Final Report

1. **Features**

   A. **Working Features**

- ***Dashboard:*** The bot now has an authenticated login for admins only web-based UI dashboard that, in the future, will be the only point of interaction with the bot.
  - I. Login page which authenticates whether a user has an admin role to access the dashboard.
  - II. Pages for each command that are accessible from the hamburger in the navbar under 'Commands' which open the sidebar on the left of the page.
    - User Activity - displays an overview of user activity within the application, including their latest messages, reactions, and voice interactions in the ford of an interactive table.
    - Inactivity - displays an overview of inactive users within the application in the form of an interactive table.
    - Purge - displays an overview of inactive users and a history of whom committed a purge.
    - Blacklist - displays an overview of all blacklisted users within the application. Allows the user to select users from the table to remove them from the blacklist.
    - Roles & Timers - displays two tables: User Roles and Role Timer. User Roles provides an overview of each individual users information, including their username, and user id, along with their role names (multiple roles if applicable) in the form of an interactive table. Role Timer an overview of the roles and timer for users within the application.
    - FAQ page displays an accordion of question and commands which are parsed from JSON files in the JSON folder within the frontend JSON folder.
  - III. 'Logout' button.
  - IV. 'Account' button that currently leads to a skeleton page.
  - V. 'Home' button to take the user back to the welcome landing page.
- **Discord Server** - Slash Commands Pertains to the commands entered within the server.
  - I. <u>/help:</u> Lists out all the different configuration commands for the bot.
  - II. <u>/blacklist show:</u> Shows the current blacklisted user/roles.
  - III. <u>/blacklist add (user/role):</u> Lets you add a specific user/role to a blacklist which makes them bypass the purges.
  - IV. <u>/blacklist remove (user/role):</u> Lets you remove a specific user/role from the blacklist.

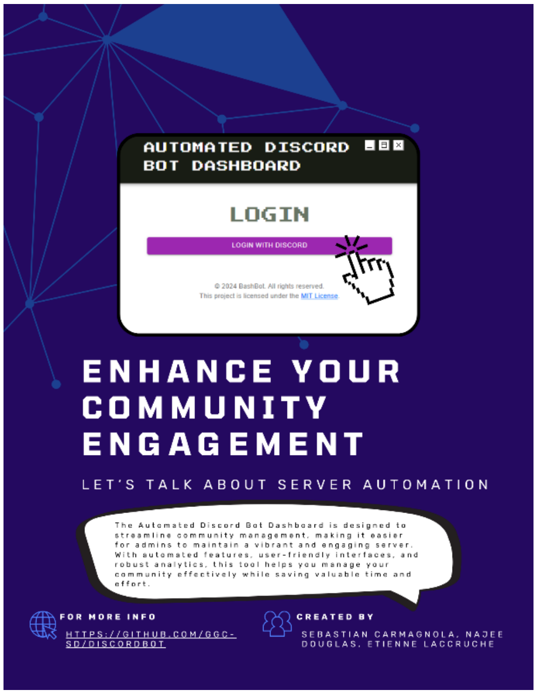
    V. /purge: Starts a manual purge which will removed inactive users from server and send confirmation to specified channel.
    VI. /roletimer (role name) (amount of time in days): Sets the grace period for a certain role.
    VII. /setpurge (time in days): Sets the specified automated purge window (in days).
    VIII. /timer (role) (time): Sets a time window (in days) for a role before considering them inactive. Also initiates the role-based auto-purging when the bot starts.
    IX. /show inactivity: Shows members who are considered 'inactive' that are eligible to be purged.

- **Missing Features**
    I. The end-product needs to be a 24/7 streamlined process where the bot does not need to be manually activated to run. This should be done when the project is complete.
    II. Automatically kick inactive users.
    III. Fix role-based inactivity timing
    IV. Integrate buttons, data grid, and/or form for updating Role & Timer section on role & timer webpage.

2. Known Issues
    - Account Details page should have the Discord settings option to manage the bot.
    - If you use the /purge command from the Discord chat it will work, but the bot will crash right after. Needs to be debugged.
    - Consider removing /setpurge since it is not being used and the dashboard will include the button functionality that this would have been used for.
    - The database does not fully forget users who are kicked from the blacklisters and inactiveusers collections.
        - ▪ Inactivity Page
            - It is currently not removing the inactivity detail of users who get kicked.
        - ▪ BlacklistPage
            - It should not show users that have previously been kicked from the server, only current users that are deemed active in the server.
            - When toggling the Add Users button, only current users in the server AND users not already in the blacklist should appear.
        - ▪ PurgePage
            - Currently when a purge is enacted the bot attempts to clear the inactivity list, however this only happens for the list within the purge function and not for the inactivity DB. So, when the page is

        refreshed the same names are populated in the inactivity list, even for people that were just purged.
        - There is no refresh button on the page, so you need to reload and log back in to get to this page again.
        - The inactivity preview has sections for last day active and days inactive since that doesn't have any data going to them.
        - Purge history doesn't show list of users purged, only the number of users.
- The time-frame of the automatic purge function within timer is hardcoded. There is a command to change said time-frame but it currently doesn't dynamically update the time. The timeframe as of now needs to be hard coded and the bot must be reset.
    - Automatic purges are not recorded within purge history. Only purges that were manually used through the /purge command are recorded.

3. Project Flyer


AUTOMATED DISCORD BOT DASHBOARD
LOGIN
LOGIN WITH DISCORD
ENHANCE YOUR COMMUNITY ENGAGEMENT
LET'S TALK ABOUT SERVER AUTOMATION
FOR MORE INFO
CREATED BY

© 2024 BashBot. All rights reserved.
This project is licensed under the MIT License.

The Automated Discord Bot Dashboard is designed to streamline community management, making it easier for admins to maintain a vibrant and engaging server. With automated features, user-friendly interfaces, and robust analytics, this tool helps you manage your community effectively while saving valuable time and effort.

