Slaq

CS5500 Managing Software Development

Team 207

Venkatesh Koka

Prajakta Rodrigues

Omar Tuffaha

Sean Ylescupidez

Table of Contents

Summary of Project Goals --------------------------------------------------------------------------------- 3

Result of Project --------------------------------------------------------------------------------------------- 4

1. Completion ------------------------------------------------------------------------------------
2. Quality -----------------------------------------------------------------------------------------

Developmental Process ----------------------------------------------------------------------------------- 5-6

Retrospective ---------------------------------------------------------------------------------------------- 7-8

**Project Goals**

The goals of Slaq were collected based on a backlog of product requirements. The product backlog requested a lot of specific features for this instant-messaging app including the ability to make group chats, recall a sent message, and having special government wiretapping privileges. Essentially, Team 207 came up with a major goal for this product: designing a flexible and robust application that allows users to message each other and perform a series of actions relating to messaging one another. However, this generalized goal was broken down into four main goals that were divided up into four different sprints.

Sprint 1 possessed a variety of major goals that aimed to kickstart the project. Specifically, the team sought to develop a deep understanding on the product that the client requested, understand the starter Prattle code, create a backlog of tickets in Jira, and add technical features that will increase the team’s organization in the future. To understand the client’s desired product, the team spoke with the client again, created a use case diagram, and fleshed out a list of functional and nonfunctional requirements. Then the team took the list of requirements and transformed them into a work backlog in Jira for tracking. After Jira was completely set up, the team also introduced both smart commits and build status notifications on GitHub to further increase the organization of the team. Team 207 spent several hours going through the given Prattle code and videos to understand the code. The team wrote unit tests to increase the line coverage to 85% and branch coverage to 80% throughout their process of learning the Prattle code.

Sprint 2’s goal was to completely set up the rest of the development environment and implement the foundational architecture of sending messages in a group and a dm. This included setting up a local database, setting up a production-level deployment to AWS, and setting up a remote database. This sprint also included creating a user login and password. These goals were set in place as a major foundation for the rest of the project.

With all of the development environments configured and the foundational architecture set up, Sprint 3’s goal took the team into implementing the core functionality of regulating group chats and user relationships. In terms of regulating group chats, this sprint incorporated adding moderatorship, group invites, and removing users from groups. With regard to user relationships, this sprint introduced friend requests and notifications.

Sprint 4 tied the rest of our project goals together. Sprint 4’s goal was to add the wiretapping privilege that is dedicated to governmental users along with adding extra user and group functionalities. This included adding group chat passwords, changing moderators for a group, deleting groups, searching for users on the application, recalling a sent message, translating messages, and adding parental controls. This sprint was one of the best in terms of sprint goals because it addressed a lot of the different requested features from the given product backlog.

Overall, the project goals were all successfully achieved because the team implemented a fully functional instant-messaging application that allowed users to message each other in direct messages and in groups, interact with one another through a series of commands and navigate through the application’s features with a help command. The application also sent notifications when appropriate and offered the government a special wiretapping feature, as requested.

**Results**

**Completion**

The project as a whole was very successful. Team 207 defined success as the following: implementing a fully-functional instant-message application that allows users to perform a series of core and unique functions while also being able to interact with one another. This section of the report will address how the team’s results mimic the team’s definition of success.

In terms of a fully-functional instant-messaging application, Slaq allows users to login to their account with their username and password. If the user does not have an account, it will allow them to create one with a specified username and password. Slaq enables users to send and receive direct messages to and from any of their friends. It also provides users the option to create group chats, making them the admin/moderator of that group. Messages can be sent and received from different groups and direct messages at the same time, but they will only show up if the user is on the corresponding message window. Still, a fully-functional instant-messaging app should notify a user when they receive messages, especially if they’re from another message window. Slaq accommodates this by sending out notifications to users when they receive messages and other important notices (which will be addressed later). With the described features of logging in to a profile, sending and receiving messages from direct messages and group chats, and receiving notifications, it is clear that Slaq is a fully-functional application.

