# Features

### Implemented Features

* + - Blacklist and blacklist database
    - Able to add and remove users from the blacklist
    - Able to display who is currently on the blacklist
    - Able to kick inactive users from the inactive list
    - Help command that lists out the list of commands
    - All commands are only accessible to users who have admin permissions
    - Able to detect when a user is inactive and add them to the inactivity list
    - Able to set the timer for the purge
    - Able to check when the next purge is

### Missing Features

* + - Log whoever conducts the purge as well as the users who have been kicked
    - The bot should detect voice chat activity and reactions
    - The bot should kick users automatically based on the timer set by the admins
    - Creating a command that will change the time that considers who is inactive

# Know Issues

### The Discord bot will only track inactive users if the bot is online

* 1. If a user is in the inactive list and then added to the blacklist they will still be kicked if the purge is completed.

# Project Flyer



# Project Demo

* 1. <https://youtu.be/8CeXSI-xgm0>

# Team Introduction

* 1. Clients
     + David Rivera Rocha: Software Engineer



* + - Mike Deiters: Software Engineer



* 1. Bit Bot Boot Team



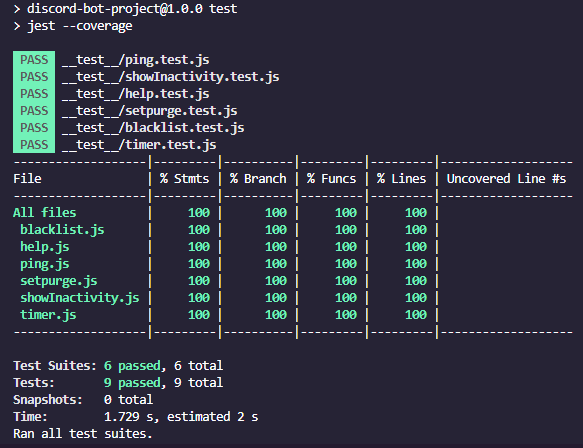
* + - Gabriel Kerven
      1. Lead Programmer
      2. Client Liasion
    - Alexis Pardo
      1. UI/UX Design
      2. Testing lead
    - Ravjot Singh
      1. Data Modeler
      2. Team Manager
    - Chidiebube Okebalama
      1. Lead Programmer
      2. Documentation Lead

# Project Abstract

* 1. Our team chose to build a Discord bot that detects inactive users and purges them from a Discord server. The bot contains many features including a database connection that holds "blacklisted" users who are immune to the purge (kicking users), a help command to show all the different commands and a purge command that lets Admin confirm or cancel. The bot is very user-friendly and contains documentation to help users set up/use the bot.

