Psychic Waffle Project Proposal

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

Since we have been gradually writing tests for our app throughout the development process, when we integrated the Travis CI tool and started more closely monitoring our code coverage, we were looking at around 50% statement coverage. This was a good place to start but nowhere near where we wanted to be. We set out on a quest to get as close to 100% coverage as possible. We mocked out a lot of system components and were able to simulate interactions in the system, including database and network communication, that made it possible to test much more thoroughly. Most of our python files are showing at least 75% coverage and some are showing close to 100% as some files were easier to test than others. Our most recent coverage report can be found using the link below. Unfortunately, we did not start tracking this document until we were almost done improving it so the history does not show how far we came over the last few days. Anyway, we were ultimately able to get our coverage up to 82% statement coverage! This was a large undertaking and we are very proud with how far our testing suite came.

While we made a ton of progress, we still fell short of our goal of reaching 100% coverage. With a lot more time we might have been able to get it there but we were starting to see diminishing returns at a certain point. Some issues that prevented us from reaching our goal was some areas of the code with hard-coded values. For example, we hard-coded the URL to the market exchange which did not allow us to test for cases when the exchange was down. This would have been possible with a more configurable environment, but we chose not to make those changes at the risk of messing up the current working state of the code right before time to demo. This issue played itself out in a few more cases where it was hard to force certain error conditions. Additionally, the code that handles the multiprocessing queue system was hard to test and hurt our coverage numbers.

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`