1. Features

- **Implemented**
    - Blacklist (User and Role Specific): Allows admin to blacklist specific users or roles from being purged.
    - User Activity Log: Updated log to store only the timestamps for when the user was last active.
    - Purge Command: Before any action is taken, the bot shows a confirmation message to allow users the opportunity to review and abort the purge if needed.
    - Purge Preview: When triggered, the command first generates a detailed preview listing the users eligible for purging, including a total count. With the option to edit any users without blacklisting them.
    - Schedule Regular Purges: The bot support user-configured scheduling for performing regular purges automatically.
    - Warning Message Command: Manual command that sends a message to all inactive users on the server that are not on the blacklist, warning them of an incoming purge. Followed up by a message that shows a list of warned users.
    - UI Design: Updated the UI design with a Discord-inspired dark theme, improved mobile responsiveness, and enhanced the navigation system.
    - Account Page: The Account Details page provides dashboard showing the user's Discord information.
    - Purge Page: Includes a refresh button to update the inactive users list in real-time. The updated data table now displays three key columns: Username, Last Active Date, and Time Since Last Active (calculated dynamically showing days and hours).
    - Purge History: The Purge History section now shows detailed logs of each purge action. Each entry displays who (bot or admin) initiated the purge, and the number of users affected. Expandable entries show the complete list of purged users when clicked.
- **Remain to be Done**
    - Multi-server Support: The bot should support server-specific configurations, allowing each server to define its own settings and preferences independently. This ensures that the behavior of the bot is not controlled by a single, universal configuration file, but instead adapts to the unique needs of each server.
    - Bug Fixes: Role-based Inactivity timing and no role-based blacklist functionality.

2. Known Issues

- Configuration Option 1: Review the Inactive Duration Role settings for text and voice channels. Verify which text channels require voice logs.
- Configuration Options 2: Review the Authorized Roles permitted to execute commands.
- Warning Command: Currently manual; ideally should be automatic on a time interval that is able to be set by the user.
- Bugs to Fix: The Inactive and Blacklist databases still retain purge users and needs to be viewed again.

3. Project Flyer

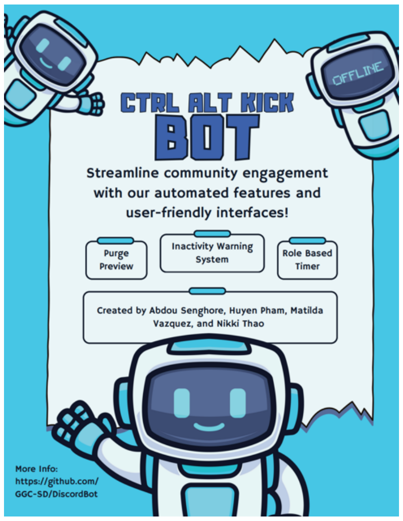
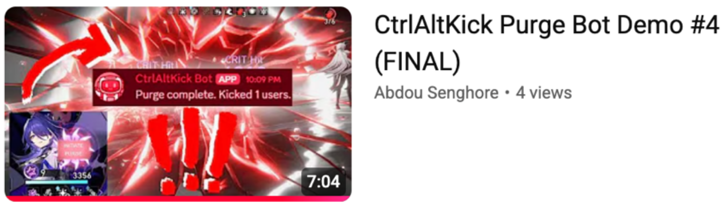

OFFLINE
CTRL ALT KICK
BOT
Streamline community engagement with our automated features and user-friendly interfaces!
Purge Preview
Inactivity Warning System
Role Based Timer
Created by Abdou Senghore, Huyen Pham, Matilda Vazquez, and Nikki Thao
More Info: https://github.com/GGC-SD/DiscordBot

Figure 1- Project Flyer

- The project flyer was created by Nikki Thao.

4. Project Demo



CtrlAltKick Purge Bot Demo #4 (FINAL)
Abdou Senghore • 4 views

Figure 2 – YouTube Video Demo Thumbnail

- The demo video was created by Abdou Senghore.
- The YouTube link to the project demo is: https://www.youtube.com/watch?v=Q5tvvNmHxUE

5. Clients

Figure 3 - David Rivera

Figure 4 - Mike Deiters

- David Rivera is a GGC alumni who is currently working as a Software Engineer.
- Mike Deiters is a GGC alumni who is currently working as a Senior React Native Developer.