Still, a fully-functional application may not be interesting if there are not a series of extra features added to it. Part of Team 207’s definition of success is to allow users to perform a series of core and unique functions while also being to able to interact with one another. Here is a list of core features that were implemented within Slaq: inviting users to group chats, removing users from group chats, adding, removing, and changing ownership of a group chat, deleting a group chat, checking which friends are active on the application, searching for users on the entire application, setting a user’s status to do not disturb, checking recent notifications, and checking group invites. Notifications were added for invitations to group chats, the deletion of a group chat, and the changing of ownership within a group.

In terms of unique functions, Slaq accommodates the special governmental privilege of wiretapping a user. This means that the government can retrieve a full history of sent and received messages from and to a user, even if the messages have been deleted and even if the user was part of a deleted group. Slaq also allows users to translate a message with a language of their choice, supporting over 100 languages. Next, Slaq enables a user to recall a sent message within a group chat or within a direct message. Last, the team implemented parental controls to block out swear words, which is applicable when a young user is using Slaq.

To allow users to interact with each other, users are capable of sending one another friend requests, which is paired up with a notification to the desired user. Once two users are friends with each other, they are allowed to direct message one another and view one another’s availability status on the app. Please see Figures 1-4 for statistics relating to the backlog throughout the project.

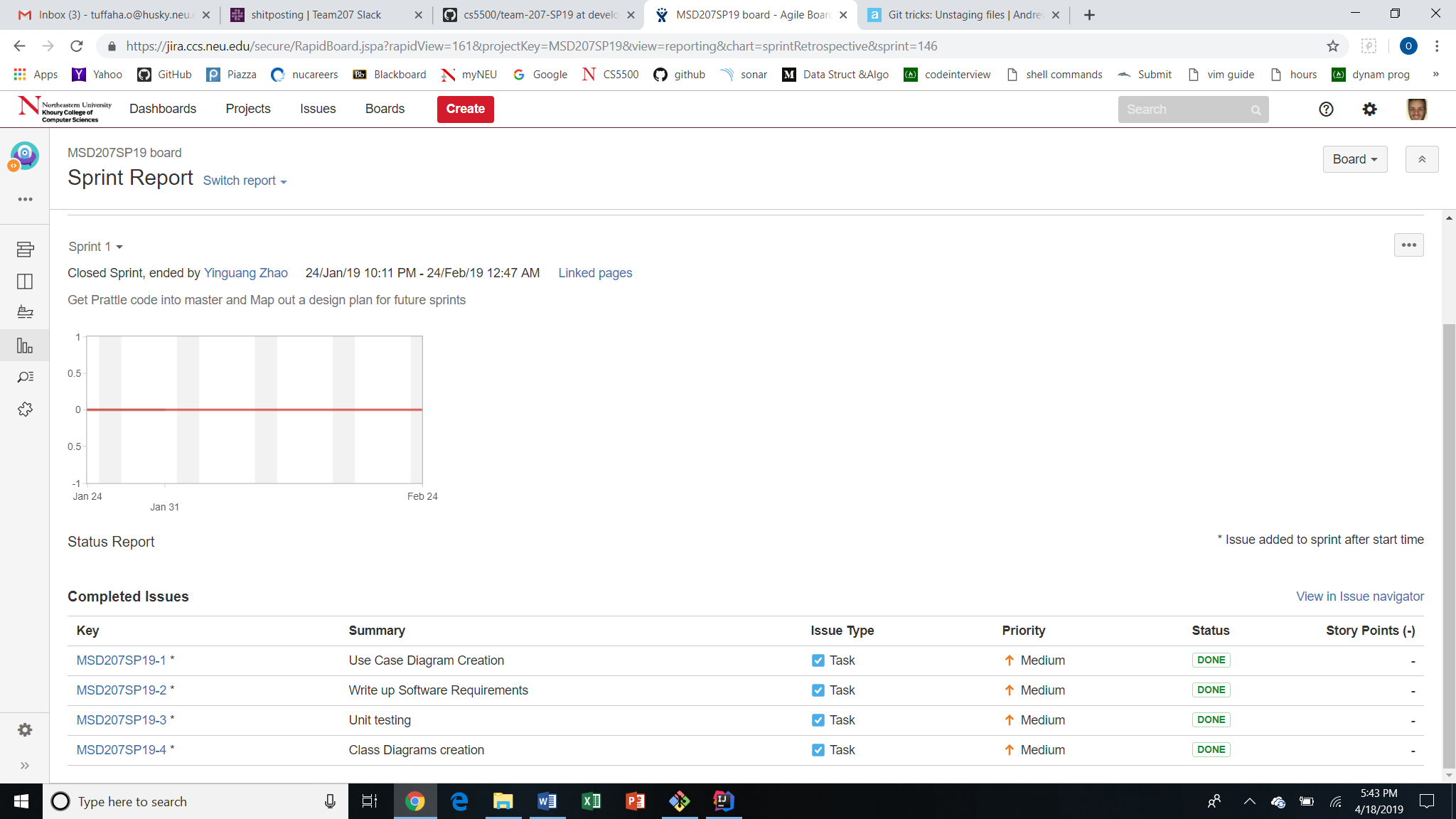


Figure 1: Displays the backlog statistics for Sprint 1.

Note that the team was still learning how to use Jira during this Sprint and spent more time setting up kickstarting the project than doing any development. Based on that, there were only 4 major stories and the team did not point them, but they were all finished.