HTTPS://GITHUB.COM/GGC-SD/DISCORDBOT

SEBASTIAN CARMAGNOLA, NAJEE DOUGLAS, ETIENNE LACCRUCHE

4. Project Demo - https://youtu.be/qnSWJWxF50M?si=Gzr8XobdUlc2RazH

5. Team Introduction

    - Clients

Figure 1 David Rivera Rocha: Software Engineer

Figure 2 Mike Deiters: Software Engineer

- Team Neck Beards

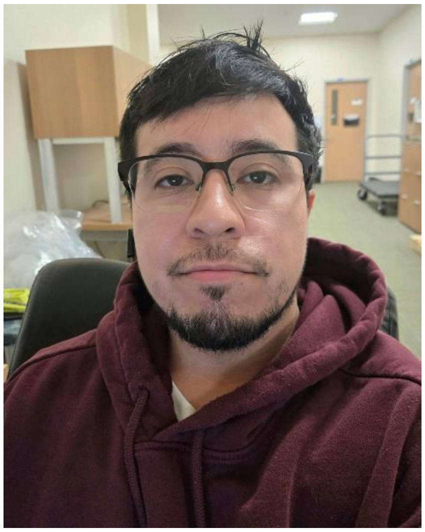


Figure 3 Sebastian Lian Carmagnola - Data Modeling Lead and Testing Lead

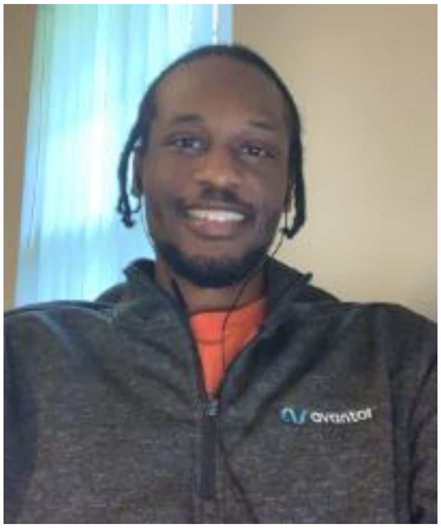


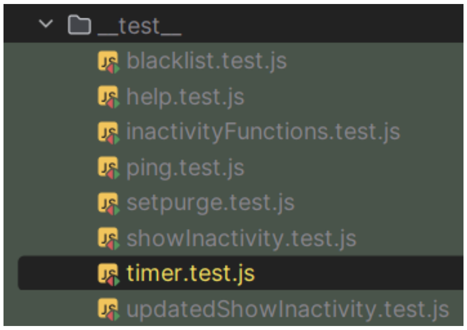
Figure 4 Najee Douglas UI/UX Lead & Documentation Lead

Figure 5 Etienne Laccruche – Programming Lead & Client Liason

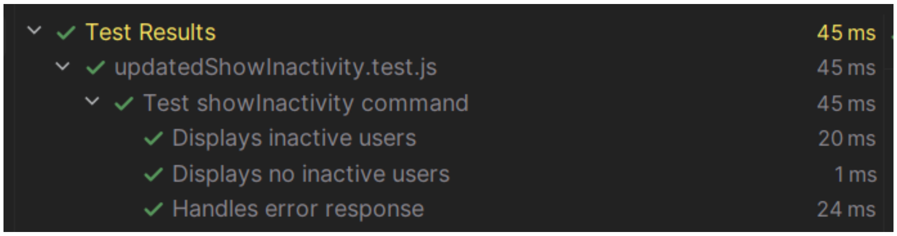
6. Project Abstract

   Discord is a communication platform designed for creating communities, where users can interact through text, voice, and video channels, while Discord bots are automated programs that can perform tasks such as moderating conversations, managing users, or providing entertainment within these communities. Our team has been tasked with creating a discord bot that can moderate servers. It can track activity across multiple channels and can purge users who have been deemed inactive for too long in order to keep the server lean. The main goal is to have this process done in an automated fashion periodically, and for the admins to be able to log into a dashboard to alter settings as needed.

7. Testing
   - Used Jest test cases to test all commands



```
⌄ __test__
    blacklist.test.js
    help.test.js
    inactivityFunctions.test.js
    ping.test.js
    setpurge.test.js
    showInactivity.test.js
    timer.test.js
    updatedShowInactivity.test.js
```



```
⌄ ✓ Test Results                                   45 ms
  ⌄ ✓ updatedShowInactivity.test.js                45 ms
    ⌄ ✓ Test showInactivity command                45 ms
        ✓ Displays inactive users                  20 ms
        ✓ Displays no inactive users                1 ms
        ✓ Handles error response                   24 ms
```

8. Documentation

Installation Steps

- Using your Operating Systems Terminal set the ExecutionPolicy. Use the command: Set-ExecutionPolicy RemoteSigned -Scope CurrentUser.
- The RemoteSigned is also a safe PowerShell Execution policy to set in an enterprise environment. This policy dictates that any script that was not created on the system that the script is running on, should be signed. Therefore, this will allow you to write your own script and execute it. You would not be able to run a script that was downloaded from the Internet or got from a friend who wrote it under his account or on a different computer. PowerShell differentiates between script you wrote on that computer, and a script that came from elsewhere using the file metadata.
- (The scope makes the RemoteSigned only effect a single user instead of allowing all the users to become effected by the terminal command. This is a security precaution.)
- Install Node.js. and its dependency Chocolatey.
- Install nodemon using the terminal and the command npm install -g nodemon (nodemon installation allows the script to run continuously in the background while also autoupdating anytime script is changed in the bot project and saved allowing all changes to be quickly deployed for the bot).
- Create a Discord bot on the Discord bot.
- Add the bot's token (Discord developer Portal-> application-> "specific app"-> bot -> Reset Token) to config.json and adjuste the configuration to indicate where the script is on your local machine.
- Run the bot by using nodemon or nodemon index.js or any other command you prefer.

- Add database token (Go to your MongoDB account. Under 'Database' select Clusters. Select the corresponding DB, Select the Connect button. Under 'Connect to your application', select Drivers. Follow the instructions to complete the setup and get the token for the database.)
- Add guild id, in the Discord app, right-click on the Server Icon. Select copy Server ID.

