CIS 526 Sprint 5 Review

Yuan (Tim) Su

Grant Unruh

**SPRINT GOALS**

During this sprint we had the following goals:

* Complete html views
* Merge created html views with ruby code to show real data
* Setup data collection from fitbit wristband
* Test html views
* Test Controllers

Accomplishments:

* Merge created html views with ruby code to show real data
* Setup data collection from fitbit wristband
* Test html views
* Performance testing the home page.

Remaining Tasks:

* Update data before it is sent to us from fitbit. Currently we’re receiving 0 for each of our values, even if we have taken more than 0 steps, for example.
* Fully complete html views. We ran into some trouble when running them from rails instead of just locally on our laptops.
* Test Ruby controllers. Once we all the data coming through correctly, and finish each page’s html, we can test the controllers for each page.

We will continue working on the remaining tasks and hope to finish them in the next sprint.

**SCRUMs**

This sprint we met about 7 times to SCRUM and discuss the project.

**TESTING WITH JMETER**

Neither of us had experience with jmeter before, so we began learning how to use it as soon as we were able to collect some basic test data from fitbit about our user. We plan on testing more in the next sprint, but we did a sample test of an http request of the login page of our site. We tested the site’s login page response time for 50, 100, and 10,000 users.

We found that with 50 users, it took between about 0.8 and 4 seconds to receive the response.

With 100 users, it took between about 0.3 and 7.5 seconds,

And with 10,000 users, the quickest time was around 0.6 seconds, but after the 205th simultaneous user, the users began getting back errors, and the few users that got responses were receiving them in around 100 to 300 seconds in some cases. This shows that our site would not do well with more than 50 users unless we created more dedicated servers to handle our requests.

We thought a good way to improve throughput and scalability is using resource pooling. The fewer times we need to access the database, the faster the program will be able to run. So if we grab as much as we can out of the database each time we access it, it will be better than grabbing pieces of data one at a time. We believing that caching won’t work very well for some parts of this app such as the step count, because the data stored for our step count in our app will be outtaded each time the user takes a step, so it would not be smart to continue to use same old information. On the other hand, we could cache the sleep, or weight values, since they won’t usually change as much in a day. If we had money to spend, we could always get more servers for a higher level redundancy.