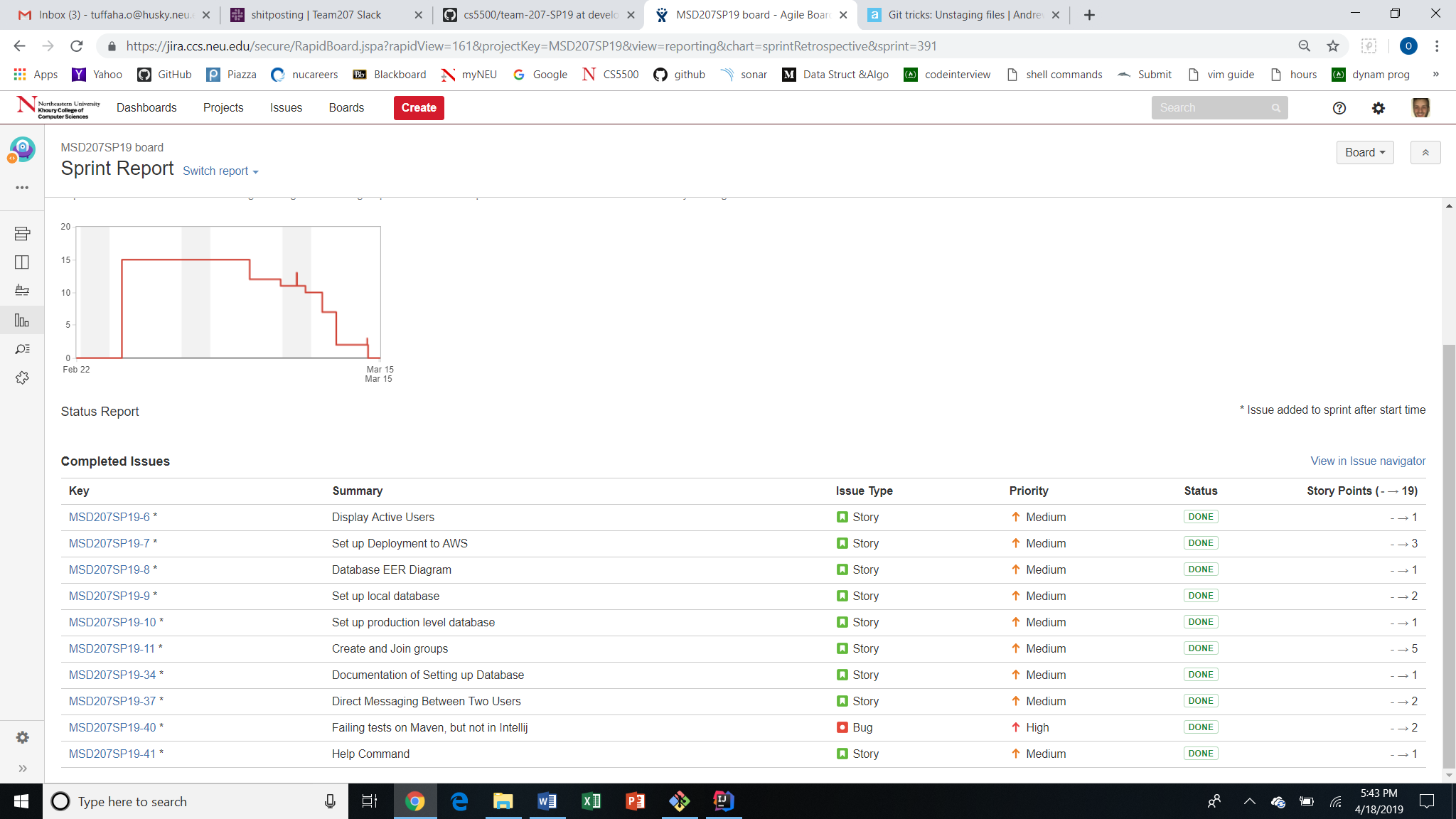


Figure 2: Displays the backlog statistics for Sprint 2.

Shown above are the tickets for implementing the foundational architecture of Slaq along with finishing up the development environment by creating a local and production level database. Note that the stories were pointed for this sprint. There were some jumps in the burndown chart that is shown in the top left of Figure 2 because the team finished some work early and decided to pull more work into the sprint backlog. Also, there is a long horizontal slope in the burndown because Spring Break occurred right in the middle of Sprint 3. Still, all of the work was completed.

