Penn State Sports Discord Bot

The Pennsylvania State University

College of Information Sciences and Technology

IST 440W Capstone Project

Final Report

By:

Justin Kelly, Evan Law, Peyton Leineweber

Abstract

This report will focus on the conception to implementation of our Penn State Sports Statsbot on discord. Our sports statistics bot will benefit all members of Penn State athletics from players to coaches and everyone in between as our bot should help drive traffic to different athletics events especially the smaller sports. Our Discord bot will also interest Penn State students who wish to watch and follow the various athletics programs at the university. Lastly our bot may interest other students at universities around the US who wish to have a similar bot for their own school.

We developed a discord bot that looks to provide easy access to information regarding Penn State sporting events and statistics. With the growing prevalence of Discord use across all aspects of life, including the creation of official Penn State Discord servers for a variety of different clubs, activities, and even a general school Discord for regular use, Discord bots that perform useful functions are becoming more and more popular.

Our sports statistics bot will aim to provide scheduling information, recent results, player/team statistics, and more. Our project approach will consist of using Javascript in order to build out our bot and a Javascript web scraping tool to obtain and parse the necessary data for our bot to use. The bot can be accessed through Discord.com and the code can be found on github. We have done an abundant amount of tests throughout the project to ensure its complete success.

Throughout the project's lifecycle we have run into constraints such as time, information availability, and lack of knowledge on a topic. We hope to bypass these constraints in future work by enhancing the user interface, becoming more knowledgeable in web scraping, and completing more real world testing.

Table of Contents

List of Figures

List of Tables

**Introduction**

Problem Statement

The purpose of our semester project is to garner more traffic to Penn State athletics, specifically the lesser known sports by collecting and reporting statistics on them so that they are easily accessible for all fans. Our team hopes to promote greater equality in the information available to fans for all Penn State athletics teams big or small.

Motivation

While forming our group for the course project we found a common interest in sports, specifically at Penn State. We agreed that the information surrounding Penn State’s athletic teams is not equal across the various sports. Some athletics are constantly in the spotlight, and popular sports such as football and basketball receive much more attention compared to the rest of Penn State’s 29 Division I athletic programs. Our group has been motivated by this, and we aimed to highlight all sports in an equal way to generate more attention for all of the athletics at Penn State.

Objective

Our specific objective is to develop a Discord bot that displays information of all 29 Division I sports at Penn State through an open source solution.This solution will drive more traffic to all of the Division I sports that Penn State has to offer.

End User Needs

Our targeted end users consist of Discord users who have an interest in Penn State athletics, whether they be current students, alumni, or just fans of Penn State. Our users may have a difficult time finding the information they are looking for on all of Penn State’s 29 Division 1 athletics programs using the current solutions available (namely the gopsusports.com website and the Penn State athletics mobile app). Our project seeks to provide the wanted information for Penn State fans over Discord in a simple and understandable format to help increase traffic to all Penn State sports.

**Literature Survey**

Literature Review

Penn state is known for its extensive athletic department, with a total of 29 division I sports teams both men’s and women’s (PSU admissions). Each year Penn State fans flock to State College to attend sporting events or they tune in through their tv’s. Football alone had an average of 105,678 fans attend each game at beaver stadium in the 2019 season (Football Foundation). This average ranked second in all FBS schools that year. Another popular sport at Penn State, wrestling, had a total of 15,995 fans attend one of their most anticipated matches that same year (Pilcher). Since the Penn State community revolves around sports it’s important for fans to be able to access extensive information and statistics relating to every team. While most fans can easily follow and watch the more popular sports like football, hockey, or wrestling there are so many more sports that Penn State fans can support.

Our group wanted to utilize all of the information on the 29 Penn State sports teams and b make it easily accessible to all fans. We decided to use discord as the preferred platform to help spread the information because it is popular amongst many students as it is a form of social media.

The popularity of Discord, a free to download voice over IP software platform, has been increasing since its release in May 2015. As of 2021, there are 150 million monthly active users. Discord is home to countless communities including sports fans, gamers, study groups, extracurricular clubs, and software developers. The service is extremely popular with college students, due in part to the fact that anybody with an account can create or join a Discord server.

Bots on Discord have been developed and utilized since the first one was previewed in 2016. Users are able to integrate bot accounts into their servers for various tools and purposes. Common functionalities of Discord bots include playing music, server moderation, user management, minigames, data logging, chatbots, and much more. Developers are able to access the Discord API to design and create applications that leverage its technology.

