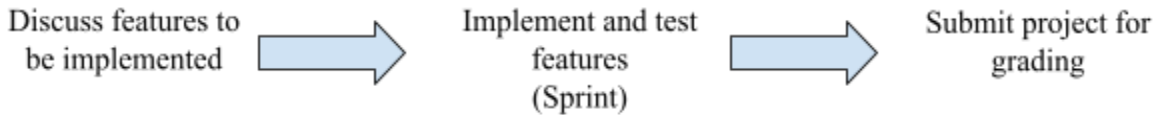
Process for Sprint 1

For the first sprint, our process model was very simple and very lacking:



While discussing features to be implemented, we also assigned which parts of the project each person would be responsible for- as seen in our UML architecture.

For team meetings, we almost only used the class time, but we had issues with attendance. We also didn't check in with each other daily, leading to extended periods where there was no discussion about the current state of the project.

Our pull request process was also vague and not very consistent. We decided that Jenna would be the one accepting pull requests into the master branch, but didn't set up any process for code review before adding code; thus almost everything was being added as long as there were no major conflicts immediately evident. This lead to a lot of errors being created that needed to be fixed with more pull requests.

We also kept pushing the deadline for code back, until pull requests were being accepted just a few hours before the deadline, which wouldn't be acceptable in a real world scenario.

We didn't have a real code review process, and mostly just accepted code without going over it with each other, leading to team members knowing how their own code worked, but having no idea what was going on with other parts of the code.

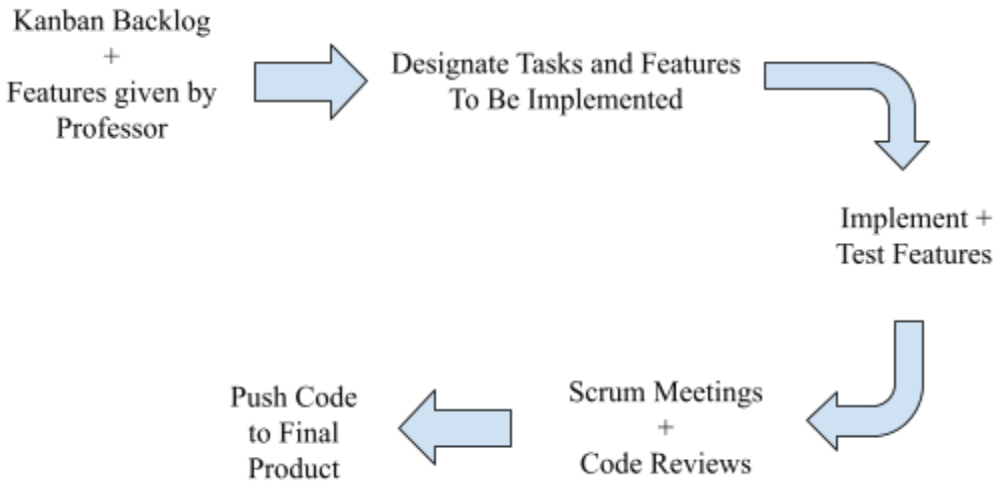
Performance reviews were done individually, but lacked any "base format" to follow, so each member ended up doing something slightly different.

Backlog items were very rarely added- even when ideas were discussed between us. We didn't make great use of the KanBan board, and barely updated it after putting it together on the first day.

From the key scrum practices, we failed to keep up with our product backlog. Although we did, by virtue of the due date for the first sprint, follow the timeboxed sprint practice, we didn't check in with each other often enough during the sprint, and not all of our assigned features were able to be implemented. Since attendance was shaky, we weren't able to make very many decisions through consensus (as indicated by the self-organizing teams practice), and most decisions were made by just one or two team members.

Process for Sprint 2

Going into the second sprint, we aimed to have a process model more like this:



This is better than the first sprint, since through the feedback, we have learned how important it is to have code reviews to make sure that, after a pull request, we don't have to do ten more pull requests to fix what was broken by the last one.

After discussing with other teams, we decided the best time to do code reviews was in our team meetings. Each member would be able to put up pull requests, but no pull request would be merged until the code was looked at and reviewed by the other team members. Jenna was still in charge of the pull request process.

We tried to remember to mention when we did these code reviews in the meeting notes, but we forgot at times- this is something that should be fixed in the final sprint. As well, we also put some of our code review in the pull request comments before merging.

Since items were added to the product backlog, we were able to assign them to team members and start working on them, meaning that many more features were implemented this sprint compared to the first one. However, there were times where a team member would be assigned something, and then this assignment would be changed three more times before it got to the person who should be working on it. This was due to being unclear of what modules each task would involve, and it led to big mistakes like two people working on the different implementations of the same task.

We attempted to follow the timebox sprint practice by setting small deadlines at different points in the sprint, with the goals we wanted to achieve by each of these deadlines. (We wanted to catch up on miss features from the first sprint by Oct 30, have everything from the new sprint implemented by November 6th, with this being the final deadline)

However, some of these deadlines were either not met or extended due to features not being done in time, and personal emergencies or time conflicts with scheduling meetings. This isn't a good strategy to have, as deadlines exist for a reason; as discussed in the notes, by the scrum guidelines, if a feature is not ready to be implemented, the sprint is never extended.

