David Stolz and Brandon Whitfield

ISA Project 6 Report

All Code can be seen in our repo: https://github.com/bjw4ph/ISAProject

Tasks we did:

1. Integration Testing:

First we pulled down the docker image for a remote selenium server with the FireFox web driver. We then used the python selenium package and python unittest framework to create our integration tests located in unitTestSel.py on the top level of our github repo. These tests were designed to go through all of the different steps a user could go through on our site, including

- Test_button_redirects
    - Goes through the various pages used to display information and check their redirects work, including the home page checking if the top 5 users and 5 most recent tasks are being uploaded and whether they redirect to the correct type of information page. Also the general info pages for tasks, users, and reviews, testing all those redirects
- Test_login
    - Goes to the login page, uses a login from our fixtures, and make sure the login redirects to the home page, and makes all of the front end changes after redirection, such as not displaying signup on the navbar, displaying profile/create task on the navbar, having logout instead of login on the navbar, and ensuring the auth cookie has been set
- Test_need_login_redirect_profile_create_review_then_search
    - Goes to the profile page first, which redirects to login because the user must be signed in to the view page. Selenium logs in and checks that we are at the profile page. From this page, we check that needed reviews are loading, and then check all of the error messaging until we finally create the new review. We then go to search, go to a review specific advanced search and then check to see that kafka indexed our new review into elasticsearch
- Test_signup
    - Goes to signup page, attempt various signups which are incorrect, such as not having all the fields filled and not having correct input type. Then does the correct sign up, checking to see if the user is now logged in, and is able to find the new user in the advanced user search functionality
- Test_create_task_with_search
    - Checks for login redirect initially, then creates a task, and uses advanced task searching to ensure the task is added to es
- Test_search_fixtures_from_batch_script
    - Makes sure all of the fixtures from the bash script were loaded up. Uses the navbar search box to do a general basic search which returns one of each type

of model. Then tests that each type of result redirects to the correct type of info page

Command lines for running these integration tests:

The testing script takes in two command line parameters, first is the ip of the selenium server, and second is the ip of the haproxy container. We used docker inspect to figure out those in the travis script. Once you have that the command from the top level of the git repo is:

python unitTestSel.py {SELENIUM_IP} {HAPROXY_IP}

Pictures of output:

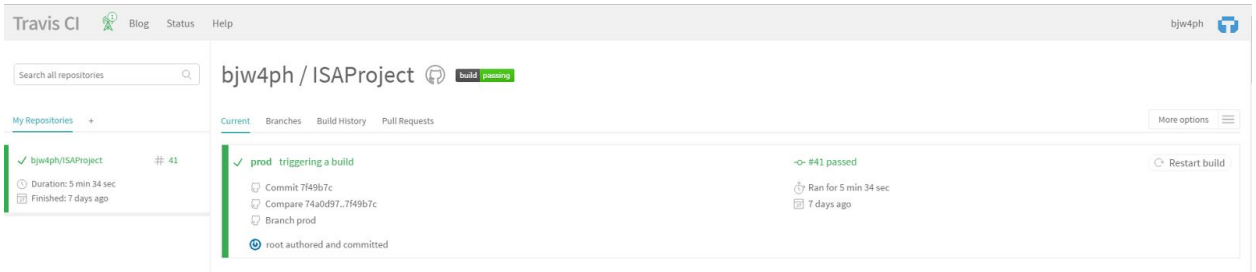
```
619 $ python unitTestSel.py ${SELENIUM_IP} ${WEB_APP_IP}
620 Command Line Argument: 172.17.0.6
621 Address: http://172.17.0.11
622 Address: http://172.17.0.11
623 http://172.17.0.11/
624 http://172.17.0.11/user/1/
625 http://172.17.0.11/task/1/
626 http://172.17.0.11/review/1/
627 http://172.17.0.11/
628 http://172.17.0.11/task/1/
629 http://172.17.0.11/review/1/
630 .Address: http://172.17.0.11
631 http://172.17.0.11/login?next=/createTask
632 http://172.17.0.11/login?next=/createTask
633 http://172.17.0.11/createTask/
634 http://172.17.0.11/createTask/
635 http://172.17.0.11/createTask/
636 http://172.17.0.11/search/?type=task&title=specific
637 .Address: http://172.17.0.11
638 http://172.17.0.11/login
639 .Address: http://172.17.0.11
640 http://172.17.0.11/login?next=/profile
641 http://172.17.0.11/login?next=/profile
642 Count: 3
643 Fill Title Error: Must fill the message title
644 Fill Body Error: Must fill the message body
645 http://172.17.0.11/profile?success=login
646 http://172.17.0.11/search/?type=review&title=Good
```

```
647 .Address: http://172.17.0.11
648 http://172.17.0.11/
649 http://172.17.0.11/search/?type=all&all=5
650 http://172.17.0.11/search/?type=all&all=5
651 http://172.17.0.11/user/5/
652 .Address: http://172.17.0.11
653 http://172.17.0.11/signup/
654 http://172.17.0.11/signup/
655 http://172.17.0.11/user/6?success=signup
656 http://172.17.0.11/search/?type=user&username=rand
657 .
658 ----------------------------------------------------------------------
659 Ran 6 tests in 59.043s
660
661 OK
662
663
664 The command "python unitTestSel.py ${SELENIUM_IP} ${WEB_APP_IP}" exited with
    0.
665
666 Done. Your build exited with 0.
                                                                        Top
```

The tests are printing out the various redirect urls which selenium tests as well as some of the error messaging

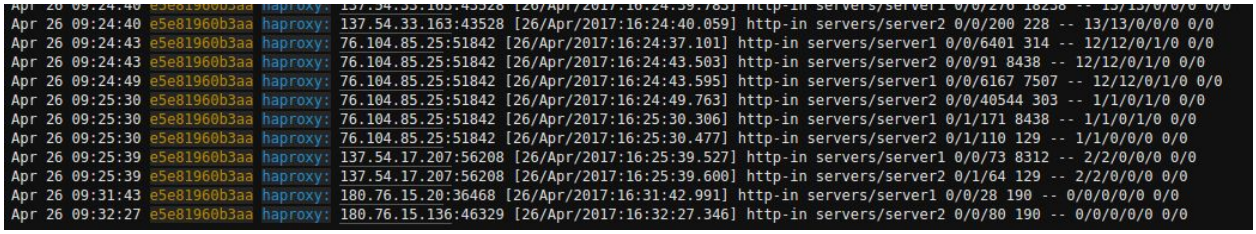
2. Continuous Integration

We first needed to update docker-compose.yml so that mysql-cmdline will create the db user every time we use docker-compose up. We then registered the github repo with travis ci and then added the .travis.yml file in the top directory of our github repo. This Travis file sets up the environment, calling docker-compose up in order to set up all of the containers. It then runs all of the django unit tests written for the models layer, and then runs the selenium tests in the method described above. When all of these tests pass, the build passes
Travis site: https://travis-ci.org/bjw4ph/ISAProject
Example Output from Travis:



3. Haproxy load balancing
Using round robin load balancing, we distribute incoming traffic to two separate Django web containers.

Example Output:



4. Digital Ocean deployment
Our “production” branch differs from the master branch in the following ways:

- DEBUG = False for all Django containers.
    - Static files are hosted by mod_wsgi
    - Allowed hosts are limited to containers that should be able to communicate
- We only expose ports on our haproxy container, no Django ports are exposed.

Link: http://104.131.95.232/