The goal of our group’s capstone project is to create a Discord bot that can access and display Penn State sports statistics upon user command. We plan to use Discord.js, a Node.js module that allows for easy interaction with the Discord API. This library is based on an object-oriented approach and is flexible enough so we can utilize it to implement our bot’s features with the newest versions of Discord.

Assessment of Available Solutions

Currently Penn State athletic fans can keep up with each team on the gopsusports website. The site has everything ranging from statistics, rosters, schedules, ticketing and more. People are able to click through all of the teams to see their schedule and rosters which is helpful when trying to get to know the sport. The ticketing section does not feature all of the sports only the most well known ones that Penn State offers. While it can be a useful place to find information, it is overwhelming with all the information that they feature on the site, and it is hard to find specific statistics for the lesser-known sports.

Penn State athletics has additionally developed their own app that almost mirrors the website. All student ticket holders are required to use the app for managing and viewing their tickets to events. This is the main use of the app and almost all students have to utilize it if they want to attend a game throughout the year. The app has been known to have significant issues with its ticketing feature as it often locks students out of their accounts and just won’t work entirely. These things make it so it is not as much of a reliable source as the standard website. All Penn State fans should be able to have easy access to their favorite teams and the opportunity to learn about other sports the school offers.

While these two things are offered directly from Penn State there are other places to get information on the sporting events and teams such as the NCAA official website. They keep all the statistics and scores on every NCAA team. This website is useful, but since it is used for every sports team in the country it can be harder to find specific information.

Pros and Cons

After concluding that we wanted to focus mainly on the statistics of Penn State athletics teams we brainstormed potential ways we best show them. We landed on creating a Discord bot from scratch that would gather the statistics and information on all the teams in one organized and easy to access place. Alternatives like the Penn State athletics website and Penn State Go mobile app both provide accurate information, but they lack accessibility and clarity. The athletics section of the Penn State Go app takes the user to the athletics webpage, which can be hard to navigate on a mobile device. We are able to integrate some of the features into our current project such as the statistics that the gopsusports website features on their page for each sport, the calendar of events, and the links to each section of the website.

We believe that a Discord bot is the best way to create our solution because they are easily accessible to anyone with internet access, and we can develop it into our own specifications. After researching how to create Discord bots we found three techniques that we wanted to analyze as potential ways to develop the bot: Java Script, Python, and web scraping. Each of these techniques offer many pros and cons.

Python was the first technique we found to create a Discord bot because it seems to be the most popular and common way to create one. A benefit of using Python is that it has the most resources and similar projects available that we can reference throughout the project, which will be useful since none of us has built a bot before. Another benefit to using Python is that the team knows a few people who have created Discord bots using this language, so they can give us tips if we get stuck on the project.

​​

There are a few cons with using Python to build the bot, the main one being that none of us can code in Python. The references we found show detailed step-by-step instructions on how to build the bot, but it would still be a steep learning curve. This con is the main concern because if we chose to use Python, we will have to put most of our time in learning the language rather than ensuring we include all the elements we want in the bot.

The second technique we were considering was using web scraping to collect the data from other websites. A benefit of using web scraping is that it is a relatively straightforward method to collect data, so this would be an easy technique for us to utilize. With this technique, we are able to use the statistics provided on the gopsusports website to help us get the correct information with the web scraping.

A con of using web scraping is that there can be issues with data integrity and information that we do not want could still be scrapped from the websites. This would give us too much data and not have it be specific enough for our goals.

Lastly, we found that a Discord bot could be developed using javascript. This solution provides us with the most benefits because we are most familiar with this language and there was an abundant number of resources we found to help us along the way. We found multiple articles and videos that detail how to create a Discord bot using Javascript. Since we all are most familiar with this language, we will be able to get the most out of the bot we create. The only con is that this is not the most popular way to create a Discord bot.

**Requirement Specifications**

Market Requirement Analysis

Currently, there are two options available for individuals to access Penn State athletics information. An individual can use Google to find recent results that come from Penn State’s athletics website, or they can download the official mobile application. When someone is looking for statistics data on Penn State sports, it is uncommon to see information for the less popular sports. Football, hockey, and basketball tend to get the most coverage on the available platforms, and as a result they are the easiest to find data for. The options that Penn State currently provides for athletics has many pros and cons that our team has considered throughout the project.