Updates 2024

- cd into src active directing for all installations from here.
- - Install dotenv on using the terminal and the command ` npm install dotenv` (This causes allows your bot project a place for the bots token to be stored later in the project and then be placed into a gitignore file in order to keep your token and other confidential project information safe if the bot is shared on GitHub.)
- Install discord.js using the terminal and the command ` npm i discord.js`.
- Install express.js using the terminal and the command ` npm i express` (Allows for a simple web server framework to handle requests received from the frontend app.)
- Install mongoose using the terminal and the command ` npm i mongoose` (Enables database integration with MongoDB so the app can read and write to the database.)
- Install cors using the terminal and the command ` npm i cors`
- Install axios using the terminal and the command ` npm i axios` (Allows front end to fetch data from the server for displaying in the UI.)
- Install the react router dom using the terminal and the command ` npm i react-router-dom` (Helps set up webpage routing on the server.)
- Install react scripts using the terminal and the command ` npm i react-scripts` --save

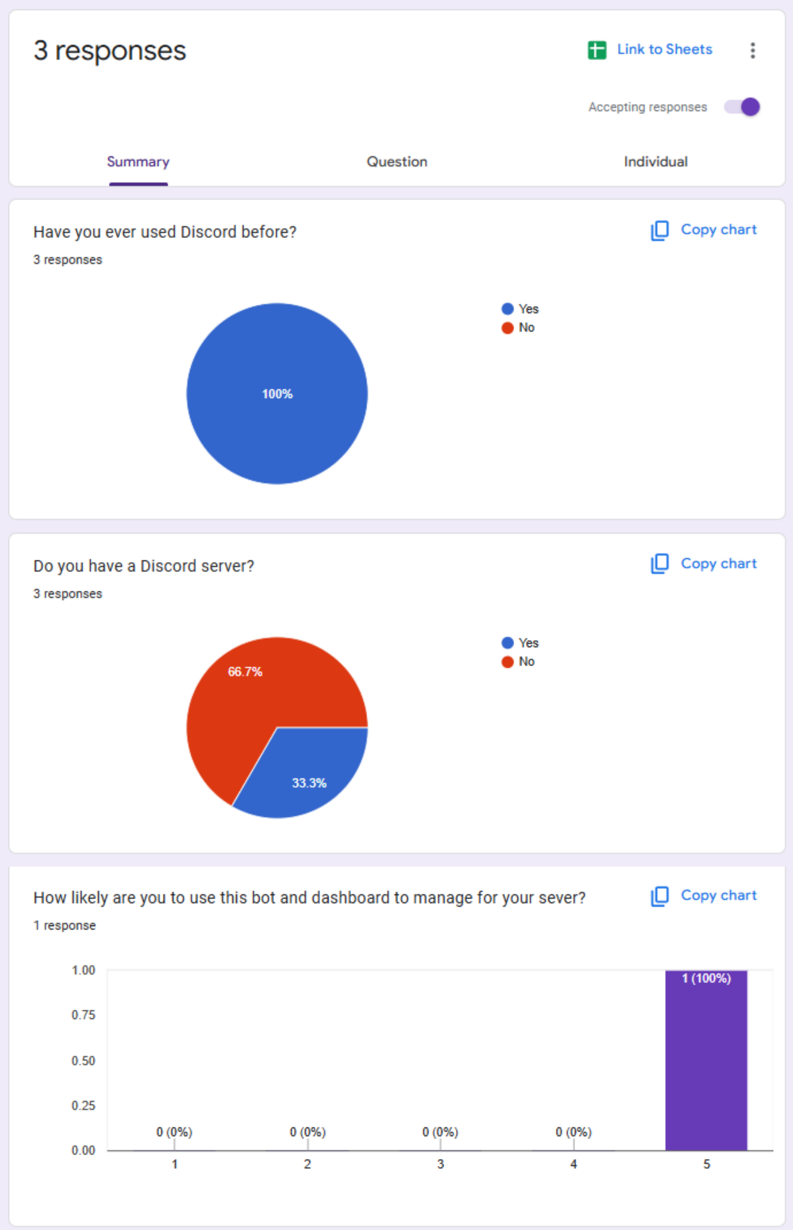
Setup Dotenv

- In the src folder, create a new file named .env which will hold the:
- TOKEN (your discord bot token)
- databaseToken (your MongoDB token)
- GUILD_ID (your server id)
- CLIENT_ID (found on the discord developer portal -> click the hamburger -> OAUth2 -> Client Information)
- CLIENT_SECRET (found on the discord developer portal -> click the hamburger -> OAUth2 -> Client Information)
- REDIRECT_URI (found on the discord developer portal -> click the hamburger -> OAUth2 -> Redirects)

How To Run

- From the src active directory, type `npm start` in the terminal and the bot, server, and react app will all start automatically. You will then see a login webpage appear in the browser automatically, presenting the admin login page for the dashboard.
- In your IDE terminal you will see logs to aid in tracing how the code runs
- In order to get past the admin login page, you must configure the .env with all the necessary fields

9. Usability Survey - https://forms.gle/QdHz8xMbSUuSWPy79



3 responses

Accepting responses

Have you ever used Discord before?

