CI - Assignment 8

A.J. Feather
af2849@columbia.edu

Andrew Grant
amg2215@columbia.edu

Jacob Graff
jag2302@columbia.edu

Jake Weissman
jdw2159@columbia.edu

December 10, 2016

1 Continuous Integration

We used Travis CI as our integration tool and linked it to our GitHub repo. We set it up so that on each commit and push, Travis runs our entire test suite and reports on the coverage statistics as well.

2 Code Coverage

2.1 Report:

```
$ coverage report -m
Name                                 Stmts   Miss  Cover   Missing
------------------------------------------------------------------
app/__init__.py                          0      0   100%
app/database_methods.py                167      2    99%   184-186
app/database_objects.py                 27      0   100%
app/market_methods.py                   65     15    77%   16-18, 26-28,
39-41, 54-56, 82-84
app/multi_processing_handler.py         20     14    30%   7-8, 12-17, 21-27
app/order.py                           148     15    90%   38, 69-71, 107,
112-113, 116-117, 133, 145, 189-190, 207, 210
app/server.py                          206     54    74%   37-56, 71-72, 101-125,
132, 163, 182-184, 195-205, 279,288, 290-291, 294-295, 302, 320-331
app/transaction.py                     109     30    72%   60-61, 67-69, 71-80,
114-120, 123-140, 160-161, 181
app/validity_checker.py                 39      4    90%   12-13, 29, 31
------------------------------------------------------------------
TOTAL                                  781    134    83%
```

2.2 Remaining Problems

We had trouble running full system tests due to what was required in order to properly ping the server programmatically. This was also an issue with the multiprocessing code because we needed to ping the server in order to get it to run properly. However, there were a number of system tests included in the unit test files where we created, deleted and waiting for transactions to properly complete. In addition, we tested all of the internal classes and functions as best we could and our coverage takes care of all code we could test successfully via unit tests.

To make sure we test the full server and remaining code we performed the following system tests. Happily, there were no errors while performing the below full system tests, which indicates our unit tests and existing system tests are robust.

2.3 Performed System Tests

User story: As a user I want to be able to create a user profile and login/logout securely

Test: included in test folder

User story: As a user I want to be able to let the system complete a sell order automatically

Test: included in the test folder

User Story (2): As a user I want to be able to view the currently executing transactions, and As a user I want to be able to see past completed history

1. Test:
2. login using known username
3. amount to sell
4. select price
5. select limit
6. submit order
7. repeat steps 2 through 5 using a price below current market price, above current market price and at current market price
8. view history and verify transactions executed properly (large limit should not have sold)
9. repeat steps 2-6 excluding 3 and selecting market instead of limit in 4
10. repeat steps 2-6 excluding 3 and selecting time-weighted instead of limit in 4
11. view history and verify transactions executed properly

12. logout
13. login
14. view history and verify it still contains correct values

3 Resources

- Code Coverage Report = https://github.com/PsychicWaffle/4156project/blob/master/docs/coverage_report.txt
- Test Suite = https://github.com/PsychicWaffle/4156project/tree/master/code/tests
- Task board = https://trello.com/b/ZAo59Z8n/2nd-iteration-j-p-morgan-project
- Repo = https://github.com/PsychicWaffle/4156project
- Issues Tracker = https://github.com/PsychicWaffle/4156project/issues
- System tests info = (here) https://github.com/PsychicWaffle/4156project/tree/master/docs/ci-coverage-writeup
- Demo Docs = https://github.com/PsychicWaffle/4156project/tree/master/docs/demo-docs