amora

Team 4 - Sprint 2 Retrospective

Alex Hardewig, Zach Johnson, Benjamin Kahlert, Eric Vondrak, David Wood, Ian Zanger

What Went Well

Sprint 2 went well overall. The team is progressing rather smoothly through the middle phase of our development cycle and we are prepared to enter the final sprint. The major risk for this sprint was getting the Today View working, and through a lot of hard work by Alex and some support of other members, we were able to get it working. This means that all the major features of our application are implemented, at least in their basic forms, so that in sprint 3 we can focus on additional “nice to have” features and polishing/bug fixing.

We successfully implemented every major feature of this sprint. Users can now set the number of hours they have to work in the day, and those hours will decrease as they add tasks to their Today View. Users can now comment on tasks and have their comment appear alongside their profile picture and the date and time the comment was made. Users can edit their comments and see the timestamp of when the comment was last edited. Many manager-only features were added, including changing of project color/name/description, adding of users after project creation, promoting other users to managers, giving other users creator status, demoting users, and duplicating a project with the same users but a new color and name. Users can now be assigned to tasks - their profile picture will display along with that task and they will receive a notification regarding their assignment. Users can now allow desktop notifications which will inform them when they have an Amora notification. Projects now have descriptions that will display under the title and are included in invites sent to team members. The Today View is now implemented, meaning users can drag tasks into the right of their view and see the task turn into a draggable box. This box matches the color of the project it is a part of, and is sized based on the estimated time to complete that task. The tasks can be rearranged in the Today View and this setting will persist across refreshes. Tasks can be removed from the Today View by dragging them out of it. Team members can view each other’s Today Views by clicking on their user profiles.

The team continues to maintain open discussions about the status of the application, design choices, and our personal workload. When there is an issue with the application that needs addressing, we discuss it in our group chat and come to a unanimous decision about how we should address it. If it's a bigger issue or worthy of extended discussion, we use an online tool to find a time we can all meet the next day for an impromptu meeting. In these meetings we hear about what is going on and offer possible solutions, eventually coming to a consensus. When there is a design decision to be made, or if a past decision is no longer looking feasible, we discuss that and the pros and cons of sticking with our older choice versus going in a new direction. We try to consider it from the standpoint of practicality, user experience, maintainability, and feasibility, and generally it works out well. Further, when one of our members has issues with a strange bug or can't figure out how to implement something, other members step in to help. This has proven invaluable as having more sets of eyes on issues helped us resolve them much quicker, saving us all time in the long run.

What Didn't Go Well

Though sprint 2 went well overall, there are still things that could have gone better and we will aim to improve them for sprint 3.

Though the Today View is functionally implemented, it still leaves a fair amount of room for improvement. All the basic functionality is there, but the dragging can be a little finicky, especially with dragging tasks into the view. The boxes still look fairly rudimentary, so we will aim to improve the look of them and make them fit the rest of our visual design better. We also need to test the Today View functionality even more, as it leaves a lot of room for bugs that we might not have found yet. Some of these tasks may prove challenging since we are using a library that only one of us has some limited experience with.

We have seen an increase in the number of bugs that are cropping up as we test functionality. Our application is getting more and more complex, and more information is synced with our database then ever before. Information is appearing in multiple places in our application, so it is leaving a lot of room for bugs where this information is not properly synced as it should be, or is delayed in syncing with the database. With the Today View we are using a library, which involves open-source code that we may not fully understand, further exacerbating this problem. Though we were able to resolve most of our issues, it took us off guard considering the relatively bug-free sprint 1.

The team has noticed a decline in quality of our coding standards as this sprint progressed. There are fewer comments being written and they are less descriptive. We have had some unfortunate instances of having to repeat large swathes of code due to earlier code structuring decisions that we didn't realize would cause issues at the time. This has made it harder to find and fix bugs than before and really increased our development time in the third week of the sprint.

Week 1 of this sprint was not as productive as it should have been. Few of us reached the number of hours we were aiming for, mainly due to the busy time of year and other exams/projects/labs cropping up at the same time. To make up for this fact, we had to work more hours in weeks 2 and 3 to reach ~30 each and complete all of our promised user stories. This was not ideal as it caused us to rush more and likely worsened the issues mentioned above regarding code standards. Further, if we had hit any major roadblocks in the third week we may not have had time to resolve them.

How We Will Improve

In sprint 3 we will aim to improve on our process from sprint 2 and avoid the mistakes outlined above. We need this sprint to go smoothly so that we have a final project that both looks and functions well, without bugs and including the features we set out to implement.

We will set aside time to refine the Today View so that it is the best it can be. We will work on improving the dragging component and the visuals of this portion of our application. We will focus a greater amount of testing on this section so that we can catch as many bugs as possible with the implementation, and leave Alex more time to fix them as the lead developer of this feature.

We will address the uptick in bugs by allocating more time for bug fixes and testing. We know more of what to expect as we add features on top of our existing ones, as we have fixed dozens of bugs already. We now have more of an idea about where bugs often crop up, such as when features involve differing manager and user permissions, syncing of data with the database, and syncing of data with different portions of our application. We will keep these in mind as we code so as to better avoid bugs in the first place. We will also start maintaining a bug list so that we can keep track of what has been fixed and what still needs to be. This will help when one of us encounters a bug with a feature they didn't implement; they might not

know how to fix it, but they can keep a record of how it occurred so the proper person can address it. Improving our coding standards will also help cut down on the number of bugs.

We will place more importance on writing maintainable code that is commented well and follows good coding standards. We will avoid duplicating code in different places of our program, which will make finding and fixing bugs faster and easier. We will spend more time thinking about and discussing major code structure decisions, to try and anticipate any issues these decisions could cause in the future. Though we essentially know how to do all of these things already, we will allocate more time for them in this sprint. We suspect this has become an issue because of the increased time commitment towards the end of sprint 2, which led to rushing and decreased code quality. Our plan for sprint 3 will take this into account so we can avoid repeating these mistakes.

We will address the issue of backloading work in the sprint by planning better for potential time issues. Although it worked out in this sprint, doing the majority of the work in the second half of the sprint is risky and leads to code issues as discussed above. Unfortunately, busy weeks like week 1 of the last sprint do happen, so all we can do is better plan for them and try to foresee them. We will discuss prior to sprint 3 what user stories we plan on implementing in each week, and try our best to follow that plan while informing everyone if anything changes. We will also try to anticipate our workload for these weeks so that we can distribute the amount of planned work appropriately across the weeks. If we foresee week 2 to be busy, for example, we will try to get more done in week 1 in anticipation of that.