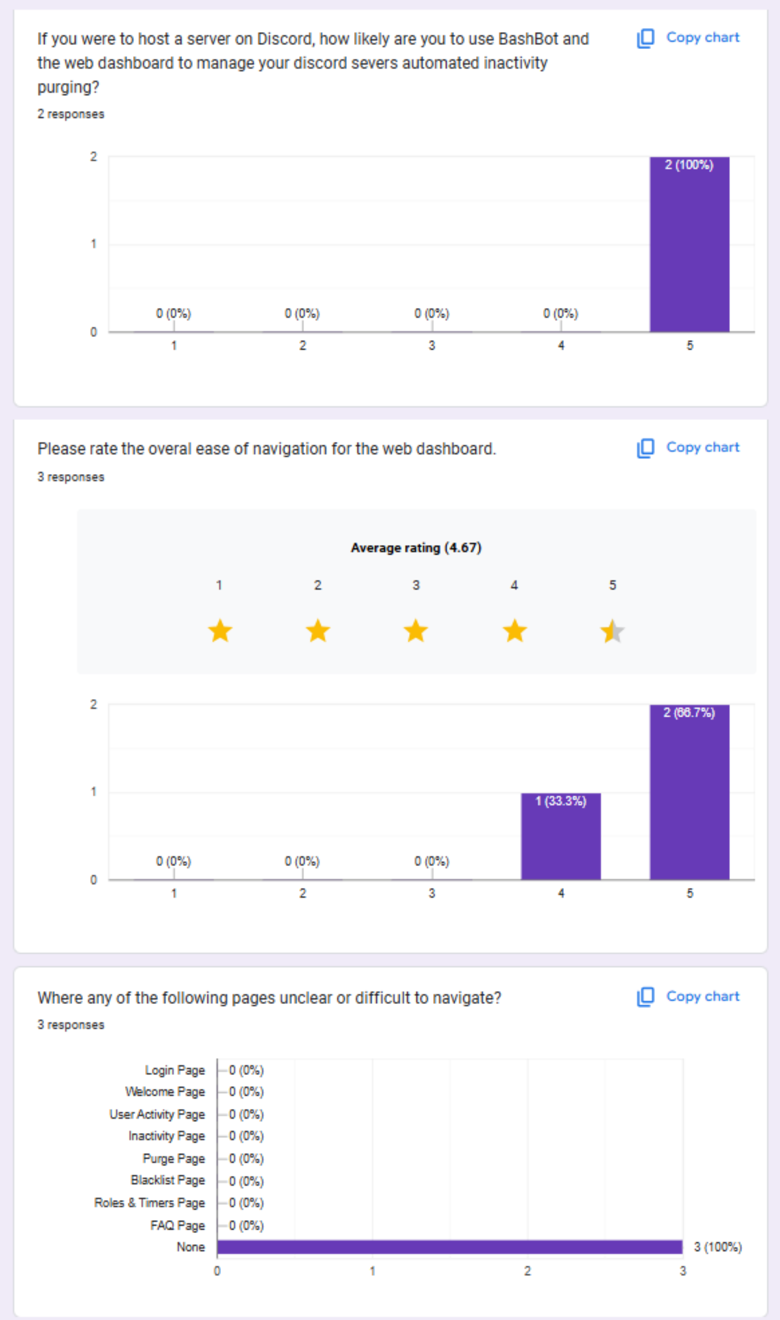
3 responses

Do you have a Discord server?

3 responses

How likely are you to use this bot and dashboard to manage for your sever?

1 response



If you were to host a server on Discord, how likely are you to use BashBot and the web dashboard to manage your discord severs automated inactivity purging?

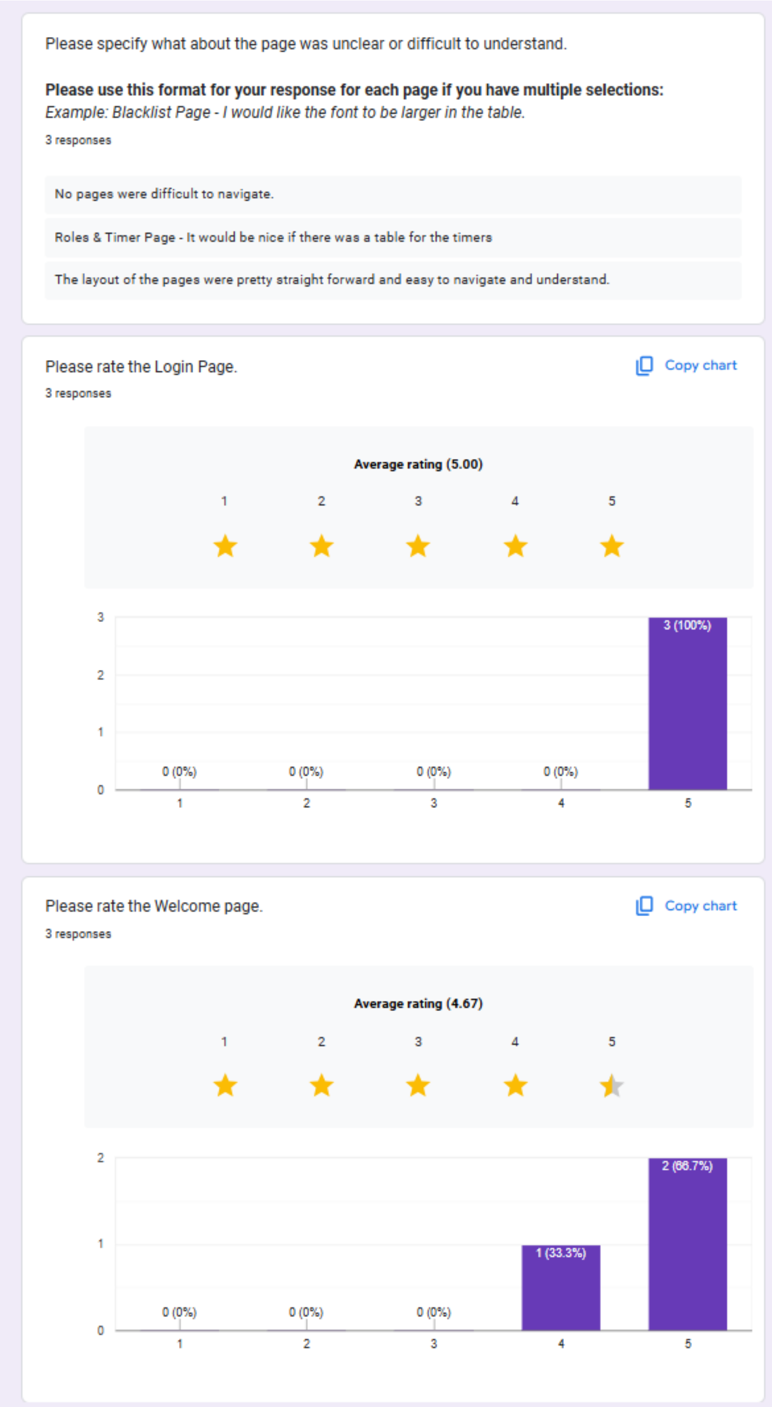
2 responses

Please rate the overal ease of navigation for the web dashboard.