CtlAtlKick Team

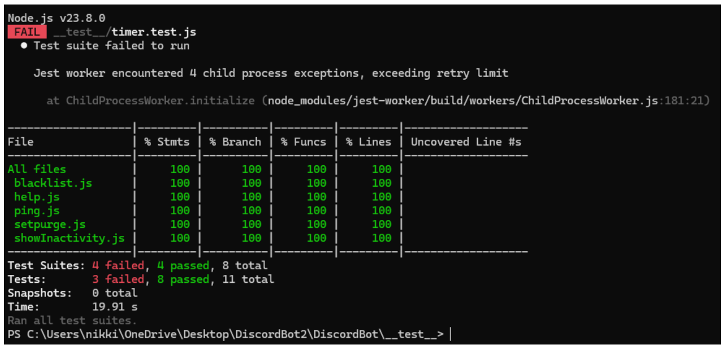
Figure 5 - Group Photo (Left to Right): Nikki Thao, Matilda Vazquez-Guzman, Huyen Pham, and Abdou Senghore

- Nikki Thao is the Testing Lead and Lead Programmer.
- Matilda Vazquez-Guzman is the Documentation Lead and Data Modeler.
- Huyen Pham is the Team Manager and a Data Modeler.
- Abdou Senghore is the Client Liaison and UI/UX Designer.

6. Project Abstract

- Discord is an instant messaging and social platform where users can communicate through text, voice chat, and video calls. As part of our Software development II course with Dr. Anca Doloc-Mihu, our team has been tasked with implementing new features and fixing existing bugs on the Discord Inactivity Purge Bot. This bot helps server owners maintain active and organized Discord communities by automatically removing users who have been inactive for a specific period. Key features include purge preview, inactivity warning message, and customizable blacklists. Through this project, we gained hands-on experience with full-stack development using Node.js, MongoDB, and React.

7. Testing



```
Node.js v23.8.0
 FAIL  __test__/timer.test.js
  ● Test suite failed to run

    Jest worker encountered 4 child process exceptions, exceeding retry limit

      at ChildProcessWorker.initialize (node_modules/jest-worker/build/workers/ChildProcessWorker.js:181:21)

-------------------|---------|----------|---------|---------|-------------------
File               | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
-------------------|---------|----------|---------|---------|-------------------
All files          |     100 |      100 |     100 |     100 |
 blacklist.js      |     100 |      100 |     100 |     100 |
 help.js           |     100 |      100 |     100 |     100 |
 ping.js           |     100 |      100 |     100 |     100 |
 setpurge.js       |     100 |      100 |     100 |     100 |
 showInactivity.js |     100 |      100 |     100 |     100 |
-------------------|---------|----------|---------|---------|-------------------
Test Suites: 4 failed, 4 passed, 8 total
Tests:       3 failed, 8 passed, 11 total
Snapshots:   0 total
Time:        19.91 s
Ran all test suites.
PS C:\Users\nikki\OneDrive\Desktop\DiscordBot2\DiscordBot\__test__> |
```

Figure 6 - Jest Testing Results

- **Coverage**:
    - This project uses Jest for testing for several key components, but there are some failing tests.
- **Methods**:
    - Unit tests for individual commands (ping, blacklist, help, etc.)
    - Mock-based testing to isolate components from external dependencies
    - Function testing for utility functions like inactivity tracking
- **Results**:
    - Test Suites: 4 failed, 4 passed (8 total)
    - Tests: 3 failed, 8 passed (11 total)
    - Snapshot: 0 total
    - Time: 1991s
- This project served as our introduction to JavaScript unit testing. While we encountered some test failures, we view them as valuable learning experiences. We also aim to further develop our skills with Jest through more practice.

