**Capstone Project Report**

Discord Tournament Administration

Kennesaw State University

IT4983-W01 Capstone

Lily Lama-Solet - Shane Wild - Brandynn Daugherty - Grant Richardson - Jonathan Pass

Project Website: https://github.com/llamasol/DiscordTournamentBot

March 24, 2024

# **Executive Summary**

With Kennesaw State’s Esports division, there still remains a great deal of manual labor needed to initiate and manage in-house tournaments. Because of this, tournaments take an unnecessary amount of time to get started. To facilitate in-house tournaments, KSU Esports admins utilize Discord for communication with players. This provides a way for players to check-in to tournaments and be placed into teams by admins based on their in-game rank.

The goal of our team is to develop a tournament bot on Discord that the admins and players can use. With a Discord bot, players can check in faster and be matched up in teams more efficiently. Admins will also be able to use the bot for moderation of the matches and push or pull player statistics from a spreadsheet instantly.

# **Table of Contents**

[**Executive Summary 3**](#_9qw9fbo1583l)

[**Table of Contents 4**](#_xnqihw6wc777)

[**Background 5**](#_251jqpcallh8)

[**Project Outcomes and Achievements Summary 6**](#_gi7i992jdnk)

[**Project Planning and Management Summary 7**](#_kgukn8a6lbkx)

[**Team Reflection on Project Experience 8**](#_28wie7d3bb4w)

[**Appendix 9**](#_yydie8pe4cuf)

[**Bibliography 10**](#_txn80enjogci)

# **Background**

Kennesaw State University runs an eSports program that regularly hosts events and tournaments for games like League of Legends and Super Smash Bros. As of right now, tournament administration tasks like team formation and keeping track of player statistics is all done manually, specifically using spreadsheets. Because of this, the overall tournament administration process can take more time than it should, which leads to players waiting an excessive amount of time to be put on teams and can delay the tournament start time. The current system also creates the possibility of human error when tracking player data and doing other tasks, since everything is done manually.

The goal of this project is to automate the tournament administration process. In order to achieve this, we will create a Discord bot that will be able to handle tournament administration tasks. For example, the bot will be able to assign players into teams automatically based on relevant player information and determine which teams will go against each other for each match. Matchmaking algorithms will be implemented into the bot’s code in order to achieve this. Other tournament administration tasks like keeping track of and storing player statistics will also be handled automatically by the bot. Also, this will be a bot that not only tournament administrators can use, but it will be something that players will be able to use as well. Players will be able to check-in by simply interacting with the bot and using a check-in command that will allow them to leave and check-in.

In terms of the more technical side of the background around this project, we used the programming language Python for the code for the bot, since it is very user friendly and easy to use when compared to other languages. We also made use of Discord’s Python API library, and it allowed us to add functionality to the bot. The Discord API library lets the bot be interacted with, and allows it perform a variety of actions, like responding to certain text prompts and performing an action after a command is issued, using Python code (Welcome to discord.py). We also made use of Google Sheet’s API library, which we needed in order to allow the bot to read and write to Google Sheets spreadsheets. The Google Sheets API allows users to do many things like create Google Sheets spreadsheets, read from and write to spreadsheets, and update or edit cells within spreadsheets using code (Google). Lastly, we made use of Riot’s API in order to pull player data from Riot’s database. Riot’s API can be used for a variety of things like retrieving player data from games like League of Legends and Valorant and analyzing it (Riot Developer Portal). The overall intent of this entire project is to automate various tournament administration tasks by creating a Discord bot that can perform these tasks with ease.

# **Project Outcomes and Achievements Summary**

The first task was to create the Discord bot. This was done by creating a new Discord Application in Discord’s Application page. The bot was given the appropriate permissions based on what features the admins will use. Once it was created, a URL was generated by ticking the necessary boxes for the bot’s scope and permissions in the OAuth2/URL Generator tab. With this URL, the bot was able to be invited into the KSU Esports test server. (Creating a bot account)

The next step was to develop several admin commands. The commands are check-in, volunteercheck, toxicity, and MVP. The check-in, volunteercheck, and MVP commands will provide buttons for users to click to check into tournaments, volunteer out or rejoin a tournament, and allow players to vote on an MVP, respectively. The toxicity command is explicitly used by an admin for moderation purposes for when a player is acting in a negative manner.

