**Gaiza: Discord Utility Bot**

Final Report

**Slippery Rock University**

Cain Kaltenbaugh <cxk1047@sru.edu>

**Reference**

<https://discordapp.com/>: Discord application download

<https://github.com/Javacord/Javacord>: Javacord GitHub repo that also contains links to important setup information

<https://github.com/IKanraI/Gaiza>: Link to the source code for the bot

**Completion**

The project is completed and is functional. Various commands have been implemented as well as a storage structure for server and user bases. Every command has some form of error handling to help the users understand how to use the commands. User and admin commands are implemented and separated appropriately and are documented in the help menus. Current working commands are listed in the manuals and welcome modules were implemented for admins of servers to utilize.

Some functionality is not entirely in, such as tracking user information for the uwu response. Purge deleting messages, headpat, hug, and slap commands were not implemented. Other commands for admin use were put to the side for structure management.

**Contribution**

This project was written by myself and the thoughts and ideas behind the concepts were my own. The source of the project was written in 2017 with some simple node.js commands that were not structured. Project conversion to Java began at the start of the semester and was made very modular and reusable. Concepts for commands and such are basic bot functionalities seen in other bots, but the database design and JSON storage was my own creation.

**Glossary**

**Javacord:** Multithreaded library for Discord that can allow the bot to call upon events, listeners, and different parts of the Discord api.

**Discord** – Platform the bot operates on. This is a social platform that contains servers with users on it.

**Bot** – This is what interacts with the user and returns responses in the Discord server.

**Problem Explanation**

Discord allows users to utilize the platform with a series of built in commands. Bot’s allow users to take on more command line approach to Discord’s use. Instead of looking for a user’s username, they can simply $ban [user] to make use of a bot command. Discord does not include very many interesting or fun commands. Bots allow for a plethora of game commands. Some bots have an entire purpose of providing a game to a server. This bot’s purpose is to provide utility to both users and admins. Users have an expanded range of commands that allow them to get more out of Discord. Admin commands are provided for allowing the admins to manage the server better. Some commands even include a modifiable module that can welcome users into the server. Early iterations of Discord did not have this functionality, and this was a coveted idea from the early days. Aside from that, welcomes can be customized instead of taking Discord’s random phrases for users.

**System Requirements**

* System running a windows environment
* Discord
* An IDE such as Eclipse
* Internet connectivity

**Caveats/Minefields**

The gif command still has some issues returning gifs properly. Partially part of the platform and partially because the error handling is not up to par. There needs to be a variation of gifs returned based on how many results are obtained from the search result. Sometimes when the search returns, it returns out of index and cannot get a result thus causing the command to break. This issue can and needs to be resolved.

Another issue is the define command will not always send the embed. This happens when the text retrieved has a larger character count than the embed can handle. If this happens, nothing is returned. This can be fixed easily; the time had just not been taken to implement the fix.

**Code Reusability**

The code is very reusable. For example, the bot has certain properties assigned to it. In order to change the properties, there is a BotInfo package to handle all of this. Any method can call upon it in order to change or retrieve these. The server management allows each server to have its own prefix. The prefixes are all called based on the server ID and it has a prefix stored in the database. Every server that calls a command goes through the same Keywords package to get their prefix. This allows the code to be reused every call and not need to be written into every command.

**Deployment / Maintenance**

The program runs off an executable jar file. Once an executable jar file for the code is exported, the user only needs to run it. A .bat file can be made to run it in a windows console with the following commands in the .bat file:

:a

java -jar gaiza.jar

goto:a

If you a bat file is not made, the task will need to be killed since the user wont be able to see the windows console for it. To kill the task the task name needs to be found with:

jps

After the task id is found the user can type in:

taskkill -f /PID [id number]

This will allow the user to end the task. Otherwise, multiple instances of the bot will appear. The bot can also be run through the Eclipse debugger to test while writing new implementations. Make sure to stop each task before running a new one with the red stop button. Otherwise, multiple instances of the bot will be running and the processes for them will need to be killed like in the example above.

**Post-Mortem Analysis**

For a more effective work environment, I would have taken more time to work on the commands. Instead of doing several modules in one week, I would have more time to debug and analyze issues of the other commands. Some issues still have not been mended yet. Most fixes occurred in the following weeks while writing new commands for the program. This is not ideal, and I did not give myself enough time to properly handle any of the previous issues by cutting close on deadlines. I would also structure the bot a little differently in regard to calling commands. Instead of instantiating every command from the start, I would make it so commands are only instantiated when the prefix is called, which will modify the internal structure of each command, but make it easier on the bot for processing what is being called. This will be in future implementations, but I have not assessed this issue yet.