8. Documentation

- **Installation Steps**
  - Set PowerShell Execution Policy:
    - Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
  - Clone the Repository:
    - git clone https://github.com/GGC-SD/DiscordBot.git
  - Install Required Software:
    - Install Node.js and Chocolatey
    - Install nodemon globally: npm install -g nodemon
  - Install all Project Dependencies into the src directory:
    - npm install dotenv (loads environment variables from .env file)
    - npm install discord.js (core library for interacting with Discord API)
    - npm install express (web framework for Node.js)
    - npm install mongoose (MongoDB object modeling for Node.js)
    - npm install cors (middleware for enabling Cross-Origin Resource Sharing
    - npm install axios (promise-based HTTP client for making API request)
    - npm install react-router-dom (routing library for React applications)
    - npm install react-scripts --save
  - Create a Discord Bot:
    - Go to Discord Developer Portal
    - Create a New Application
    - Navigate to “Bot” tab and click “Add Bot”
    - Under “Privileged Gateway Intents”, enable all required intents
    - Navigate to “Bot” tab and under “TOKEN”, copy your bot token
    - Navigate to “OAuth2” tab copy your Client ID and Client Secret (both under “Client Information”)
    - In “OAuth2” tab, under “Redirects”, add a redirect and copy your Redirect URL
    - Invite the bot to your server
  - Create a .env file in the src directory with:

    TOKEN=your_discord_bot_token

    databaseToken=your_mongodb_token

    GUILD_ID=your_server_id

CLIENT_ID=your_oauth2_client_id

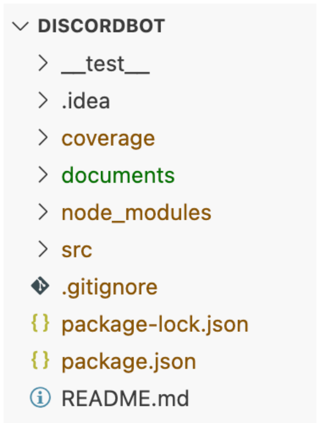
CLIENT_SECRET=your_oauth2_client_secret

REDIRECT_URI=your_oauth2_redirect_uri

- Run the Application:
  - From the src directory, run "npm start"
  - This will automatically start the Discord Bot, the backend server, and frontend dashboard
  - You should be able to see a login page
- Troubleshooting:
  - Check terminal logs for any errors
  - Ensure all environmental variables are correctly set
  - Confirm Discord bot has proper permissions in your server

- **Usage**
  - The Discord Bot is designed to help manage Discord servers by tracking user activity across channels, identifying inactive users based on configurable time thresholds, and purging them either manually or automatically. It also includes a blacklist system to manage user exceptions.

- **Help for User**
  - The /help command displays all available commands.
  - The dashboard's FAQ page answers common questions about the bot's functionality.
- **Code Documentation**
  - The codebase is structure as follows:



```
DISCORDBOT
  > __test__
  > .idea
  > coverage
  > documents
  > node_modules
  > src
  .gitignore
  package-lock.json
  package.json
  README.md
```

Figure 7 - Code Structure

9. **Usability Testing and Survey Results**
    - No usability testing and survey results to provide.

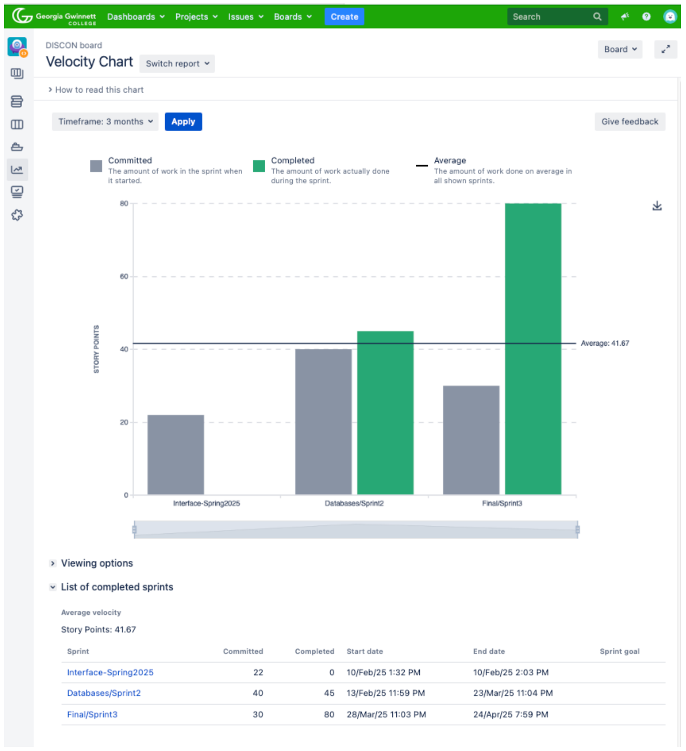
10. **Where to Find Installation, Developer, and User Documentation**
    - For Spring 2025, all documentation is primarily located in the documents/docs-Spring2025/ directory.
    - Installation instructions can be found in the main README.md file under the "Installation Steps".
    - Developer documentation is location at documents/docs-Spring2025/TODO.md

