What went well

* More accurately defined scope. Effectively accomplished our main sprint goals (deployed the API to Heroku, Established Stock Page, Established Price Forecast Calculator, Setup User Management, Made fixes from last sprint), Worked better together than last sprint, More effectively distributed the work over the time of the sprint
* We think the UI of the website has really come together after switching themes and looks much more professional
* We started to do some pair programming this sprint as we had to start combining the front end and back end. Solutions for problems were found easier this way
* The Flask API has been hosted and all parts of the app can be tested on a local build of the app

What didn’t go well

* Despite better management, still a big push at the end of the sprint, Spent a lot of time on bugs that likely could have been solved much quicker
* We did not have data well prepared to be transferred from language to language. For example we had issues with creating a nested loop in typescript that could reach within a nested array of stock information.
* We ended up finding some ways to break our app at the very end of the sprint. Had to triage into the night. We believe the issue has something to do with the amount of tickers and positions a user adds and if they say a stock was free.

What could be improved

* Team up more to solve bugs even within our own systems and tickets, planned some coding parties but did not execute as well on them, communication when multiple pull requests are out
* Again hitting on that more even work distribution within the sprint. It's not uncommon for the workload to ramp up at the end of the sprint in a real job, but we would like to move away from that and move towards a more balanced one.

Challenges

* Issues in data transfer, lots of bugs, a lot of time spent setting up infrastructure which slowed down development, although we improved time management is always an issue
* Went through a few different javascript libraries before choosing one for the stock chart. Did not get the one we initially wanted due to some inject errors within angular, however the highchart seems to do well and actually has a lot more features than we originally thought and was very easy to format to consume API data.
* Late in the sprint, while testing our stocks page, we seemed to have broken our API connection to the DB on some accounts or atleast done something that makes all requests hang. We are definitely going to have to go through our code and add in some error checking for edge cases and how to navigate around those.