Bug Tracking

Methodology

We only tracked bugs that A.) Made it to the production environment, and B.) Were identified by another member of the team.

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

1. Setting the Zip Code to an arbitrary string that is not an actual zip code caused the Weather page to experience an unrecoverable error. This was fixed by handling the case where the external weather api returns a non-200 status request.
2. As of the time of this writing, the calendar page doesn't actually get events, nor can it create them. This is due to underspecified query parameters in the get request, and sending the wrong JSON in the post request. There is no formal ticket for this bug, but the ticket for the completion of the calendar integration remains open instead. It really is just an incomplete ticket, but the incomplete status of the work causes crashes in the frontend, and therefore we consider it a bug.

Backend

The backend had a couple of bugs, mostly caused by path matching specifications not terminating early when that is expected. This caused some calls to activate routes with the subsection of the path actually present, leading to 400 responses. These were entirely detected by Henry and were fixed before anyone else noticed.

Frontend

Poor branch hygiene led to some contributors delivering patches that due to significant manual merging, clobbered arbitrary sections of code, leading to regressions, and occasional code bloat. Given the untyped nature of the frontend, more errors were present throughout development.

How many bugs did you collect?

Two

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

One. The person responsible for completing the task has not completed the task.

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

It wasn't practically useful. Most submitted pull requests had glaring bugs that Henry had to fix before merging, or immediately after. It wasn't practical to itemize all of these, especially because compiler errors often obscured other compiler errors, and so fixing them one at a time was the only possible route. Most work for R2 not done by Henry was performed in the last two weeks of class, so intra team communication with respect to bugs wasn't useful, as Henry could just fix them as he encountered them.