11. **Software Usage, License, and Intellectual Property**
    - **Software Used**
        - Core Technologies: JavaScript and Node.js
        - Frontend: React, Material UI, React Router DOM, and Bootstrap
        - Backend: Express.js, MongoDB with Mongoose, and Discord.js
        - Testing: Jest
        - Utilities: Axios, Dotenv, and Cors
    - **Licensing**
        - The Spring 2025 license information is stored in documents/docs-Spring2025/License.md
        - CtrlAltKick Bot © 2025 by Abdou Senghore, Huyen Pham, Matilda Vazquez-Guzman, Nikki Thao, Anca Doloc Mihu, David Rivera, Mike Deiters is licensed under the MIT License.
    - **Intellectual Property**
        - The Spring 2025 signed Intellectual Property Agreement is stored at documents/docs-Spring2025/IntellectualPropertyAgreement.pdf
        - The document includes the signatures of all four team members (Abdou Senghore, Huyen Pham, Matilda Vazquez-Guzman, and Nikki Thao), as well as the clients (David Rivera and Mike Deiters) and the ITEC 3870 professor (Dr. Anca Doloc Mihu).

Name	Percent	Signature
Abdou Senghore	10%	[illegible]
Huyen Pham	10%	[illegible]
Matilda Vazquez-Guzman	10%	Matilda Vazquez
Nikki Thao	10%	Nikki Thao

Anca Doloc Mihu	10%	*ADM*
David Rivera	25%	DAVID RIVERA
Mike Deiters	25%	Deiters

12. Velocity Chart Report



Velocity Chart

Timeframe: 3 months

▨ Committed — The amount of work in the sprint when it started.

🟩 Completed — The amount of work actually done during the sprint.

— Average — The amount of work done on average in all shown sprints.

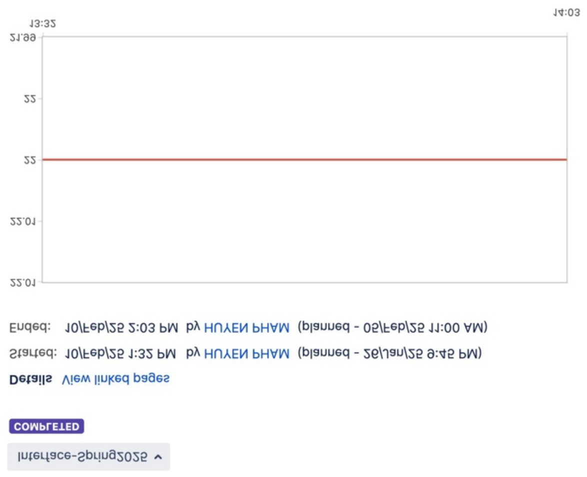
Average velocity

Story Points: 41.67

Sprint	Committed	Completed	Start date	End date	Sprint goal
Interface-Spring2025	22	0	10/Feb/25 1:32 PM	10/Feb/25 2:03 PM	
Databases/Sprint2	40	45	13/Feb/25 11:59 PM	23/Mar/25 11:04 PM	
Final/Sprint3	30	80	28/Mar/25 11:03 PM	24/Apr/25 7:59 PM	

- The velocity chart from Jira displays the team's progress from Sprint 1 to Sprint 3.

