## Bug Tracking

### Methodology

Bug tracking was handled via the same interface as our features. This was done through GitHub's Kanban-style projects board feature, accessible here.

As bugs were discovered, issues corresponding to the bug were entered into the project board in the `To Do` column. As they were claimed for work. they would be moved to the `In Progress` column. When fixed, they would be moved to the `Done` column.

### Bug Counts and Analysis

Backend

The backend was mostly finished and tested by the time dependent work on the frontend started. The only features implemented in the backend that are related to R1 requirements is file-serving with transparent redirection support. As concequence, the implementation of the backend API hasn't been tested against in a development scenario.

Frontend

Frontend work was started later, with significant amounts of work being completed in the last week before release of R1, with more devolpers working on it than the backend. The increased communication overhead, broader featureset to be developed, and larger quantity of untyped dependencies to integrate all contributed to a higher observed quantity of bugs.

### How many bugs did you collect?

1. Creating a post does not update the homepage feed
2. Creating a post does not update the user's Facebook feed on facebook.com
3. Entering blank for the create a Facebook post still creates a Facebook post
4. Entering blank and clicking search for the stocks search bar does not yield a warning or error
5. Resizing the input textbox for the Facebook post gets layered under the other web components on the same page.
6. Logging into the site using FB authentication while already having a Facebook account logged in yields the following error: "App Not Setup: This app is still in development mode, and you don't have access to it. Switch to a registered test user or ask an app admin for permissions." User most go to incognito/private mode to log in.
7. Merge error in package-lock.json prevented builds in Production from working.

8. Misconfigured error handling in backend API caused the server to rewrite all 300 level redirects and $>$ 404 level errors to be 500 internal service errors.

**How many open bugs do you have? And why are they open?**

6 bugs remain open. The first 6 listed bugs are still open as a concequence of integrating the Facebook API late into the R1 development cycle. Facebook limited its API about 10 months ago in response to abusive third parties with respect to their users' privacy. The neutered, now poorly documented API has been difficult to work with.

Bug #7 was easy to resolve by just rebuilding the project after deleting the lock file and merging the result.

Bug #8 was identified by sending GET requests to endpoints that should serve POST requests. This should have returned MethodNotAllowed errors, but instead InternalServerErrors were returned. This was fixed by not rewriting any non-application-specific error to InternalServerErrors, and instead allowing the framework-specific errors to pass through the error-rewriting function untouched. There wasn't any meaningful impediment towards work on the frontend or this being a failure to meet a requirement; this was more of a temporary inability to follow the HTTP standards and REST guidelines.

**How has the bug collection/mitigation process helped or hurt your project.**

There is an inherent time overhead to tracking bugs in a more formal manner instead of just telling each other to fix them when we meet. Formally keeping track of them provides a means to ensure that each bug is addressed or at least known about. Bugs that appeared late in the development process have not been addressed in time for R1, but because they are recorded, they should be resolved in time for R2.