Our project creates another option for individuals seeking this information. Our team believes this solution will be beneficial to Penn State sports fans because of the growing popularity of Discord. Many Penn State affiliates organizations utilize Discord and would likely be interested in integrating this bot in their server for fast and easy access to athletics statistics. The sports market has always been large, and this solution makes information more accessible to a large number of users.

Design Requirement Analysis

When designing our bot we had to take two main requirements into consideration to ensure it worked for our intended use cases. For our project this meant designing how our bot will collect user inputs, and how our bot will obtain the proper information requested from the user, and how our bot will output this information in an understandable way.

The first thing the design required was our Discord bot framework to not only obtain user input but also to display the bots output for the user. This would need a simple command layout for users to input their desired commands to our bot, along with commands that explain our bots full capabilities and how to use them. This design also needs an easily readable output for the user's commands so that the user can easily identify and understand which output contains the information they are looking for and read said data.

The next design requirement will be a way to take the user's input and web scrape the correct information from a reputable website or websites on the internet. The only feasible way for our application to obtain the information real time and be able to obtain future information for every sport is through a web scraping application which makes this perhaps the most important aspect of our design plan.

Constraints

An important constraint is the data that we hope to access for the success of our project. We will need to have the data accessible in a timely and accurate fashion to display statistics to our project’s users. Another constraint is the number of user interactions we will have to account for. We plan for our project to support multiple sports, and there will be many commands that a user can enter to receive the various statistics they are looking for.

Our project will utilize multiple application programming interfaces (APIs) to collect and implement data for the statistics. The navigation and management of these APIs is another constraint of our project. The bot will need to properly implement the Discord API, Node.js APIs for web scraping, and work with Penn State’s official athletics website.

Assumptions

One assumption being made is that the statistics will be uploaded by Penn State shortly after a sporting event. Users of a sports statistics bot will most likely check a score or statistic not long after a game or match has concluded. If we cannot get the most recent data right soon enough, then this could be another constraint. Another assumption we are making is that our bot will be allowed to hook into official Penn State Discords. A third assumption we are making is that Penn State athletics will continue to field the same number of Division 1 teams going forward and not add further teams or drop current teams from competition. The last assumption we are making is that Discord will continue to remain a free-to-use and open source platform, including its bot API.

Risks

A risk our project faced is the fact that many fans may not be familiar with Discord and how to access it, so our project could only be seen by a niche group of people. This would defeat the purpose of our project because the goal is to spread more awareness of all of the sports at Penn State. Almost all fans know about the current offerings, gopsusports.com and the Penn State Athletics app because people mainly access the ticketing website/information through them. The risk is that fans would not be open to a new source of information because they already have two sources.

Another risk is that Penn State or the NCAA will not post any or all of the statistics for some of the lesser known sports right after the games. This could hinder our bot because it is looking to collect information for users shortly after the games. The last risk we currently face is that the bot could malfunction and collect unnecessary information that could confuse the user. This may make our bot seem unreliable and potentially deter individuals from utilizing it to get information.