- Sprint 1:



Interface-Spring2025

COMPLETED

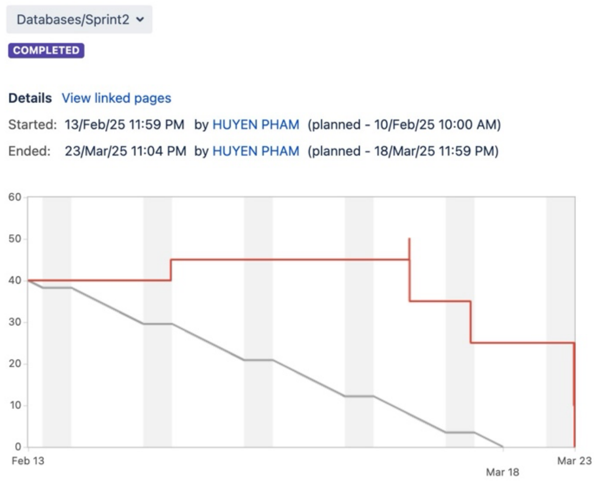
Details View linked pages

Started: 10/Feb/25 1:32 PM by HUYEN PHAM (planned - 26/Jan/25 6:45 PM)

Ended: 10/Feb/25 2:03 PM by HUYEN PHAM (planned - 05/Feb/25 11:00 AM)

  - This sprint did not go well. We committed to 22 story points but completed 0. One of the biggest reasons was that we didn't fully understand how to start and complete the sprint in Jira. Because of this, we messed up the sprint flow, and tasks were not properly tracked or finished. We also struggled with unclear user stories and didn't plan carefully. From this, we learned how important it is to understand the sprint process and prepare better before starting. This sprint showed us what happens when planning and process are weak.

- **Sprint 2**:



Databases/Sprint2

COMPLETED

Details View linked pages

Started: 13/Feb/25 11:59 PM by HUYEN PHAM (planned - 10/Feb/25 10:00 AM)

Ended: 23/Mar/25 11:04 PM by HUYEN PHAM (planned - 18/Mar/25 11:59 PM)

  - This sprint was much better. We committed to 40 story points and completed 45. After learning from Sprint 1, we fixed our mistakes. We understood how to properly start and finish the sprint, and we planned our tasks more clearly. We also communicated more during the sprint, which helped us stay on track. This improvement showed how learning from earlier mistakes and working together better can make a big difference.

- **Sprint 3**:

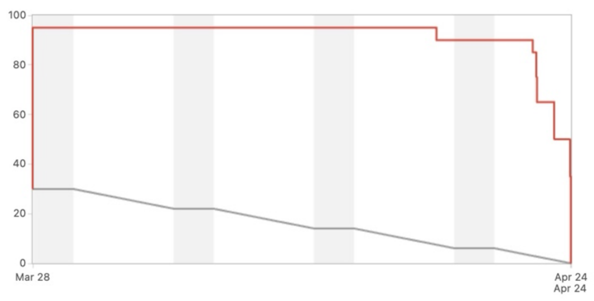
Final/Sprint3

COMPLETED

Details View linked pages

Started: 28/Mar/25 11:03 PM by HUYEN PHAM (planned - 27/Mar/25 7:23 PM)

Ended: 24/Apr/25 7:59 PM by HUYEN PHAM (planned - 24/Apr/25 7:23 PM)



  - The final sprint was our strongest. We committed to 30 story points but completed 80. By this time, we were very comfortable with the sprint process and how to work together. We divided tasks clearly, stayed motivated, and solved problems quickly. This sprint showed how much we had grown as a team and how effective we became when we communicated well and used our experience from the past sprints.

- **What We Learned and What we Would do differently?**
  - Overall, we learned a lot about teamwork, planning, and using Jira properly. At first, we didn't really know how to start and complete sprints, which caused problems in Sprint 1. But we figured out what went wrong and made big improvements in Sprint 2 and Sprint 3. If we could do this again, we would make sure to fully understand the sprint process before starting. We would also plan our tasks more clearly and hold more regular check-ins during sprints to avoid confusion or slowdowns. This project taught us that with experience, good communication, and teamwork, we can overcome early mistakes and finish strong.