For the buttons that appear from the check-in, volunteercheck, and MVP commands, we had to implement them to respond directly to each individual player privately so as to not flood the text channel with messages. We also implemented a class in the code to create the buttons and their functionality. The buttons were given a timeout so it would be unusable after 15 minutes. Each one was also given a different color and label to help users differentiate between them. We initially wanted a button to be disabled for a user when they clicked on it; however, if you set a button to be disabled, then after the first click it is disabled for everyone. To work around this, when a button is clicked the first time, it will send the initial message based on the button; then if a user tries to click on a button a second time, it will let them know they have already done so. (API docs for bots and developers; lykn)

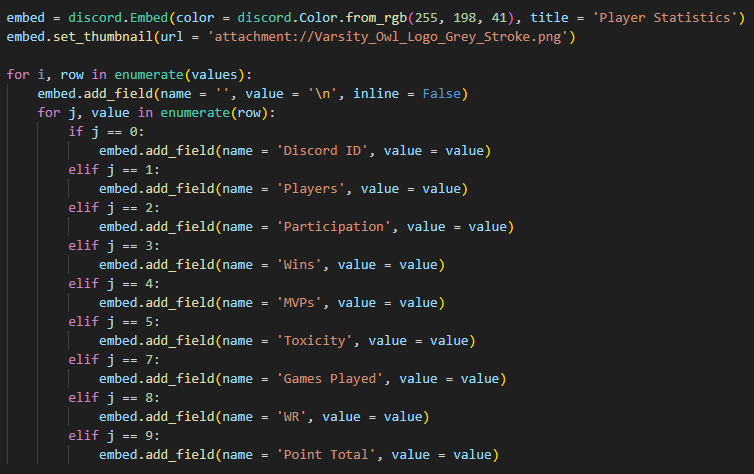
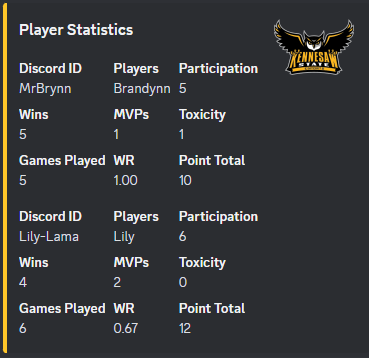


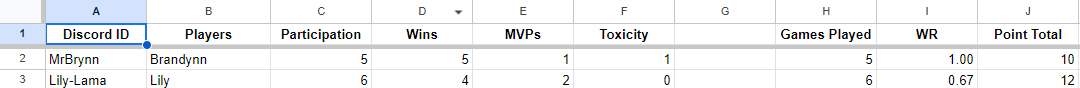
To get started on using Riot’s API so we can gather player statistics, we needed to use a Riot Games account to log into the Developer’s Portal in order to register for a personal API key. From there we had to send an application for a personal key. The personal key fits our project best due the esports being a “small private community.” This key fits our needs best because every second it allows 20 requests and every two minutes it allows 100 requests. (Product registration)

Our key request was accepted, and we easily wrote a method which will take in a username and return the rank and other statistics of the player. These stats can be then used in the matchmaking algorithm.

In order to connect the Discord bot to the Google Sheets API, there were several prerequisites that needed to be done in order to establish a connection between them. For starters, the Google Sheets API needed to be enabled within a Google Cloud project. The next step was to configure consent in the OAuth screen and add a gmail account as a test user allowing it to have permission to access the Google Sheet from the Discord bot. The third step in setting up the API environment was to create credentials that will allow test users authorization. The credentials created then must be downloaded as a JSON file and stored as “credentials.json.” Lastly, Google’s client library must be installed using the command “pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib”. (Python quickstart)

Once the environment has been set up, the code can now be implemented. Google has provided several helpful methods to get started on reading and writing to Google Sheets using multiple coding languages. The downside to these methods is they are only quickstart ones, so it gets us connected to the API but getting it to read and write how we need it requires extra code. (Python quickstart)

To get the bot to display an output that was readable and visually appealing, the data that is read from Google Sheets is displayed using Discords embeds. For Milestone 2, the information was displayed in an embed for demonstration purposes and will not be coded the same for Milestone 3. For the purposes of a demonstration, the bot used a nested for loop to add a field to the embed when it read a new cell from Google Sheets. It also added a new line (‘\n’) field to add a little extra space after it read through an entire row. (API reference)