**System Development**

Concept Generation

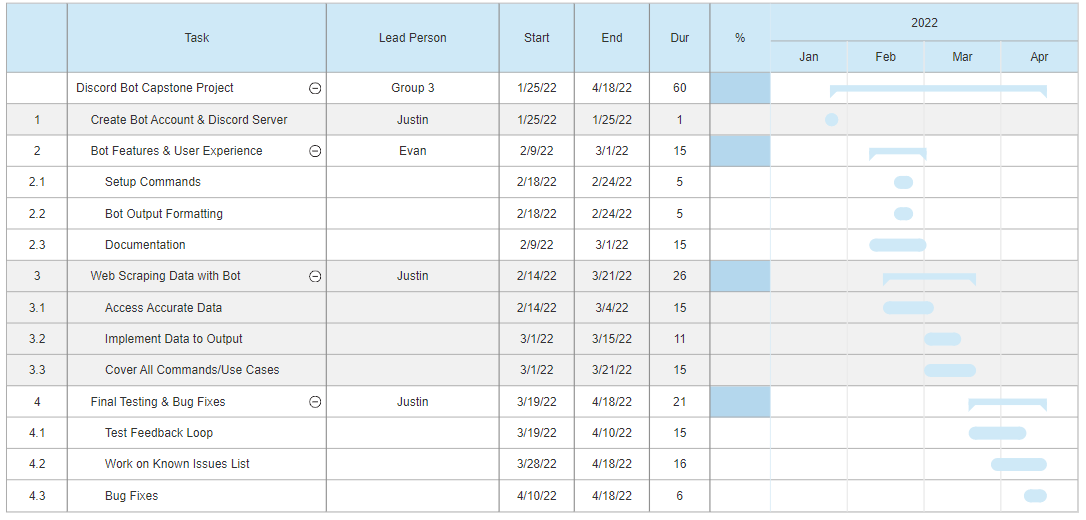
Our team brainstormed many ideas for our project. We first started out with listing topics each of us had interest in and then we found commonalities. The one major thing that we had in common was our love for sports, more specifically Penn State sports. Each year we all follow all of the big sports at Penn State, football, basketball, and hockey. We look out for stats on the players, win-loss records, and our current rankings. This got us thinking how the school has in total 29 Division I sports to offer. While we are all avid Penn State fans none of us knew that there were this many teams at the school with things like fencing, golf, and tennis among them.

All of the sports at Penn State deserve equal coverage, so they can get a chance to become one of the schools big sports. After doing some research we found it hard to find information and statistics on all of the sports. There was plenty of coverage on the main sports to be found on the official Penn State website.

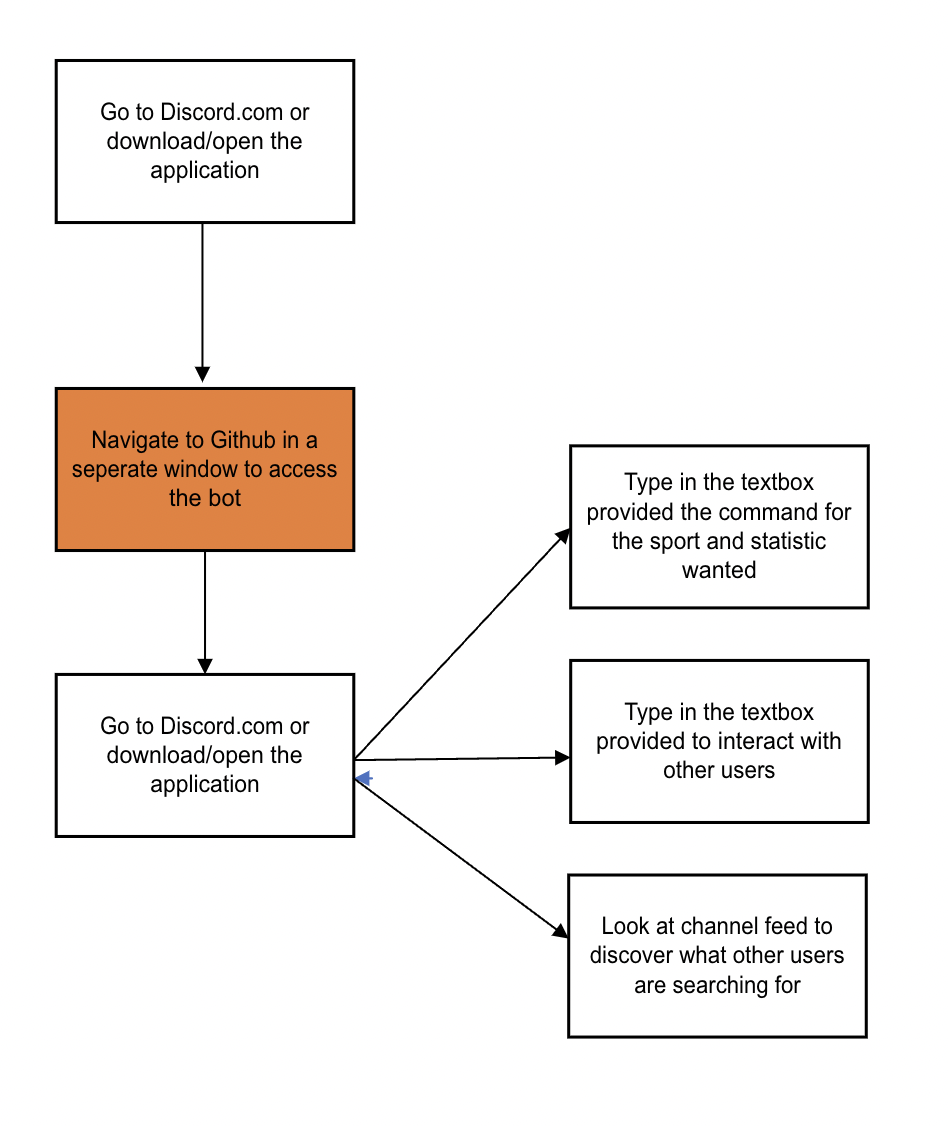
With all of this information now collected, we decided that it would be beneficial to fans and Penn State Athletics to create an easier and more concise way to find information on all of the teams. We wanted to utilize the popular platform discord as many Penn State students use it and many others are active on the site. Our solution was to create a discord bot that helped give out information on the teams. It would use web scraping to collect the relevant statistics and event information, so discord users can easily find and chat about upcoming events.