3 responses

Where any of the following pages unclear or difficult to navigate?

3 responses



Please specify what about the page was unclear or difficult to understand.

Please use this format for your response for each page if you have multiple selections:
Example: Blacklist Page - I would like the font to be larger in the table.

3 responses

No pages were difficult to navigate.

Roles & Timer Page - It would be nice if there was a table for the timers

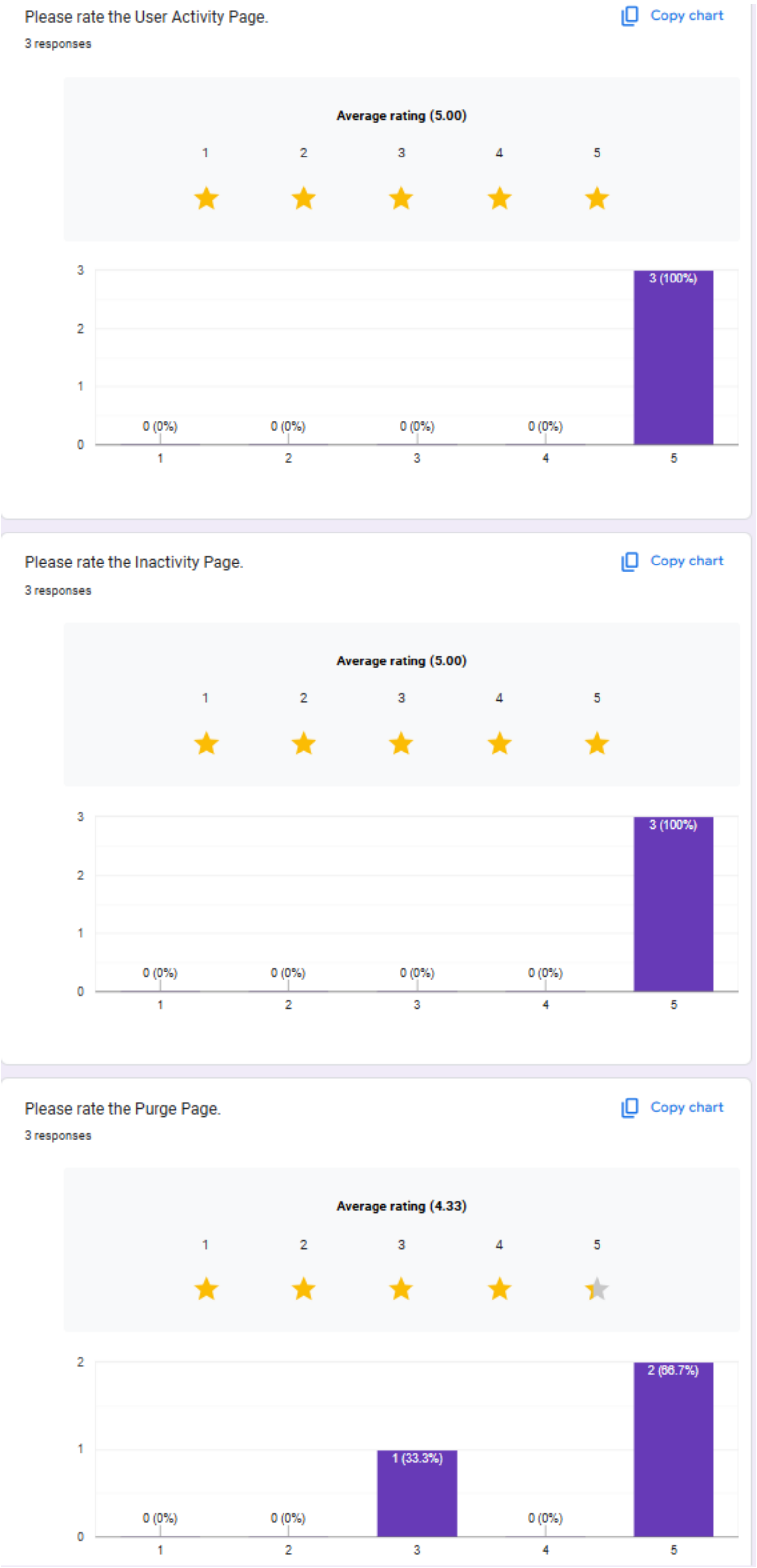
The layout of the pages were pretty straight forward and easy to navigate and understand.

Please rate the Login Page.

3 responses

Please rate the Welcome page.

3 responses



Please rate the User Activity Page.