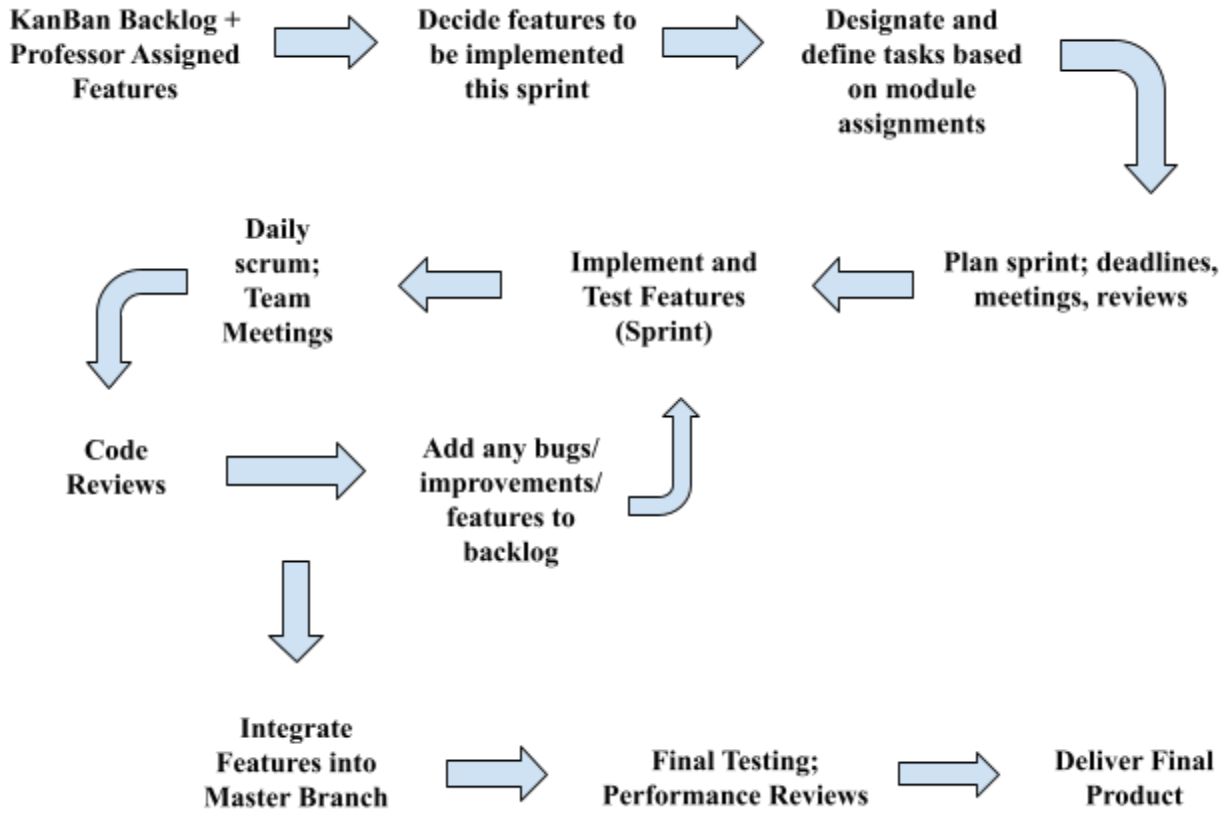
Attendance was better for this sprint, since we made sure to have more meetings outside of class time that everyone could attend. This way, the entire team was able to weigh in on decisions, instead of just one or two people deciding everything (self-organizing team practice).

Early on in the sprint, many items were added to the backlog, and the KanBan board was updated with labels and assignments, so that it was clear who should be working on what. This time, the KanBan board was used and updated during the sprint to make it easy for team members to check on the progress of the project.

Overall, compared to the first sprint, we were a lot more organized and communicative with each other. However, there is still a lot that can be improved on for the second sprint, such as more frequent code reviews, daily "check-ins" to see how everyone is progressing and to re-assign tasks if needed, and being more clear about who is doing what.

Planned Process for Final Sprint

When discussing what could be improved for the final sprint, we ended up on a process like this:



This process model is obviously a lot more developed than the previous two. The first few steps, up to implementing and testing features, would all be done on the first day or two of the sprint, to avoid any issues with planning further down the line.

An important part of this is “Designate and define tasks based on module assignments”. The idea with this step is to clearly designate which modules each task is involved with; for example, if a task has to be implemented in the server module, or through app logic. By doing this, it will be a lot clearer and a lot easier to assign everyone the proper tasks.

The middle loop would indicate a daily check in of working on features, discussing these features and any issues we’ve encountered with the team through discussion and code reviews, adding any features or bugs that come up from the reviews into the backlog, and then repeating this process every day. Finally, once everything was done, we would integrate everything into the master branch- the final product- and then do our final testing. We’ve decided to treat the performance reviews as an “end of the sprint” process, where we reflect on the sprint and how things could be improved for next time.

Code reviews will be done whenever a task is completed and ready to be implemented, and a pull request is made. This will be done at the daily check-ups, or during a team meeting if there is a lot of code to look at. Any completed code reviews will be mentioned both on the pull request itself and in the team meeting notes.

We developed a new pull request process near the end of Sprint 2 that we will be using for this sprint.

*Whenever a group member is ready to make a pull request:

1. Explain what task progress is being made towards
2. Explain how the code works
3. Allow time for Team members participate by conducting code reviews based on the SOLID and any other relevant design principles before the merge.
4. If anyone spots an issue/ room for improvement we will to document this in the issues table/ KanBan board backlog. If a project-breaking the issue is spotted, we will close the request.
5. There should be no major code conflicts that need to be resolved when a pull request is made.
6. Finally, the pull request is merged.

* Note: This protocol does not apply to pull requests involving updating meeting notes or documentation notes.