# Testing

* 1. Used Jest test cases to test all commands

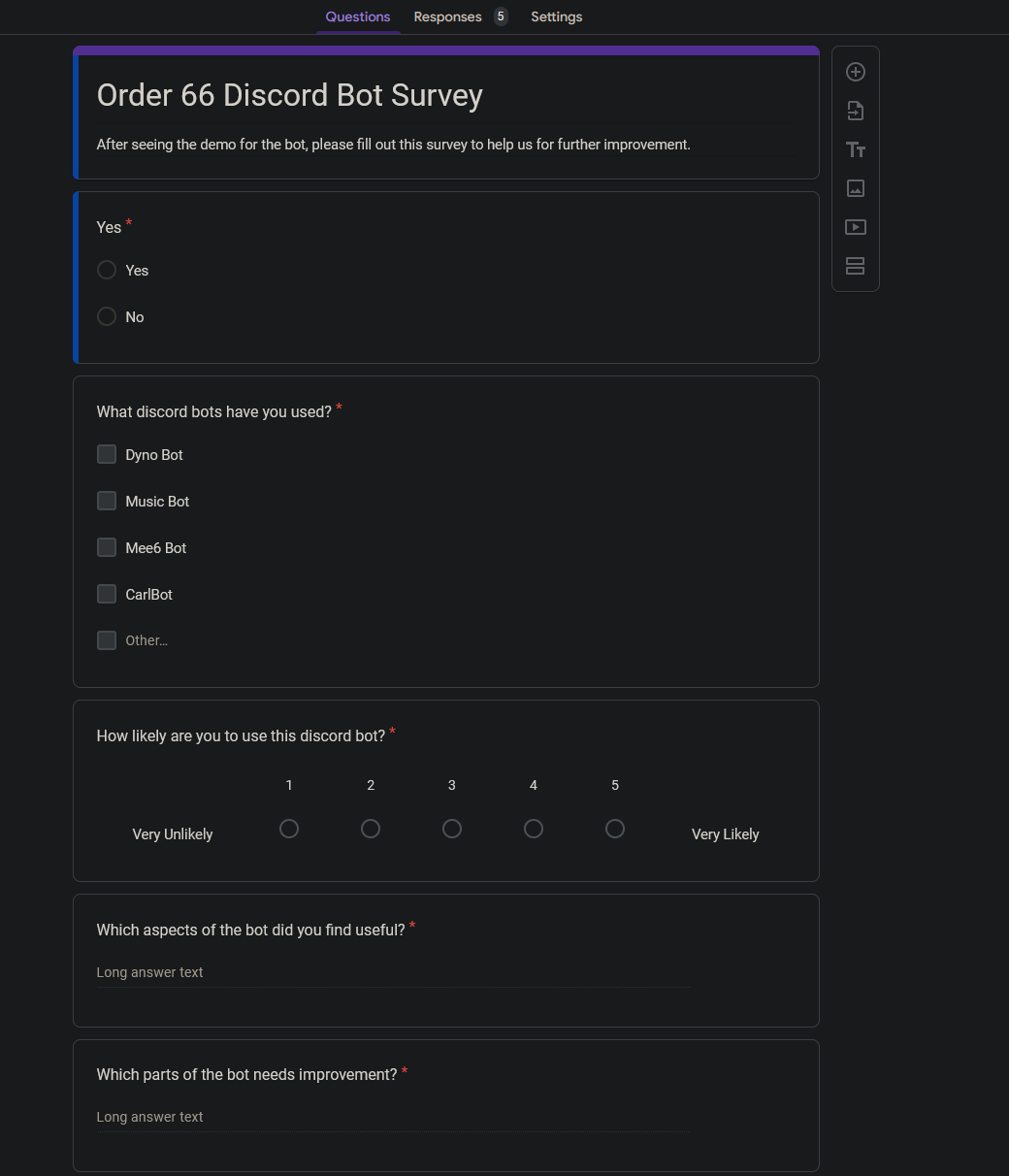


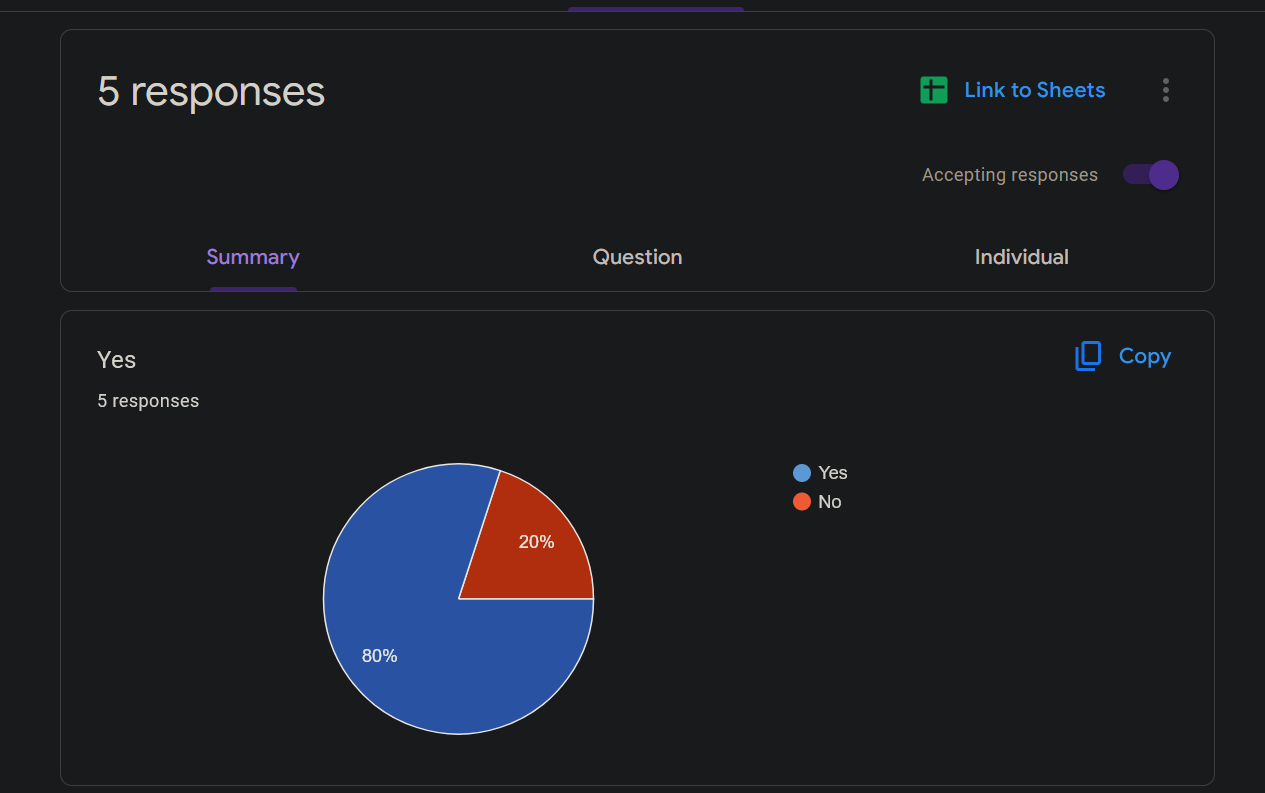
# Documentation

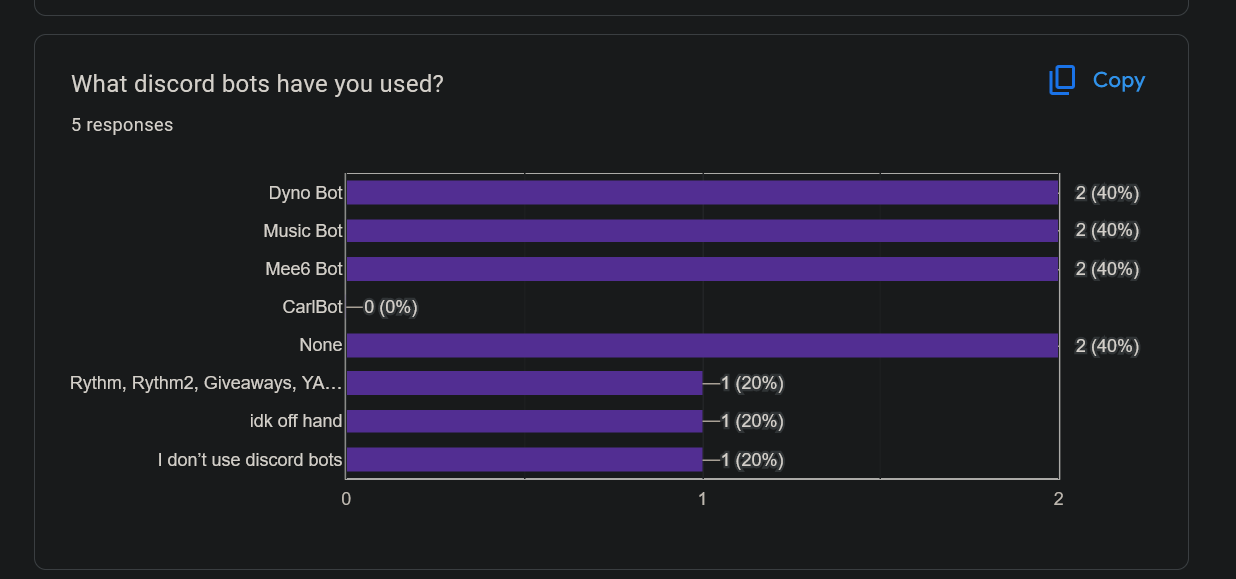
1. Install Node.js and its dependency Chocolatey
2. Initialize the bots project using the command “npm init -y”
3. Install nodemon using the terminal, using the command “npm install dotenv”
4. Install discord.js using the command “npm i discord.js”
5. Invite bot to discord server
6. Create a .env file to hold the bot TOKEN, GUILD\_ID (Discord server id), CLIENT\_ID (Application id), databaseToken (MongoDB)
7. Run the bot constantly
8. Test run commands to create databases and ensure proper function

