Cloud-Based Discord Bot

Elijah Baldwin, Sean Larkin, Shane Maloney, Michael Pastorino, and Noah Winand

https://github.com/sm938792/CSC468-Group4/tree/Project

West Chester University

CSC-468-02: Introduction to Cloud Computing

Dr. Linh Ngo

May 6th, 2022

Project Summary

The goal of the project is to develop a cloud-based Discord bot. The main utilization of the bot will be the ability to use the bot for moderation purposes within a Discord server without needing to manually type commands. This will be accomplished by using a web UI in order to simplify the process for ease of use. The web UI will be using the Discord API for the execution of the commands, and for checking the login credentials of users. MongoDB will be used to maintain a database of tables which will check if a user's login credentials are accurate to moderate a specific Discord server. It will also be used to track the data of members within the server and if moderative actions were taken against them. The cloud aspect will be the hosting of the bot itself on the cloud via CloudLab. The web UI, MongoDB database, and bot itself will all be containerized. Kubernetes and Jenkins pipelines will be in place to automate and streamline changes to each aspect of the project.

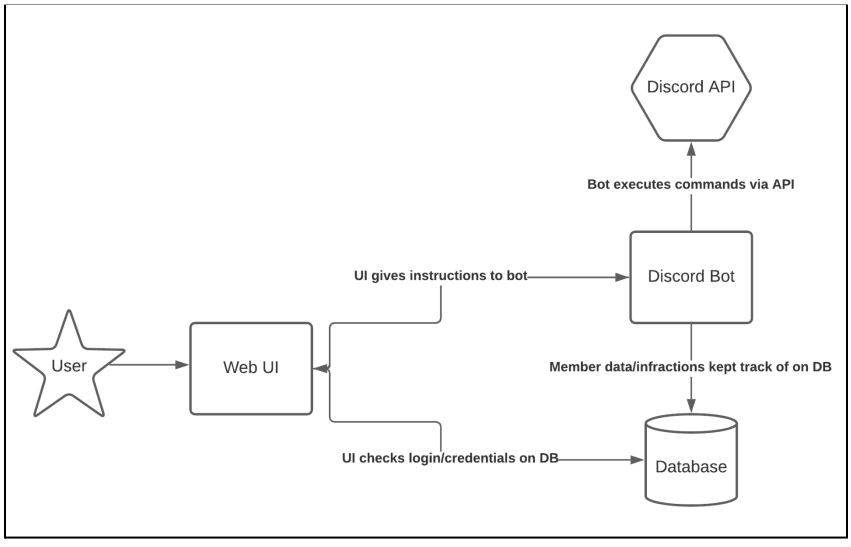
Implementation of said pipelines, though, proves challenging primarily in the setup process. Due to the nature of the project, there is a great security risk and concern regarding the privation of the bot's token. Courses of action to privatize the token while maintaining functionality also has been a challenge to properly implement. The mechanics of the database, primarily the functionality of the database table, has also been an obstacle in maintaining data produced by the bot. A lot of the sources to these problems and challenges have been found, but implementation and course of action is the main obstacle. The bot's basic and core actions work completely locally. Commands for the bot have been added and implemented for further functionality and use within servers. Also, the bot itself is able to establish a connection to the database with ease. Progress was made to implement a Jenkins pipeline, and the groundwork and core functionality of the web UI has been completed.

Team Description

Overall, we all have experience with a lot of different programming languages, techniques, and medium-scale projects, so any new languages that may be a part of this project should not pose that much of an issue for us. Additionally, we are mostly balanced in our experiences as a whole, so we should all be able to contribute evenly and effectively without one person knowing a lot more than the rest. One of our team members has already created a Discord bot before, so we expect to use that bot as a template for creating a new, cloud-based one. Resumes can be found at the end of this report.

Technical Report

Chapter 1: ***Team Vision***



Currently, our vision for this project is to fall within the A-level technical requirements. While creating the cloud-based Discord bot will be relatively straightforward, as we plan on following a similar front end model to most other discord bots, the challenging part of this project will be to get the Discord bot to properly talk to the Discord API and database.