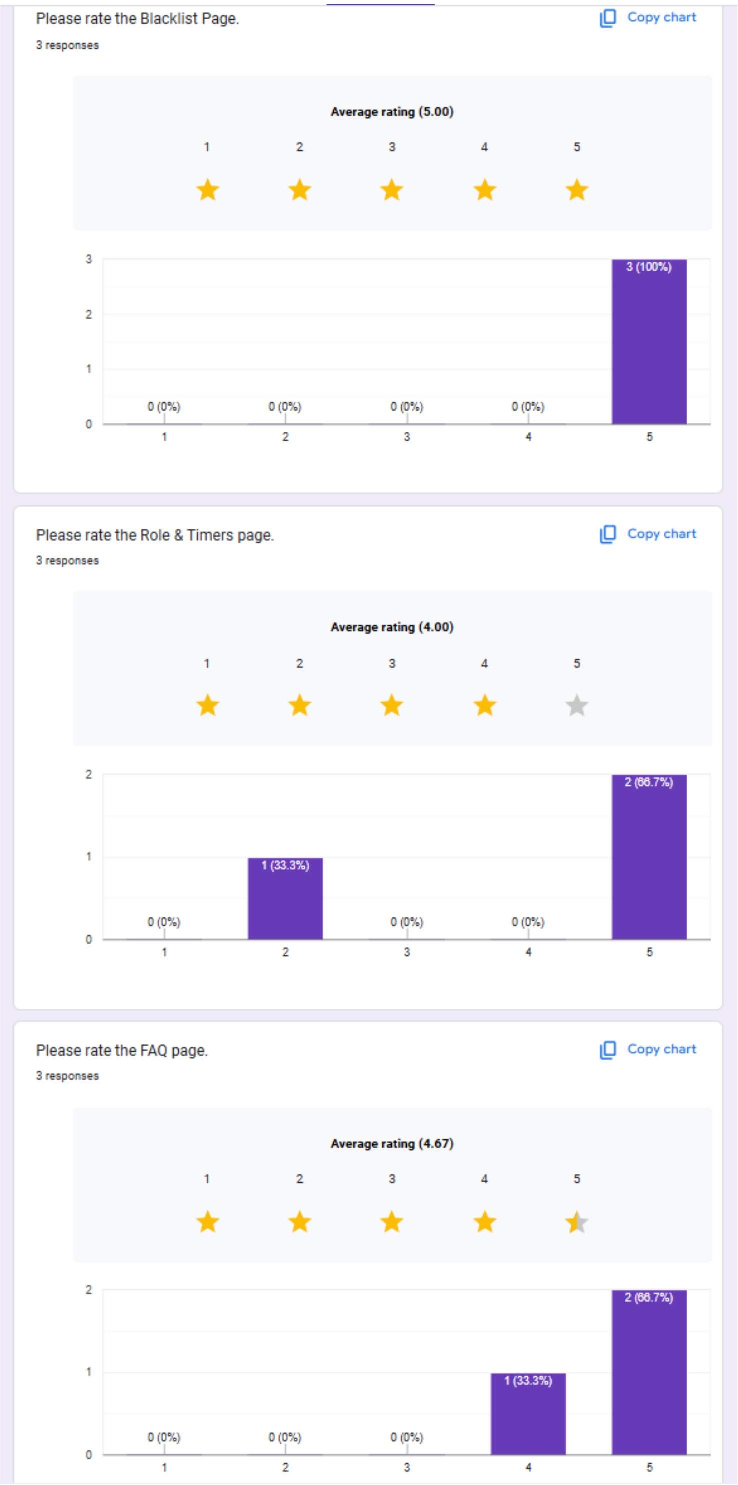
3 responses

Please rate the Inactivity Page.

3 responses

Please rate the Purge Page.

3 responses



Please rate the Blacklist Page.

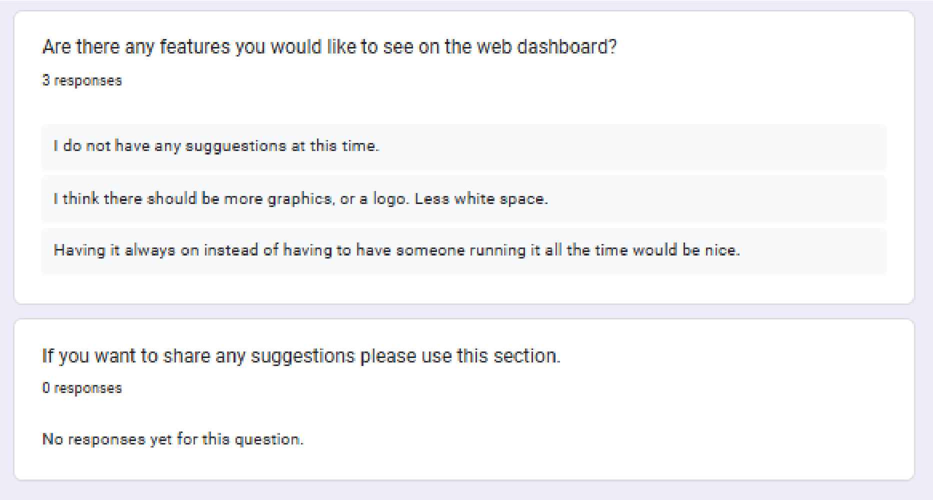
3 responses

Please rate the Role & Timers page.

3 responses

Please rate the FAQ page.

3 responses



Are there any features you would like to see on the web dashboard?

3 responses

I do not have any sugguestions at this time.

I think there should be more graphics, or a logo. Less white space.

Having it always on instead of having to have someone running it all the time would be nice.

If you want to share any suggestions please use this section.

0 responses

No responses yet for this question.

10. Where To Find Documentation

- Installation information should be on the readme
- Documentation should be in the DiscordBot/documents/docs-Fall2024/