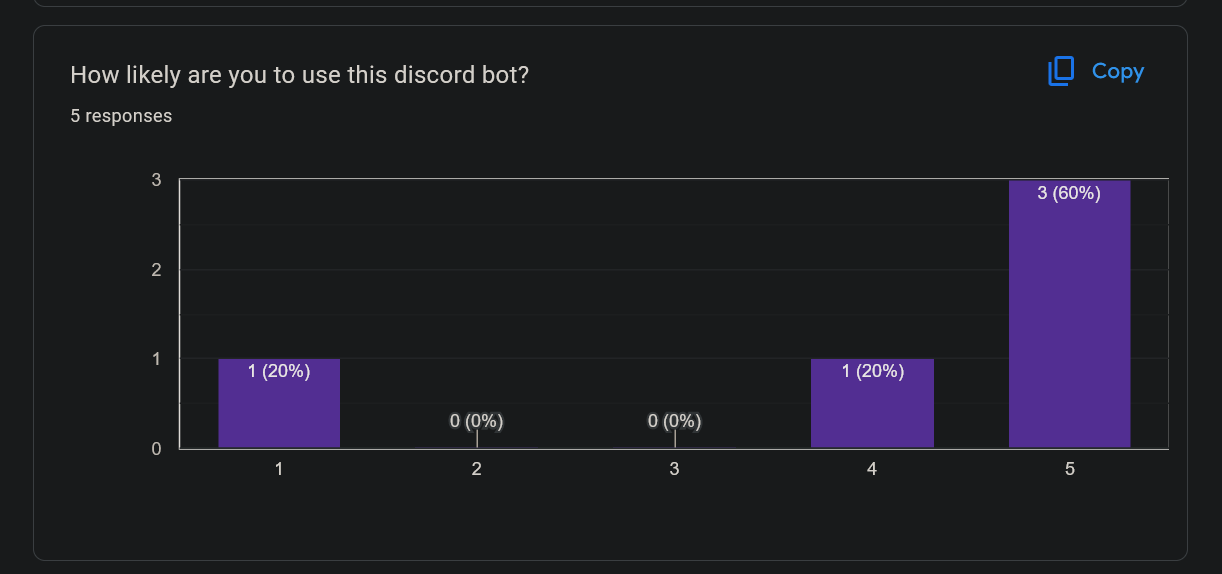
# Usability Testing

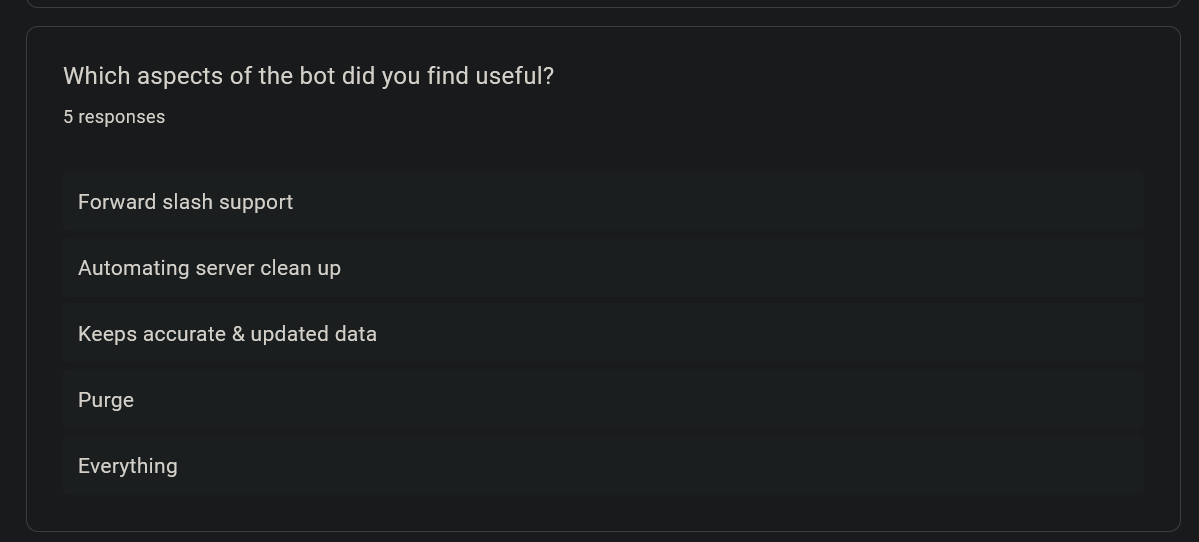
* 1. Allowed people from create to test the bot

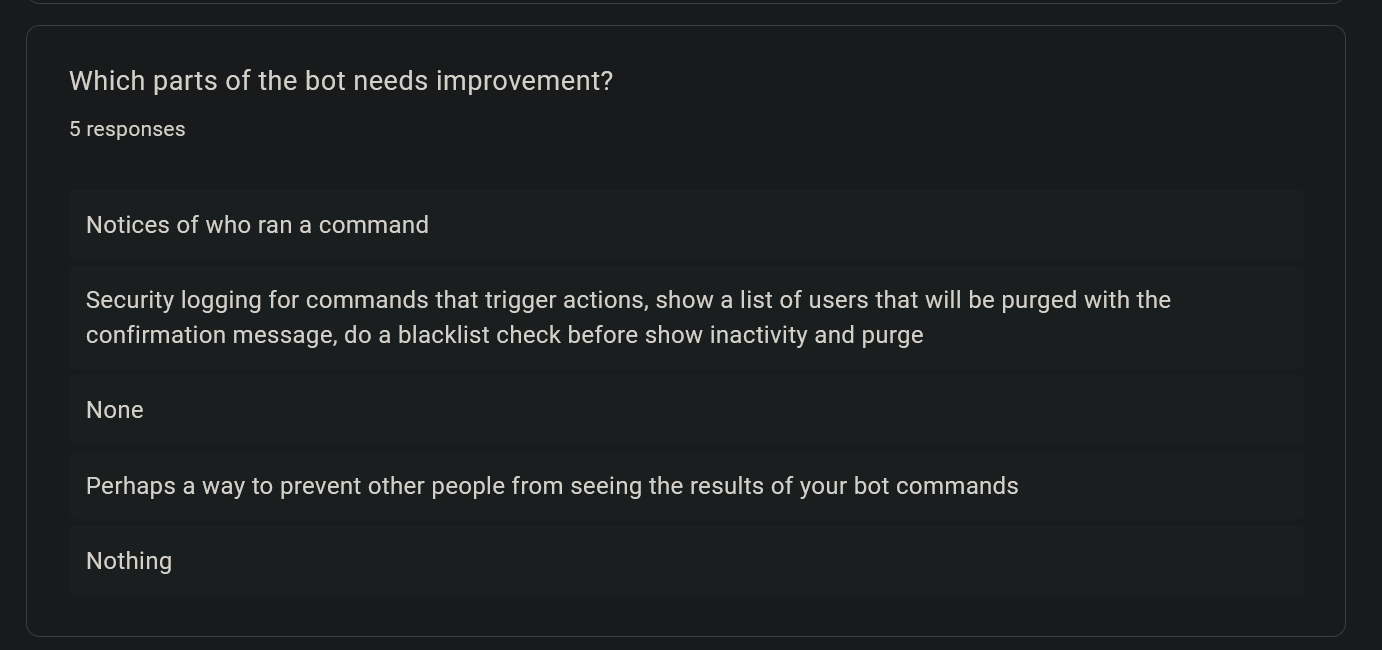












# Where to find documentation

* 1. Installation information should be on the readme
  2. Documentation should be in the DiscordBot/documents/docs-Fall2023/