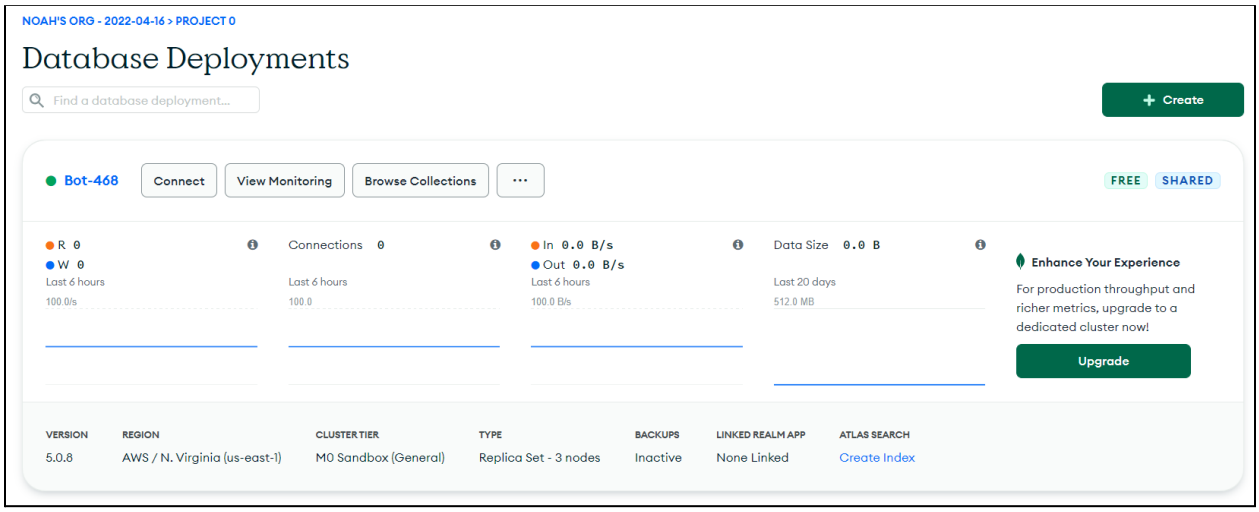
As for the design of our project, we are adapting pre-conceived models of Discord bots to include Jenkins and CI/CD services, a docker image hub, a webui separate from Discord, and a kubernetes deployment converted from docker-compose infrastructure. Additionally, we will be replacing the traditional redis database service with MongoDB, which is known to work well with Discord's JavaScript services. In essence, our project focuses on the conversion of moderative Discord bots to a more user-friendly and cloud-based service.

Chapter 2: ***Addressing Technical Requirements***

Keeping mostly in-line with the standard discord bot model, all services should work in a similar fashion, but with the addition of cloud services. First, the user will enter their username and password with the Web UI. The web UI will then send the credentials to the database. Once verified, the user will have access to all of the bots features, and be able to carry out commands with a single press of a button. The bot will then be able to execute commands given to it on the Discord server it is a part of through the Discord API. Any actions taken by the bot will be logged to the database for future reference.

Currently, with our limited knowledge of cloud computing, we expect to fulfill most of the technical requirements as we progress through this course. Doing the hands-on activities as a group and adding on to the project over the course of the semester will ensure that we meet all the C-level requirements in a reasonable timeframe. To address the A-level and B-level requirements, we will need to translate all we learn about redis into MongoDB, in order to replace redis with MongoDB. Some of our team members have prior knowledge of MongoDB, so all that will need to be done is to have MongoDB provide roughly the same functionality as the redis database by translating redis's language into MongoDB.

Chapter 3: ***Intermediate Milestones***



Initially our project was a B-level coin miner utilizing MySQL. Upon becoming more familiar with cloud computing, the expectations of the project, and learning more about the capabilities of our team members, we transitioned towards a cloud-based Discord bot while using MongoDB as a way to manage the structure of the database. This has resulted in time taken to assess the requirements needed to accomplish this project as some of our current ideas and tasks differ from our original project for the coin miner.