11. Software Usage, Licensing & Intellectual Property Terms

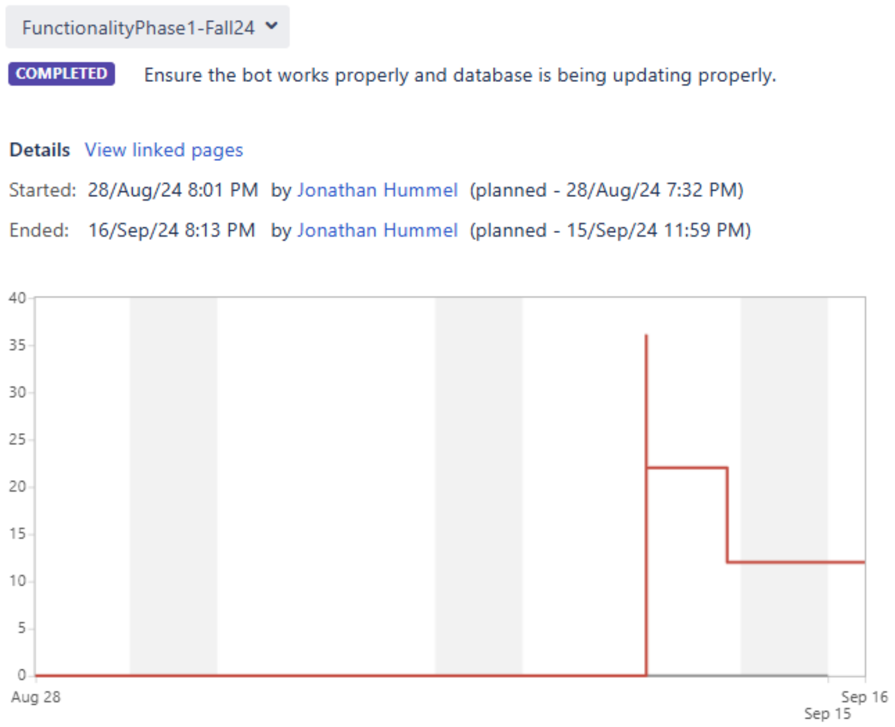
Software Used

- Discord.js
- MongoDB
- Express.js
- React
- Node.js
- Mongoose
- Jest
- JavaScript
- CORS
- Axios
- MUI
- Bootstrap

Licensing & IP Terms

- Intellectual Property Terms for "BashBot" © 2024
-

12. Velocity Charts



FunctionalityPhase1-Fall24

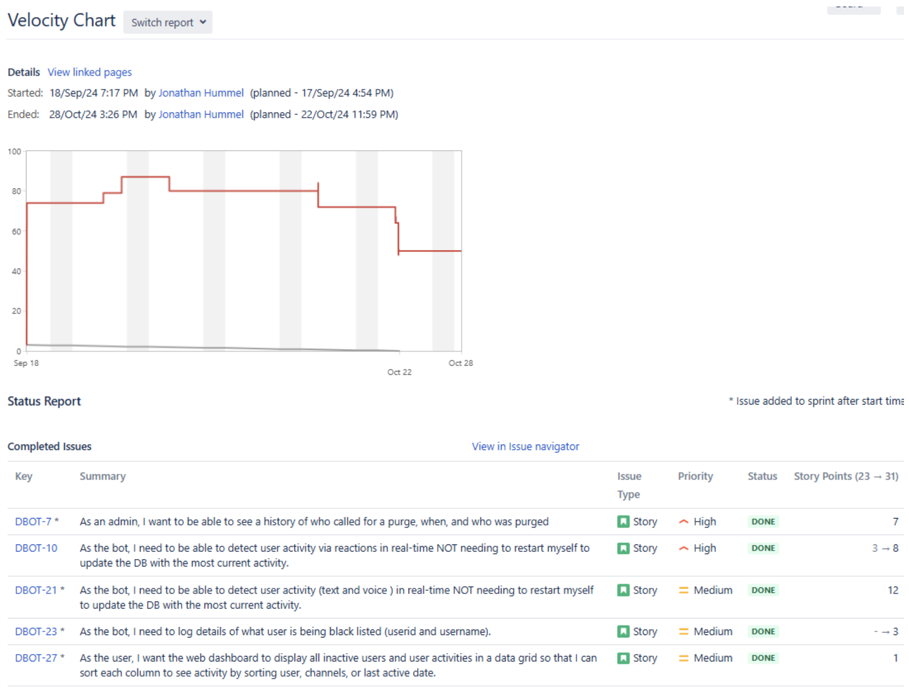
COMPLETED Ensure the bot works properly and database is being updating properly.

Details View linked pages

Started: 28/Aug/24 8:01 PM by Jonathan Hummel (planned - 28/Aug/24 7:32 PM)

Ended: 16/Sep/24 8:13 PM by Jonathan Hummel (planned - 15/Sep/24 11:59 PM)

Sprint I: During this sprint, our team faced challenges in figuring out how to log work effectively and properly break down tasks, which accounts for why the sprint's progress seems to start later in the timeline. The graph represents the cumulative effort and tasks completed over the sprint duration. The graph shows the cumulative effort or tasks completed over the sprint duration. The initial spike indicates a significant amount of progress made early in the sprint. A dip or plateau is visible midway, suggesting that progress slowed down or faced challenges during that period. The final reduction in the graph implies that some tasks or adjustments were made as the sprint concluded, potentially marking a completion or review phase.



Velocity Chart

Details View linked pages

Started: 18/Sep/24 7:17 PM by Jonathan Hummel (planned - 17/Sep/24 4:54 PM)

Ended: 28/Oct/24 3:26 PM by Jonathan Hummel (planned - 22/Oct/24 11:59 PM)