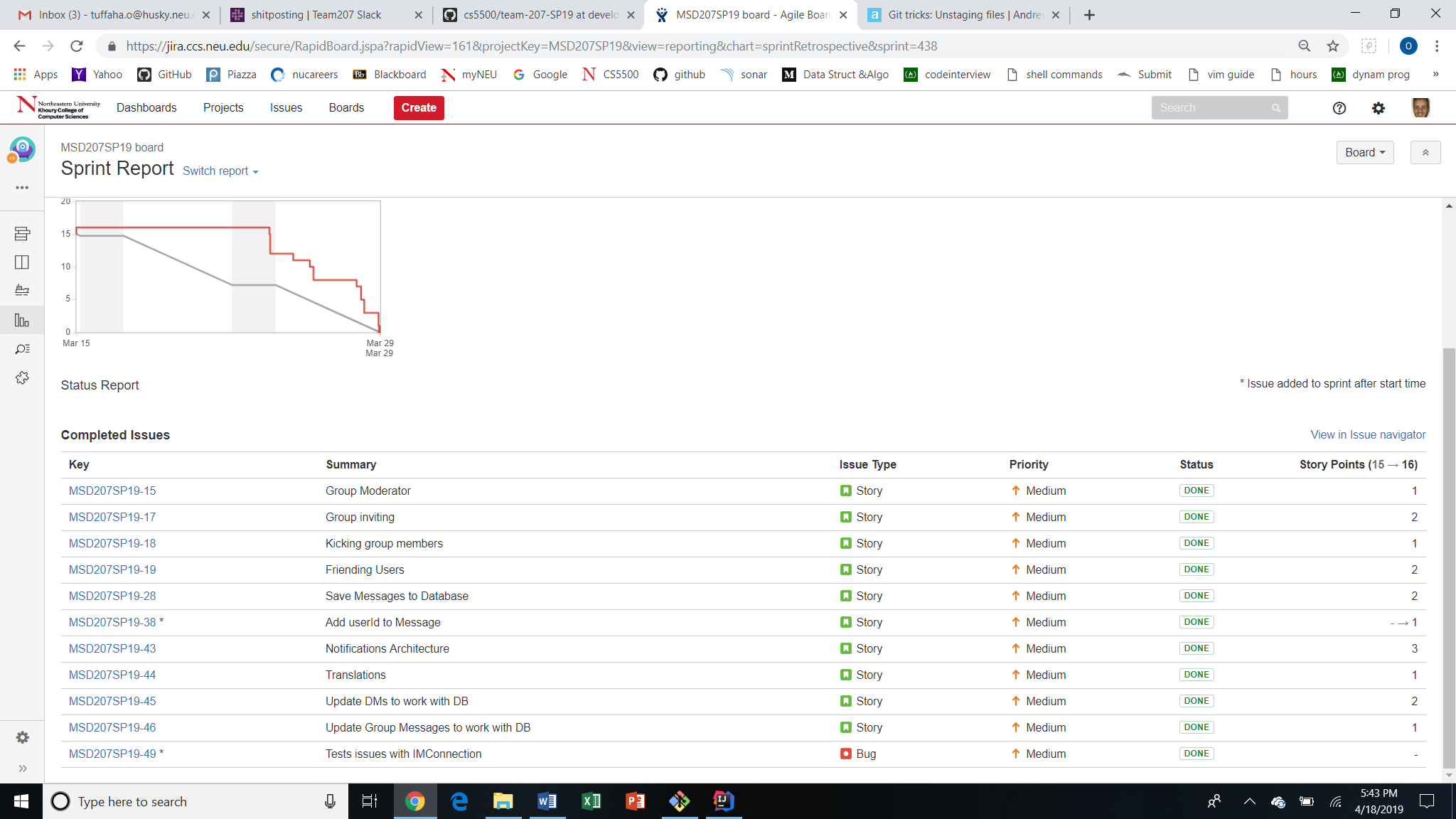


Figure 3: Displays the backlog statistics for Sprint 3.

Shown above are the tickets for implementing the core architecture of group inviting, notifications, friend requests, and other special commands. Although the burndown chart shows slow progress throughout this sprint, it is important to note that burndown charts do not keep track of ticket progress, they only keep track of ticket completion. Therefore, it is very normal for the chart to take a steep dip at the end of the sprint. All the sprint backlog tickets were completed.