Thus far we have created a basic functioning Discord bot, and the implementation of MongoDB has started. Due to the nature of MongoDB being a noSQL language, the bot and database work in the same files. These files are written in JavaScript, the language being shared by all components of this project. As far as how it accesses Discord, the bot first asks that the host provide the required “discord.js” installation. Although Docker has not been fully implemented yet, the Dockerfile is in the position that, were it implemented, it would successfully install the prerequisites the JS files require. This is done via 5 RUN statements, calling various “npm” libraries. It then executes two commands using “node” to run the JS files themselves. Moving away from the Dockerfile and back to the main bot, after the host has demonstrated that the required packages are installed, it calls for the .env file. This is where we have run into a minor hurdle. For the purposes of this project, it would be insecure but adequate enough to leave the SRV of the database and token of the Discord bot out in the open for anyone to utilize. However, in the real world, this practice would be a severe security risk, as access to the SRV and token give practically full control over the discord bot and data stored on a database, leading to password leaks. For now, we have decided to focus on other aspects instead. After the calling of the .env file, the bot logs its progress in the consoles. Command implementation is barebones so far, but what is important for now is that the DB was successfully accessed by the Discord-integrated bot. The library “mongoose” gives access to MongoDB, but the bot needs the specific database’s SRV in order to connect to it. For now, this SRV is located within the uncommitted .env file. By using the function mongoose.connect(), it can be determined whether or not a connection between the two elements has occurred. Unfortunately, we have not figured out how to connect from anywhere, as MongoDB only allows pre-specified IPs to access the cluster, so for now we have manually added all of the group members’ IP addresses to the whitelist. Nevertheless, if this connection is successful the bot will log that success into the console, otherwise it will throw an error. Tests have shown that with a whitelisted IP and a correct SRV token, the discord bot can always enter the database with no issues. Finally, the bot is ready to log into a discord server. Again, the token required to login to the server is within the .env file, but even so, tests have shown that the bot successfully runs all previous commands and shows up as “online” on the test discord server it is stationed on. Through one JS file, we were able to connect the DB to the bot and have it successfully appear on a server.

The core planning for how the web UI will function and interact with the Discord bot has been decided, and with that the web UI has also begun being developed. Although there is not much to show related to the web UI, the files needed for it have been created, and a template has been added to assist in the process.

Overall, assessing the project and what needs to be accomplished seems very feasible. The established skill sets of our team members should transition well into implementing MongoDB and understanding the functionality using the Discord API. The bulk of our work will come down to the forming of the web UI and the implementation of the cloud-based elements, with MongoDB and the Discord bot being a satisfactory position.

Upcoming challenges are the complete implementation of Kubernetes and Jenkins as well as the implementation of the Discord API to communicate between the bot, database and the web UI fully. We understand what goes into accomplishing these tasks conceptually, but currently taking we know of these systems and implementing their framework into our current infrastructure may prove quite challenging. This is on top of ensuring our docker and yaml files are properly configured for their implementation. Additionally, due to the small number of components relative to the template “ram-coin”, dividing up the work so that everyone is contributing adequately has proven difficult, and having the Discord bot and MongoDB aspects a part of the same component means that one cannot go on without the other’s contribution. We are unsure if we will complete the project in time, but with enough dedication to each of the aspects, as well as a far greater emphasis on Kubernetes and Jenkins, we will be able to create something of value.

Chapter 4: ***Project Final State/Self-Evaluation***

In the end, our project was not able to become “cloud-based” like we had hoped due to a number of reasons. However, we have made significant progress on some of our aspects, and have learned much from this project as a result.

On the side of MongoDB and the Discord bot, we have since found out that MongoDB can allow all IPs to be whitelisted by adding in the IP 0.0.0.0/0. With that hurdle out of the way, we were able to successfully commit all commands the bot should be able to use. After redoing the infrastructure due to deprecation of previous Discord.js versions, the bot is now able to perform and successfully execute three basic moderation commands. These commands include Timeout, Kick, and Ban. Event handling is used to read the user text inputs. It is triggered by a “/” followed by the command along with the target’s username and an option to add a reason for the punishment. Before a command is executed, Bot 468 makes sure the user trying to execute the command has the proper moderation permissions needed for each.

On the Kubernetes/Jenkins and Web UI side, progress was made in both fields, though they remain largely incomplete due to complexity or time constraints. For the cloud deployment, we began by copying the template files from csc468cloud’s kubernetes-jenkins branch. This way we could follow the class slides in order to get the basics of Jenkins running. At this stage in the project, it was unlikely that full CI/CD integration was possible, so we settled on just getting the standard parts complete. Unfortunately, we ran into some issues while following the slides and could not get it fully operational after the CloudLab project went down for some time, but the basic yaml files for Kubernetes and Jenkins were created, so development could theoretically continue if it were to be picked up in the future. For the WebUI, a basic website exists and the Nginx web server is set up with a default configuration. The website provides a login page when accessed, though unfortunately that is the end of its functionality as it has not been configured to work with the Discord bot and database.

