**github.com**

**Austin Choi, Andrew Harper**

**CS 1632 – DELIVERABLE 3: Web Testing with BDD**

Summary:

GitHub was selected due to it being feature rich (clearly too much to easily do this style of testing), well known, and familiar. Fortunately, most actions on GitHub involve little to no dynamic content, making the site generally easier to test. In selecting what aspects to tests, I tried to select mostly ‘core’ features, namely account creation issues and control over repositories.

User Story 1 was testing the ability for a new user to create an account, and the scenarios given make to cover a large number of potential issues a user will have. For example, we test that on incorrect input of username or password an account can’t be created, specifically looking at site prompts at what is wrong and how. The tests are thus meant to focus on the user's view of the feature, in line with BDD.

User Story 2 and User Story 3 refer directly to handling a registered user’s repos, as that could be considered the core feature of GitHub. Between the two, ability to set and alter various settings is tested. Scenarios cover situations from what you decide on creation (US2) to changing ownership or deleting a repo (US3).

Issues:

An issue with the testing the site came from pure scope. There are many features that probably should be testing, and it would take much time to test them all properly. Another issue with designing the test specifically, is knowing how far the testing should go and what should be used to verify success. For example, on creating a new user we can assert against what the site should display on success, but we cannot be sure that the user was actually created correctly and rather just know that the page was displaying what it should if a user creation was successful. That seems to just be an inevitable fact of doing black box testing however.

Another issue that cropped up was that many buttons lacked proper ids. I had to find the correct buttons to test by looking at either the link text or simply the order they appeared on the page.

Biggest issue, and even with looking into it no satisfying answer was found, was how to handle authentication. The answer most pleasing was to simply have cookies set for the browser so that automatic user authentication occurs. Doing this introduces another issue in that when features for unregistered users or not logged in users are also tested. One hacky solution would be to start logged in, ending with logging out and then do any unregistered user testing. However, for our purposes, we mostly resorted to extensive hard coding of username/passwords (we would delete accounts when done with them, but still questionable) and even creating users for specific tests. These fixes are not a scalable solution, and introduce an issue of order the tests are ran in mattering.