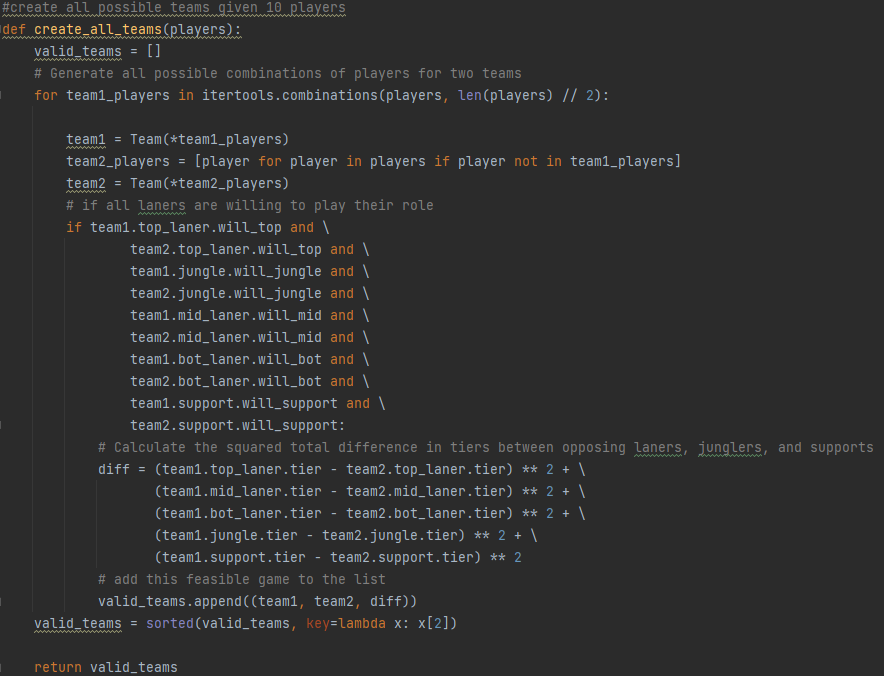


In conjunction with working on the Google Sheets reading and writing, the matchmaking algorithm has been worked on. Players will get a tier from 1-6, this will be primarily based on the players rank found from the Riot API.This matchmaking method shown below will take 10 players which consist of name, tier, as well as their preferred roles (top, jungle, middle, bottom, support). These role preferences are likely to be changed, they are currently a true or false, but may become a 1-5 priority system, where each player ranks their preference of roles. Currently the algorithm will loop through all possible player combinations, splitting them into two different teams. It will then check if all the players are in one of their preferred roles. If all players are, it will then assign the game a score, which is calculated as:

score=(t1top laner​ − t2 top laner)2 + (t1 mid laner ​− t2 mid laner)2 + (t1bot laner​ − t2 bot laner)2 +

(t1 jungle​− t2 jungle​)2 + (t1 support ​− t2 support​)2

Once every feasible team is given a score it will then sort them by the lowest score to the highest. The lower the score the more balanced the teams. We can then pick from the most balanced teams to choose which games to play.



**Project Planning and Management Summary**

Our project was broken up into 3 milestones. The first milestone was divided into 3 parts. The first part was to research information needed to create the bot, develop the commands and buttons, and prepare all documentations for submission.

To get us started on the research, we decided on using Python for the bot’s code. With this, we could get started on researching the documentation on how to develop the commands and buttons using Python. Discord.py provided a lot of helpful documentation on developing the bot because it explained the purpose of the keywords and how to use them (Welcome to discord.py). After doing research on Discord’s API library, we moved on to developing the bot. We created two commands; a check-in command and a re-check/volunteer command. The check-in command allows players to both check-in and leave games and the re-check/volunteer command allows players to volunteer to sit out of games and re-join games. Our main goal for milestone 1 was to have the bot be able to be interacted with by using commands and buttons.

For milestone 2, we started off doing some additional research, particularly on Riot’s API and Google Sheets API, which are two API’s that we needed to utilize for this stage of the project. For the Riot API, there is a developer portal that has a lot of useful information, like how to properly use the API, information on API keys and how to receive one, error codes, and rate limits (Riot Developer Portal). Google also has a developer portal with an overview on the API and how to use it. There’s a lot of good information on it that was very useful for us like creating spreadsheets, reading and writing to spreadsheets, and updating spreadsheets (Google). After that, we went right into working on creating the code that will allow the bot to read and write to spreadsheets. We also decided to get started on the matchmaking algorithm, which is something that is meant to be a requirement for milestone 3, but we decided to get started on it during milestone 2. Having a demo for the matchmaking proved very useful, as it allowed our sponsor to offer suggestions which will go into milestone 3. While we were mostly successful when it came to milestone 2, we did run into some challenges, particularly when it came to having the bot read and write to spreadsheets. While we were overall successful with getting the bot to read to Google Sheets spreadsheets, we ran into some trouble with getting the bot to write to spreadsheets and couldn’t get that method fully working before the milestone 2 due date. Also, Discord’s embeds, which is what we are using to display teams after matchmaking, is also pretty limiting since the field limit is lower than we’d like it to be. Overall, even with the setbacks, milestone 2 was mostly successful.

# **Team Reflection on Project Experience**

# **Appendix**

# **Bibliography**

*API docs for bots and developers. Discord Developer Portal. (n.d.). https://discord.com/developers/docs/interactions/application-commands*

*API reference¶. API Reference. (n.d.). https://discordpy.readthedocs.io/en/latest/api.html?highlight=embed#embed*

*Creating a bot account*. Creating a Bot Account. (n.d.). https://discordpy.readthedocs.io/en/stable/discord.html

Google. (n.d.). *Google Sheets API Overview*. Google for Developers. https://developers.google.com/sheets/api/guides/concepts

Google. (n.d.). *Python quickstart | google sheets | google for developers*. Google. https://developers.google.com/sheets/api/quickstart/python

lykn. (n.d.). *A Gist which shows/tells you how to make buttons using discord.py v2(version 2.0.0a)*. Gist. https://gist.github.com/lykn/bac99b06d45ff8eed34c2220d86b6bf4

*Product registration*. Riot Developer Portal. (n.d.). https://developer.riotgames.com/docs/portal

Welcome to discord.py. (n.d.). https://discordpy.readthedocs.io/en/latest/