In chapters 1 and 2, we discussed the technical requirements of our project. Overall we completed some of them, those of which turned out easier than previously expected, but were not able to complete all of them. Strangely enough, it was parts of the back-end that were easier, while the front-end ended up not being completed in time. Of course, the back-end process is more demanding, and overall more important to the project, so it is understandable why this was the case in the end. The back-end process that we thought would be very difficult is the communication between the Discord API and the database, with the bot itself being the middle-man. Upon further research on the noSQL database MongoDB, which is said to work well with all discord bots, it turned out that communication between these two aspects was not too challenging, at least on the local end. As stated in the previous chapter, JS has a library that can be installed to allow for easy MongoDB integration and, since all of our project is based around mainly JavaScript, this meant that there would be no issue in communicating with the Discord API unlike what we believed. That was one requirement we fulfilled successfully, but one requirement from chapter 1 that we did not finish was the integration of Jenkins and CI/CD services. At the time of this project’s creation, we believed the correct course of action would be to first focus on getting everything working locally instead of starting directly with CI/CD services. Unfortunately, we realized too late that CI/CD services were the hardest aspects to integrate, and that they can dramatically change how a bot operates from outside local systems. In retrospect, the better course of action would have been to integrate Jenkins inside a mostly generic repository, then procedurally add more to make it into a working discord bot. Overall, for chapter 1, we succeeded in the task we thought would be difficult, and failed in the one we thought would be easier.

For chapter 2, we were able to address the requirement for carrying out commands and verification for those commands, but were not able to integrate it with a web UI nor add proper tables. Discord bots come pre-equipped with a way to not allow those without the proper

permissions to carry out important commands such as “ban” or “kick”, so the integration of that requirement without the use of a web UI was simple. Additionally, by adding a few JS files and parts of the Discord library, we were able to have the bot carry out the moderative commands we desired. “Ban” blocks an account from ever returning to the server it was banned from, so in a way this action can be considered “logged to a database”, though not to our database specifically. In the end, bot functionality was not an issue, but the database and web UI aspects were. Because of some sickness and busy schedules, the web UI was not able to be completed on time, and remains a relatively barebones webpage. Additionally, although a MongoDB cluster was created and successfully connected to the Discord bot, no tables were created for storage. Due to the very likely possibility that the web UI and cloud-based services were not going to be completed on time, we decided to forgo the adding of tables while we worked on getting the mostly incomplete aspects at least slightly more complete. Overall, we were unable to fulfill the cloud-based and web UI technical requirements of chapter 2, but were able to get some items working for the database and everything working as intended for the Discord bot itself. In other words, some technical requirements were completed while others were not.

In hindsight, a more rigid schedule for getting certain parts done throughout the semester, as well as focusing most efforts on the cloud-based implementation, would have served us better to get more done by the end. That being said, the busy schedules with both work and school that each of our team members had hindered us a lot more than we had anticipated. Even if we did not manage to complete the project, however, each of us learned much about how cloud services work, and how a company deployment of a cloud-based infrastructure might look and be implemented in the field. The process was a learning experience, and one that we enjoyed undertaking.

References

Dr. Ngo's Introduction to Cloud Computing slides

Project repository: https://github.com/sm938792/CSC468-Group4.git

CloudLab profile: https://www.cloudlab.us/show-profile.php?uuid=a2e13ddf-82e4-11ec-b318-e4434b2381fc

MongoDB: https://www.mongodb.com/docs/manual/tutorial/getting-started/

(Resumes located on the following pages)

Elijah Baldwin

Philadelphia, PA

Education

- West Chester University of Pennsylvania B.S. Computer Science(May 2022 Anticipated)
- GPA: 3.4

Relevant Coursework

- Software Engineering
- Data Communications and Networking
- Data Structures & Algorithms
- Computer Security & Ethics
- User Interfaces
- Digital Image Processing

Experience/Volunteer

Franklin Institute, PACTS Program, Robotics Explainer (2015-2018)

- Assisted students in building robots and understanding the fundamentals of robotics
- Assisted judges at the First Lego League Competition
- Explained and demonstrated the many functions of robots to visitors
- Designed robots to showcase at events

Skills

- Java
- C
- React-Bootstrap
- Python
- HTML
- CSS
- Google Products(Drive, Docs, Slides, Sheets, Forms)

Awards

- Dean’s List Spring 2020 & Spring 2021 semesters at West Chester University of Pennsylvania
- Kappa Alpha Psi Scholarship
- McKee Scholarship

