What went well

* Ability to incorporate 3rd party stock data API from Alpha Vantage and communicate with them regarding more requests per minute
* Creating a local Flask API to send the positions and pillar data to the front end from the backend processing
* Communication in regard to how the backend was going to look like in the front-end regarding UI
* The CI/CD pipeline to ensure the live application never went down including never having merge conflicts with our Pull Requests with frequent merging
* The login flow and implementation of Auth0 which offered a lot of other useful security tools for our website.
* A lot of work got done this sprint and should set us up well for the future sprints.

What didn’t go well

* Edge cases with business data: lots of different business with different financials so constantly finding edge cases to our data management
* Creating a Change in Shares Outstanding Pillar because of a fundamental flaw in how the Alpha Vantage API doesn’t change their historic share data when stock splits and reversals happen
* Due to lack of a defined scope for sprint 1, we created a lot of stories that might’ve created a very front-loaded PI and a time crunched sprint.
* Everyone had a few issues with Angular, something about having the wrong version of the “ng build” package off a fresh install.

What could be improved

* No Clearly Defined Sprint 1 Scope with the tickets created for the goals we wanted to complete
* Creating sprint documentation and comments in our code so others can easily help others find bugs and resolve blockers for others
* Use a ticker branch system instead of a local developer branch system
* Do a better job of getting tickets that need joint work done on them before the last week of sprint, so that bugs can be troubleshooted easier.

Challenges

* Adding authentication as a requirement for the app side
* As stated earlier, stocks from different countries and different business sectors cause for many tickers to have one off bugs with the data that is given to us and are having to be dealt with as they appear
* Hosting the Flask API via a live deployment vs having it run locally
* Many on the team had never used Flask and had been sometime since using Typescript in Angular so blockers took longer to solve than would have originally been expected
* There were some issues initially with the route navving of the website that would cause it to 404 if the page was refreshed, although we found a workaround by creating a copy of the index that would appear when the page would 404 on refresh. This makes it seamless to the user if there was an error or not.