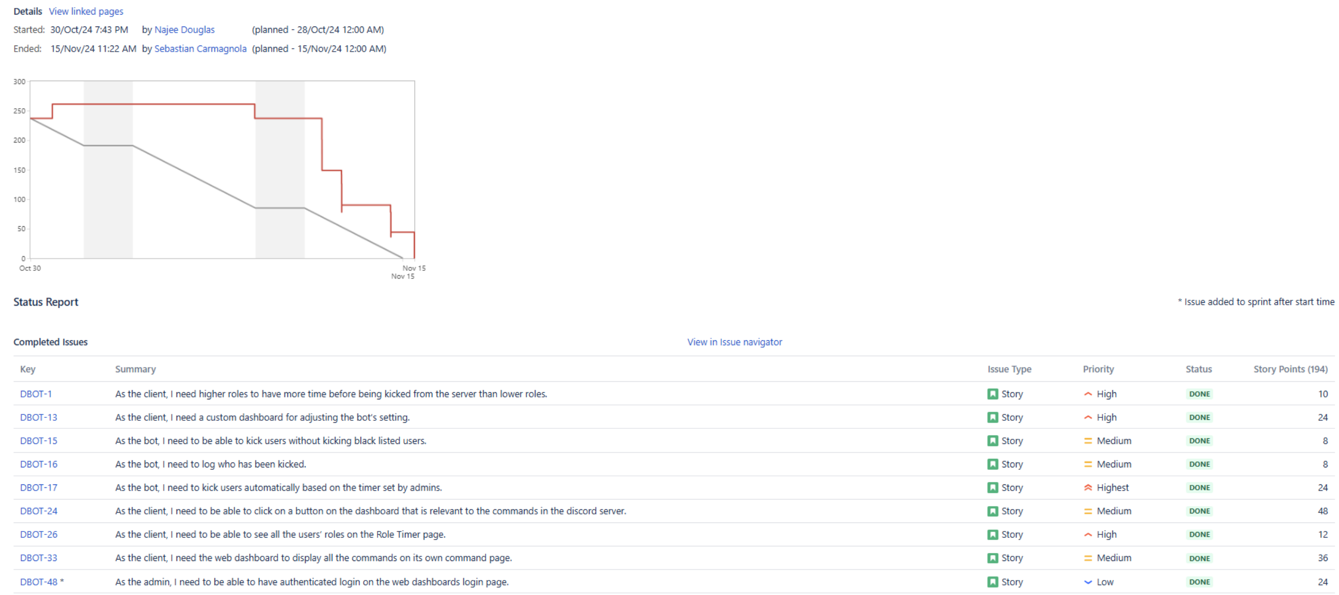
Status Report

* Issue added to sprint after start time

Completed Issues View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (23 → 31)
DBOT-7 *	As an admin, I want to be able to see a history of who called for a purge, when, and who was purged	Story	High	DONE	7
DBOT-10	As the bot, I need to be able to detect user activity via reactions in real-time NOT needing to restart myself to update the DB with the most current activity.	Story	High	DONE	3 → 8
DBOT-21 *	As the bot, I need to be able to detect user activity (text and voice) in real-time NOT needing to restart myself to update the DB with the most current activity.	Story	Medium	DONE	12
DBOT-23 *	As the bot, I need to log details of what user is being black listed (userid and username).	Story	Medium	DONE	- → 3
DBOT-27 *	As the user, I want the web dashboard to display all inactive users and user activities in a data grid so that I can sort each column to see activity by sorting user, channels, or last active date.	Story	Medium	DONE	1

Sprint II: This sprint aimed to complete several high and medium-priority stories. The velocity chart reflects a steady start, followed by a mid-sprint plateau and a significant drop towards the end, indicating a focus on finalizing tasks. Key accomplishments included successfully implementing real-time user activity detection without database restarts, logging purge history details, and managing blacklisted user data. The team also added functionality for displaying inactive user activities in a sortable grid. Despite minor timeline extensions, the sprint concluded with all targeted stories marked as done, showcasing effective task management and progress throughout.



Details View linked pages

Started: 30/Oct/24 7:43 PM by Najee Douglas (planned - 28/Oct/24 12:00 AM)

Ended: 15/Nov/24 11:22 AM by Sebastian Carmagnola (planned - 15/Nov/24 12:00 AM)

Status Report

* Issue added to sprint after start time

Completed Issues

View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (194)
DBOT-1	As the client, I need higher roles to have more time before being kicked from the server than lower roles.	Story	High	DONE	10
DBOT-13	As the client, I need a custom dashboard for adjusting the bot's setting.	Story	High	DONE	24
DBOT-15	As the bot, I need to be able to kick users without kicking black listed users.	Story	Medium	DONE	8
DBOT-16	As the bot, I need to log who has been kicked.	Story	Medium	DONE	8
DBOT-17	As the bot, I need to kick users automatically based on the timer set by admins.	Story	Highest	DONE	24
DBOT-24	As the client, I need to be able to click on a button on the dashboard that is relevant to the commands in the discord server.	Story	Medium	DONE	48
DBOT-26	As the client, I need to be able to see all the users' roles on the Role Timer page.	Story	High	DONE	12
DBOT-33	As the client, I need the web dashboard to display all the commands on its own command page.	Story	Medium	DONE	36
DBOT-48 *	As the admin, I need to be able to have authenticated login on the web dashboards login page.	Story	Low	DONE	24

Sprint III: During this sprint, we focused on completing high and medium-priority stories. The velocity chart shows steady progress at the start, followed by a plateau and significant drops indicating major task completions. Key achievements included implementing higher role-based server kick timing, further customizing the web dashboard for and adding a role timer view page and blacklist page – along with the functions being implanted for the blacklist. Additionally, critical high-priority tasks such as enabling authenticated login for the web dashboard and setting up auto-kicking mechanisms based on admin timers were completed. The mid-sprint and end-sprint declines in the chart signify focused efforts to finalize complex tasks. Despite challenges that might have contributed to periods of slower progress, the team successfully completed most of targeted stories, showcasing effective task management and collaboration throughout the sprint. We only had to backlog 3 or 4 tasks related to Role Timer Page.

What We Learned & Would Do Differently

Over the course of these three sprints, we learned valuable lessons in effective task management, work logging, and team coordination. In **Sprint I**, our main challenge was understanding how to break down tasks properly and log work efficiently, which led to delays in progress and late starts. This experience taught me the importance of structured planning and task allocation at the beginning of a project. By **Sprint II**, we improved by prioritizing and managing high and medium-priority tasks more effectively, allowing us to complete real-time user activity detection features, purge history logging, and enhanced data management functions. This sprint demonstrated our ability to adjust and deliver results even when facing minor timeline extensions. In **Sprint III**, we built on this momentum, showing steady initial progress and focused efforts on completing complex tasks like implementing role-based server kick timing and enabling authenticated logins. Although we faced challenges that slowed our progress mid-sprint, we were able to finish most of our targeted stories and only had to backlog a few tasks.

The biggest improvement we saw throughout these sprints was in how we managed and tracked our work. We learned to better break down tasks, prioritize efficiently, and focus our efforts when it mattered most, which enhanced our ability to complete more complex features on time. If we were to approach these sprints again, we would ensure a stronger emphasis on task definition and preparation in the initial planning phase, enabling smoother transitions through the different stages of each sprint. We would also incorporate more frequent team reviews and retrospectives during the sprint to identify and address potential issues early, keeping momentum consistent and reducing any mid-sprint slowdowns.