Team 5 - Sprint 1 Retrospective

Andrew Bass, Will Borland, Pravin Sivabalan, Devin Sova

What went well

Over the course of Sprint 1 we set up the basis of our product. We were able to meet all the user stories and all but one acceptance criteria. Every team member worked hard on their portion of the project while providing meaningful feedback about other portions. While working through the three weeks we never encountered any majors barriers. The basis of our Flask framework backend and iOS front end have been setup along with basic functionality. After the first week the team started working more efficiently when they got more familiar with the frameworks and how we want to structure and shape the project. We prepared ourselves to take on the more complex tasks of Sprint 2.

Below are a generalization of the tasks that we successfully completed in sprint 1:

- Created a backend API that allows the frontend app to communicate with our Linux server.

- Implemented a database that communicates with our backend server that securely stores users information.

- Created an iOS app that allows a user to make an account, login to that account, reset the password for that account and delete that account.

- Built upon the iOS app the ability for users to add and delete medications to their account.

- Setup modules to test our backend and iOS code.

What did not go well

While we were able to meet all of our user stories, there were a couple of issues during development, primarily in regard to our workflow. On both the backend and frontend repositories, there were moments where we accidently overwrote each others code. While these setbacks weren't fatal to completing Sprint 1, it's important that we avoid these kinds of scenarios as we enter Sprint 2 and take on harder tasks. There were also issues with communication in the group where we did not fully communicate what we were working on. This may be the

root cause of the overlap in work. It also may be better to decide on how the design looks before actually implementing it in the code. This is due to the fact that we had conflicting design languages, and immediately implementing those designs caused issues later on with code having to be rewritten.

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

- “Given the backend mailer works, the backend will send an account deleted email to the account email.”

We failed to complete this acceptance criteria by running out of time. Completing this task for Sprint 2 will be fairly easy, as the necessary foundational work is complete (primarily, getting the mailer setup and accessible from Python).

How can we improve?

We had a good start in Sprint 1, but if we want to ensure success in Sprint 2, we need to solidify our workflow and ensure that we fully meet our planning document. In our next planning document, we need to split up tasks on the frontend more evenly across our team members. Furthermore, it's important that we relegate the design of individual components to a single person. We had design conflicts that could’ve been avoided if we delegated isolated responsibilities on the frontend.

It is also vital that we fully understand the consequences of the git commands we run. We learned this lesson the hard way. Doing doing force pushes or forgetting to pull never ends well. We also need to take better advantage of features on GitHub. We didn’t use the issue tracker to keep track of what we were currently working on, which could’ve prevented some of the redundant design work.

To ensure we never miss an acceptance criteria again, in the future, we’ll closely examine our planning document and make sure our acceptance criteria are fully meet, or, if a criteria is no longer valid, have it modified on the document.

While our current backend API is very simple, as we grow more complex endpoints, we need to create better documentation. One tool that can help is called “Swagger”. It lets us describe our API using the YAML format and export documentation that can be added to our GitHub repository.