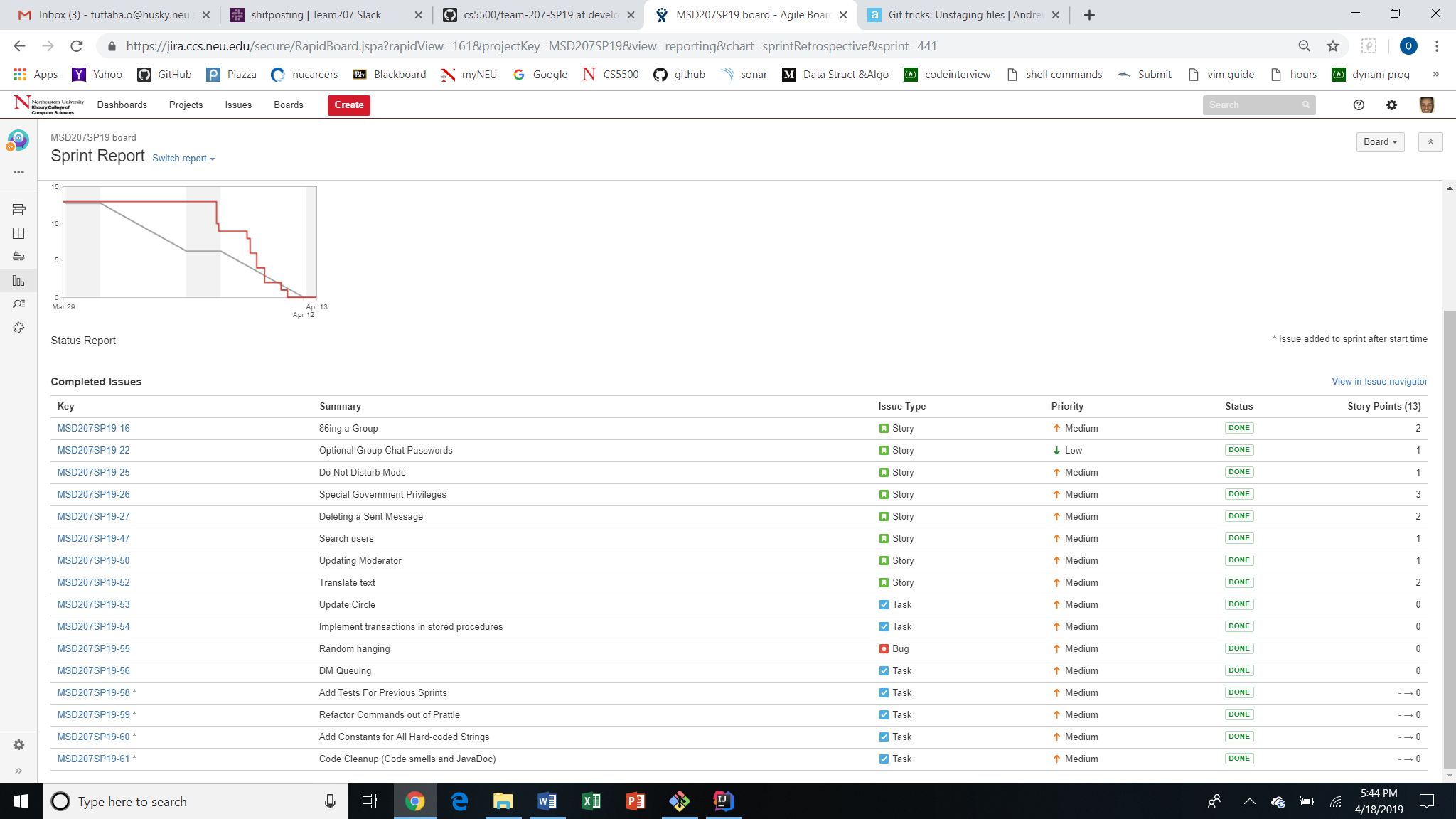


Figure 4: Displays the backlog statistics for Sprint 4.

Shown above are the tickets that wrapped up the project in Sprint 4. Notice how all of the tickets were completed, resulting in an empty product backlog.

**Quality**

In terms of quality, Team 207’s results were very sufficient. According to SonarQube’s metrics, Slaq maintained an 85.1% code coverage with over 400 unit tests. There were only 41 code smells and only 1.7% code duplications. However, many of these code smells and code duplications were uncontrollable due to the given code or due automatic smells when querying a database. Similarly, a lot of exceptions and branches were not realistically testable from Dr. Jump’s code. Please see Figure 5 below for exact metrics in terms of quality.