Sean Larkin

West Chester PA | 17slarkin@gmail.com | (484)-999-1900

Education

West Chester University
August 2019 - (Expected Graduation) May 2022
B.S. in Computer Science
Computer Security Certificate (*program certified by the NSA*)
GPA: 3.42

Penn State Brandywine
August 2017 - May 2018

Downingtown West High School
Graduated 2017

Skills

- Proficient in Microsoft Office
- C/C++ Experience
- Java Experience
- Effective Communicator
- Team-player
- Quick Learner
- Logical Thinking / Problem Solving

Shane J Maloney

Philadelphia, PA | (215) 373-9492 | shanejmaloney@gmail.com

Education

Delaware County Community College, Marple, PA
August 2016 - May 2018
Certificate: TestOut Network Pro

- GPA: 3.6

West Chester University of Pennsylvania, West Chester, PA
Bachelor of Science in Computer Science | December 2022 (Expected)
Certificate: Computer Security

- GPA: 3.7

Skills

- Help Desk Support
- Windows and Linux Operating Systems
- Networking and Security Protocols
- Java
- JavaScript
- C
- C++

Experience

Legal Secretary | July 2015 – present

Information Technology Assistant | November 2020 – present

- Providing and maintaining company networks and servers
- Troubleshooting and solving technology and system problems

Michael Pastorino

West Chester, PA ✚ 484-947-7972 ✚ Pastorino173@gmail.com

Education

-Expected Graduation Spring 2022 Bachelors degree in Computer Science / West Chester University, (PA) GPA 3.70
-2018 Associates in Mathematics / Delaware County Community College (PA)

Projects

-Virtual ATM, a Virtual ATM that asks a user for a pin with a max number of attempts that then allows the user to check the balance, withdrawal, and deposit money into the account. It also has a set limit on the amount of money to be withdrawn in a transaction.
-Sudoku checker, which utilizes arrays and loops to run through a sudoku puzzle to check for any mistakes in that puzzle and if a mistake is found it reports back to the user where the puzzle fails.

Experience

February 2021 - present: Part time installation specialist / Castelli Flooring, Elsemere, DE

- Engages with prospective and current clients to install the solution requested on time and under budget.
- Performs comprehensive removal and installation services for tile flooring products with precision and
- Demonstrates focus and attention to detail to ensure a smooth installation.

March 2020 – present Part Time Shopper Assistant / Instacart, *PA, DE, and NJ*

- Performs grocery shopping for customers via list provided on Instacart App.
- Responsible for accuracy of selection, and offering suitable alternatives to customers for items not available
- Provides utmost customer service to ensure customer's needs are met

Program Languages

➢ Java, C, Python, javascript

Software

➢ J-Graphs, Eclipse, Visual Studios, ios Terminal, Anaconda, Java Script, Microsoft Office Suite.

General Skills

- Problem Solving
- Working in teams
- Learning new programing languages
- Both leading and following
- Multi Tasking
- Customer Service

Noah Winand

277 Edgehill Rd.
York, PA 17403
(717) 814-0195
nwinand128@gmail.com

Education: West Chester University of Pennsylvania

Bachelor's Degree in Computer Science -- Expected in 2022
Certificate in Computer Security (current overall GPA: 3.7)
Relevant Courses Taken:

- Operating Systems + Data Communications + Programming Paradigms
- Big Data Engineering + Modern Malware Analysis + Software Security

Work Experience: KioWare Kiosk Software

Custom Development Intern -- June 2021 - August 2021

- Created a Java GUI to automate the process of publishing companies to KioWare's SQL database from Excel spreadsheets
- Created two additional, reusable Java GUIs that allows the user to edit company information and publish/unpublish companies from the public website
- Modified a pre-existing trade show example by improving graphics and making it WCAG accessible up to level AA (HTML and JavaScript)

Awards

- Tri-M Music Honor Society - A high-level musical club that requires a solo tryout session, and only accepts a few members
- Achieved Dean's List during the Spring 2020 semester at West Chester University of Pennsylvania

Technical Skills

- **Programming languages Java, SQL, C, Haskell, HTML, and Python**
- The conversion of back-end code to easily readable front-end GUIs
- Experience with common computational tools such as ghci, podman, SQL Server Management Studio, MobaXterm, Eclipse, and Cisco Packet Tracer
- Conversion of assembly code into high-level programming languages
- Microsoft & Google Products (Word/Docs, Powerpoint/Slides, and Outlook)
- Adobe Photoshop & Illustrator