# Software Usage and Intellectual Property Terms

### Software Usage

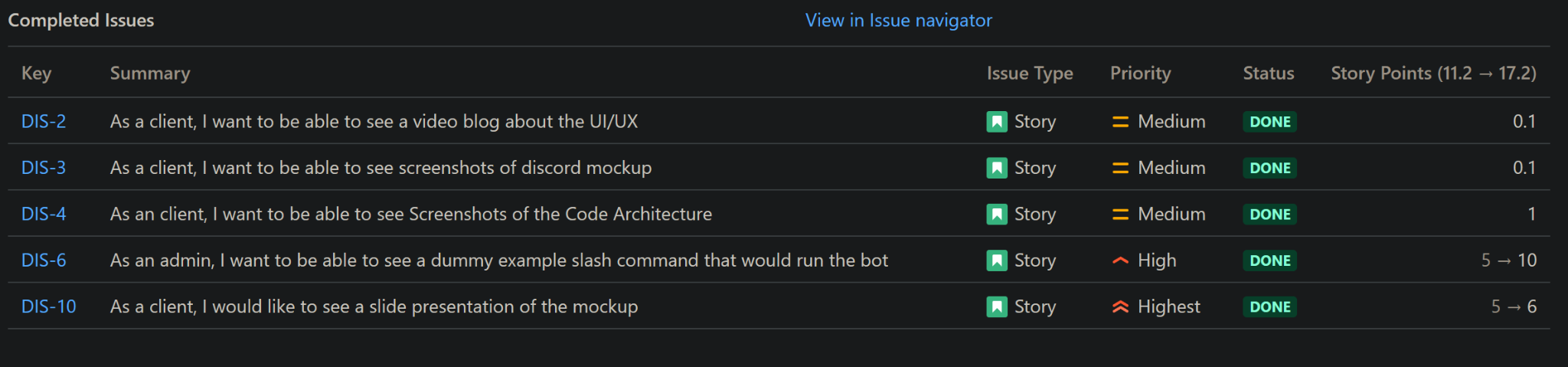
* + - Node.js
    - Discord.js
    - MongoDB
    - Mongoose
    - Jest
    - Javascript

