personalized to them.

3. Favorite workouts

   When a user does a workout and likes it, they should be able to save that workout so they can do it again later. These workouts should be accessible later.

   To test this, the tester will access the website from some user account, generate a workout, and save it. The tester will then attempt to access this workout again after logging out and back in through the favorite workouts section of their page.

Automated Test Cases

Provide a link to the tool you use to automate testing, or explain how to run the automated test cases or schedule time with the TAs to demonstrate your automated tests. Provide a copy of the output showing the results of the automated test cases running.

We will be using Python's automated test framework in order to ensure functionality of the backend. These tests are contained in the `buff2s/databaseTests.py` file, and can be run with the following steps:

1. Ensure all of the relevant Python libraries are installed using any Python package manager. Here is the list of necessary libraries:
    1. pymongo
    2. numpy
    3. scipy
    4. flask
    5. passlib
    6. unittest
2. Ensure that mongodb is installed (https://docs.mongodb.com/manual/installation/)
3. Create a directory for the database to exist *outside* of the git repository directory. This can be done with the command (from the git repo) `mkdir ../database`.
4. Start the mongo server with the command `mongod -dbpath ../<database directory>`.
5. Now that the database server is running, the database must itself be populated. This can be done by (from the main git repo directory) running the following command: `mongo < createDatabase.js`.
6. Now that the database is both running and populated, the tests themselves can be run with the following command: `python3 databaseTests.py`.

Here is a screenshot of these tests running on the database:

```
.....
----------------------------------------------------------------------
Ran 5 tests in 0.093s

OK

Process finished with exit code 0
```