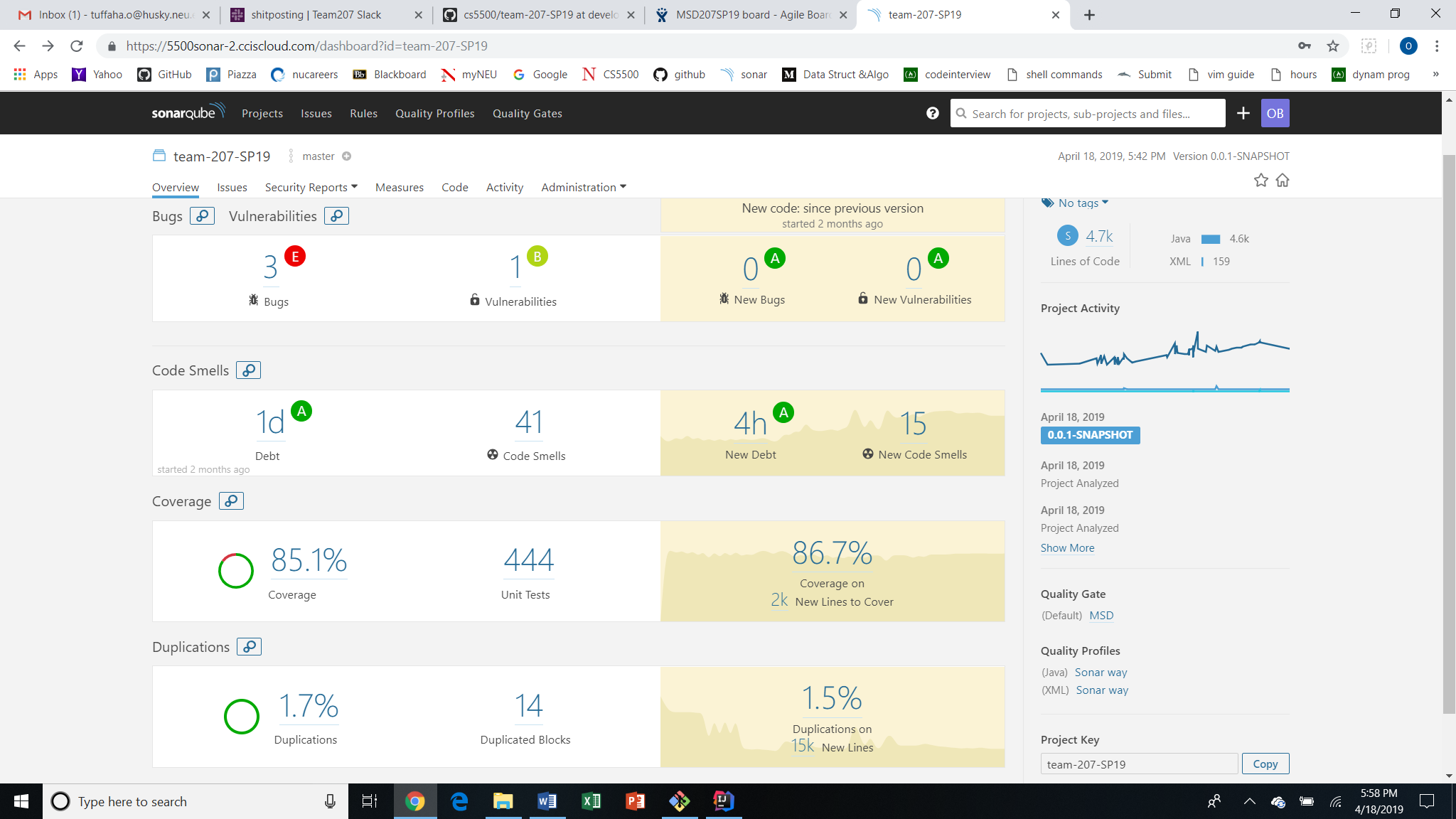


Figure 5: Shown above are quality metrics for Team 207’s Slaq software.

**Developmental Process**

Team 207 possessed a very structured and organized developmental process to help ensure success throughout the project.

**Sprint Planning**

After the end of every sprint demo, the team would immediately meet for the following sprint’s planning. This included the creation of new issues, the story pointing of new issues, and pulling tickets from the product backlog into the sprint backlog. This way, the sprint backlog was finalized before the start of the subsequent sprint and every team member was aware what was expected of them. Each team member was informally assigned different tickets to divide up the work evenly based on the team members’ different skill sets and time budget.

In terms of story pointing during Sprint Planning, the team used PlanITPoker as a software to anonymously point stories.

**Meetings**

The team had routine meetings that occurred twice a week. These meetings were every Tuesday during our TA’s office hours in case we needed help. Even if we didn’t need help, we would meet at that time and discuss progress and see if a team member would need any guidance. We also met every Thursday, which was useful every two weeks before the Sprint Demo. There, we would quality assure the tickets that were considered done in unison with other tickets that were considered done. We essentially performed mini integration testing on Thursdays. We also met after each Sprint Demo for a Sprint Planning. The team also organized plenty of random meetings as well where every team member always attended. These random meetings were in addition to the routine meetings and were organized for important discussions.

**Daily Standups**

The team participated in daily standup meetings on Slack with a channel specifically dedicated to the standups. There, the team updated one another with the work that was completed in the previous 24 hours and the work that each team member planned to complete within the next 24 hours. Any blockers were also brought up during these standup meetings. These were especially useful in keeping the team honest to their work and always aware of what they were expected to complete.

**What Worked Well**

**What Didn’t Work Well**

What might be done to address issues