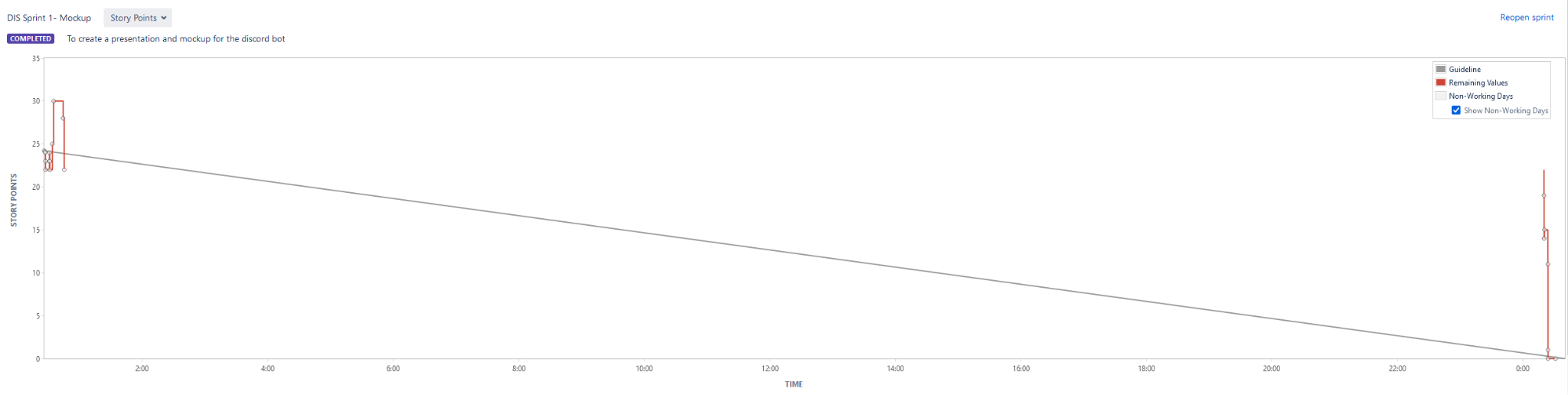
### Intellectual Property Terms

# Sprint Reports

### Sprint 1

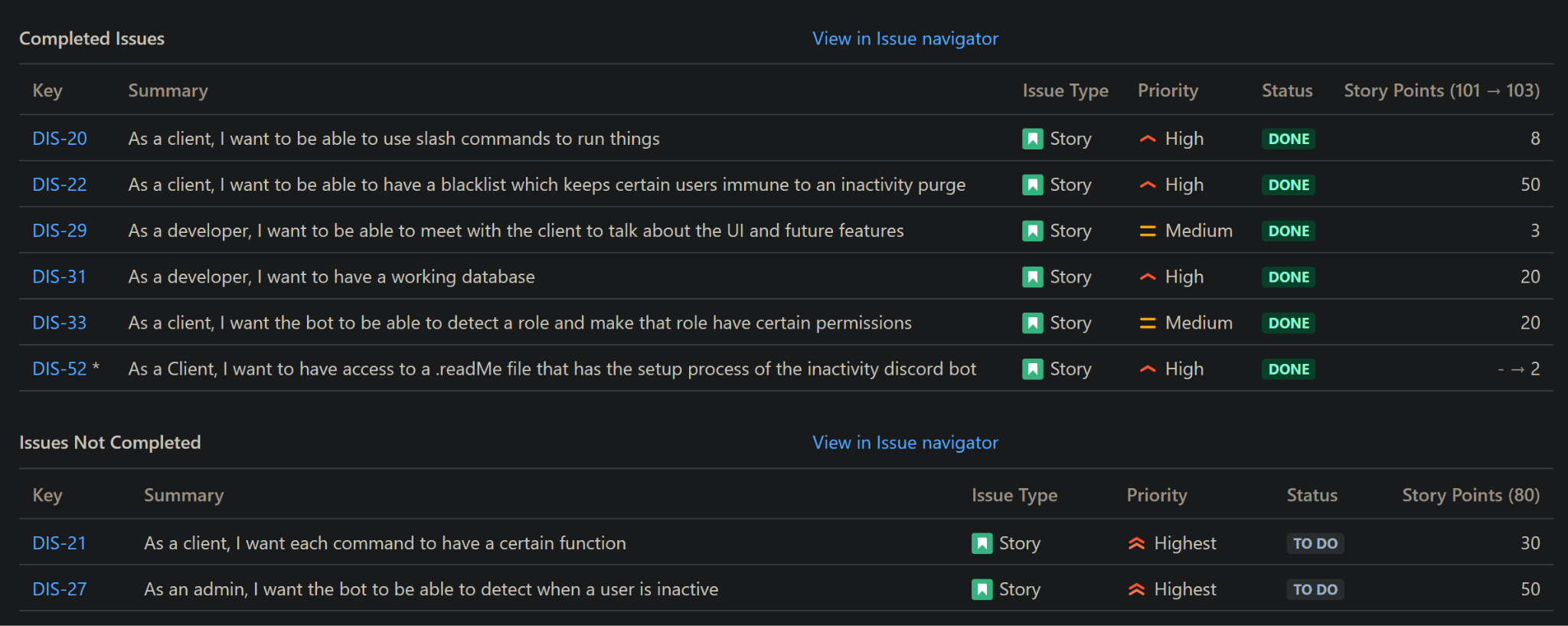
This sprint was our first and as such we found it difficult to adjust to the tool Jira while learning our new coding language JavaScript. Due to our inexperience, we found ourselves adding tasks and deleting tasks as well as starting and then stopping and restarting the sprint. When we did this sprint we learned that starting a sprint should not be stopped or paused so that the sprint is understandable.





