MariGold

Team 5 - Sprint 2 Retrospective

Andrew Bass, Will Borland, Pravin Sivabalan, Devin Sova

What went well

Over the course of Sprint 2 we extended upon our basic app to include extended features and a more complete UI. We added features that gave our app real use, whereas in Sprint 1, our finished product was not useful in its own right. We tackled ambitious tasks on both the frontend and the backend.

In terms of the front end, we managed to implement a much more complicated user flow successfully. Our search functionality for finding medications was reliable, and the process for adding medications in general was greatly cleaned up. We also did a good job of interconnecting all of our views, such as how we were capable of showing detailed medication information from a variety of contexts without code duplication. One of our more intimidating user stories for Marigold, medication conflicts, was implemented largely successfully. By ensuring the frontend was given cleaned up data from our API, we were able to robustly show conflicts to the user, with only minor bug fixing required to fully polish the feature. We also managed to adds visual enhancements to the frontend as well, such as the usage of spinners to show when the app was busy doing network requests.

As for our backend, our hard work in creating a flexible foundation to build off of in Sprint 1 paid off as we tackled more difficult features. Our clean separation of database and routing code allowed us to easily add the required routes for conflicts and medication lookups without laboriously refactoring our code. We also gained great knowledge of two very useful APIs for our backend, the National Institute of Health Rx family of APIs and the FDA’s openFDA API. Rather than learn the bare minimum needed to integrate these APIs into our backend, we took the time to learn them in depth. In doing so, we’ve prepared ourselves for further success in Sprint 3 as we dive deeper into these APIs.

What did not go well

While we were able to meet all of our user stories, there were a issues involving time management and software testing. The majority of our work was done closer to the deadline causing panic and rushed code. Some of which needs to be implemented in a better fashion. It’s important we space out our time and communicate where we are at with our teammates. We need a roadmap that shows day to day or weekly targets and stick to it. We work off each others code and need to wait for teammates to finish parts before we start working on them. Working close to the deadline isn’t fair to teammates that need to work off of your last minute work.

We had issues with user stories overlapping with each other in terms of the code being touched. This made it so that we had to do manual merges of our code due to the fact Xcode made handling merge conflicts difficult. This created the chance for a lot more things to go wrong. Things worked out where we were able to put things together and merge, but there is not guarantee that this will happen every time. Another thing that did not go well was our testing. Out acceptance criteria for testing was not established well. This led to less than ideal testing and brought with it some unforeseen problems.

Below are the things which were unsuccessful during our second sprint:

- We did not fully test every user story and many of our tests were not thorough enough.

We were able to complete all acceptance criteria this sprint but the sub par testing reveals flaws with our implementations that need to be corrected.

How can we improve?

We continued some successes from Sprint 1 into Sprint 2, but we need to improve things to ensure success in Sprint 3. Our time management of components that link together was not optimal. Sometimes a team member would have to wait on another team member to get a component done before they could start work on another component. To mitigate these problems we need to analyze what needs to be done first so that we are not waiting on one another. Another time management improvement we can make is pacing out our individual work so that code and merging is not rushed. This can be improved by setting team deadlines to when a certain feature or task will be done and hold each other accountable.

While we improved our documentation in Sprint 2 we were still sometimes unsure on what should be passed between the frontend and backend. We should continue to properly document and communicate our backend endpoints. Github's builtin wiki system will be very helpful in this regard. Better communication will also be beneficial in this regard. We'll also use the Github project tracker to communicate our current status with our various user stories.

To ensure our final product is as bug-free as possible, we'll refine our testing criteria and create more tests that explore various edge cases in both our frontend and backend. We'll also do testing continuously throughout the Sprint, rather than save tests towards the end the our assigned user story.