### Sprint 2

Our second sprint was met with more positive results because we were more familiar with the tool Jira and how to properly create user stories and tasks. Still, we messed up when we added a short task in the middle of the sprint, and this altered the otherwise okay-looking Jira Sprint results. It was during this sprint that we learned how to properly break down user stories into appropriately sized tasks.

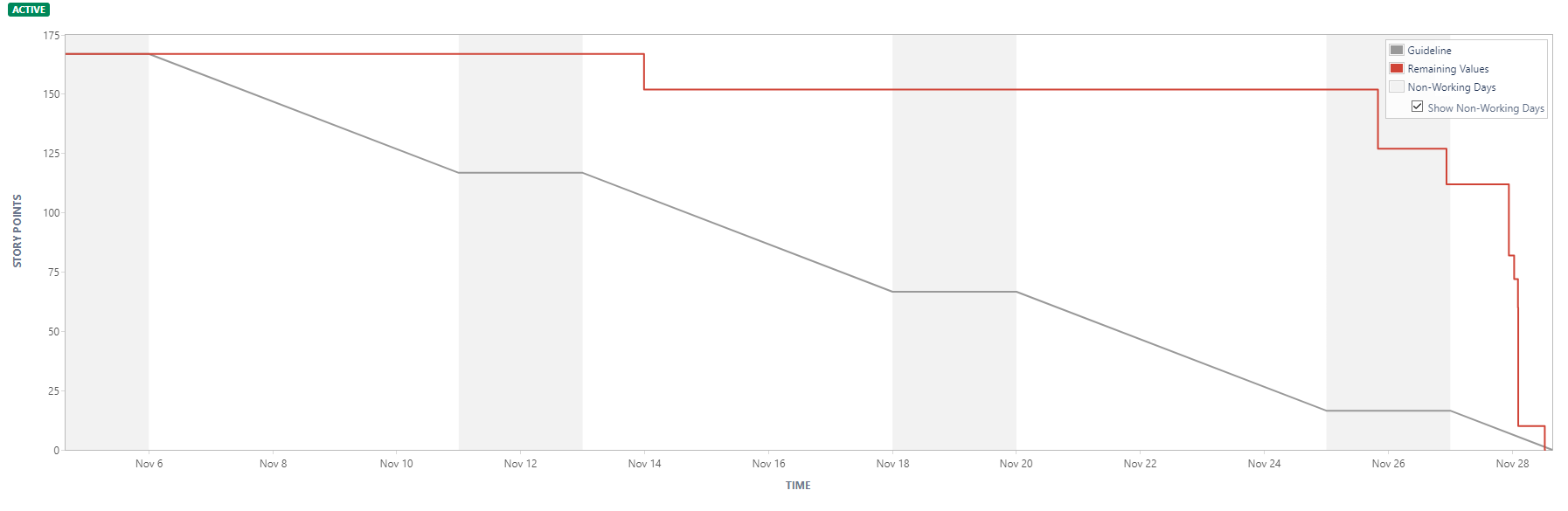




### Sprint 3

On Sprint Three we understood for the most part the ins and outs of how to use almost all of our tools including Jira. During this sprint however, we found ourselves forgetting to enter what tasks and user stories had actually been completed and therefore our Jira results show that we held off on the bulk of our coding till the end when in actuality we had completed a lot of what was still incomplete on Jira. This combined with our previous experiences in Jira taught our group to be more proactive when it came to entering information for Jira, and also for those pieces of code that we did procrastinate on these charts taught us that it’s always better to get ahead than to rush towards the end.





So in the future, I would make more of an effort to reach out to teammates and organize more opportunities for our group to work on the bot. This would at least provide us with better burndown charts, and we would also enter Jira as soon as our deliverables get done.