System Planning

After we decided to select the discord bot as our project we generated a Gantt chart and other flow charts to ensure that all of the requirements for the project were being met. We broke down the Gantt chart into four easily identifiable phases: Creating the bot and discord server, adding bot features/user experience, begin web scraping, and final testing and bug fixes. This breakdown ensured that we were completing the necessary items within our time constraint. Throughout the project we updated this chart to stay on track and we utilized other flow charts to map out or project.

Gantt Chart 

Operational Flow (below)



We hoped to design the Penn State Sports StatBot in a way that can be operated intuitively by someone who is familiar with Discord and those who are new to the platform. The steps to utilize the Discord bot include adding it to a server through the official Discord website, then getting the bot online (either running on the local machine or once we have the bot on a web hosting platform), and once the bot is online any user can type various commands to receive information that is included in the bot’s outputted replies. A complete list of input commands can be found by typing the “!help” command in Discord, and this list is also presented on the [GitHub page](https://github.com/justinkelly8/PennStateSportsBot) for the bot.

Design Process

When designing our application we had to consider three main components, how we would design our front end and middleware, how we would design our backend, and how we would handle API requests. While we only had one realistic option for our front end and middleware, we had a variety of options for developing our backend. We decided early on in our development process what we wished to use for each component to help guide our efforts.

Our front end will need to utilize the Discord API through a Discord bot to interact with users, receive inputs from them in the form of commands, and display outputs. The Discord API runs mainly through Javascript which meant for our frontend we were required to use javascript for our development.

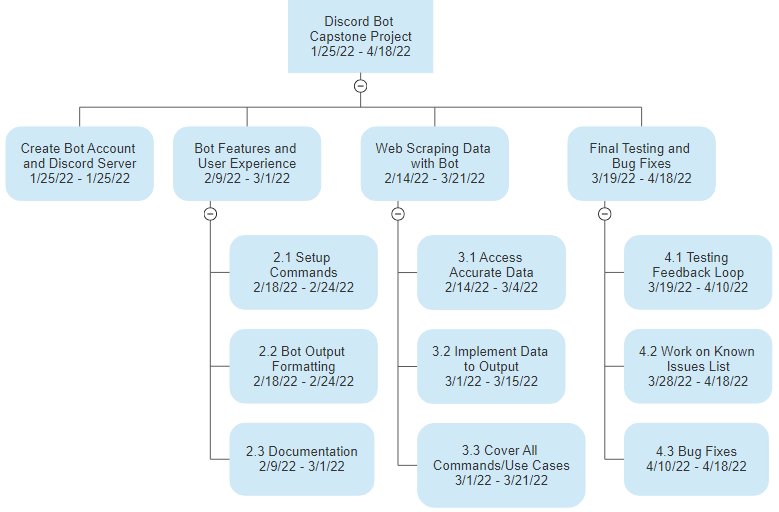
We had many options to design the backend of our project which handled all of the web scraping, python and javascript can both be used to develop a web scraping backend to hook into the Discord Bot API.

System Design

For the front end we used the Discord recommended javascript base with their proprietary Discord Bot API. For our backend we decided to use javascript as our group had the most experience using the language. We specifically decided to use NodeJS for our runtime environment. We used three main Javascript libraries to create our web scraping functions, Puppeteer which is a library that provides a high-level API to control headless Chromium (an open source web browser), Axios which is a promise based HTTP client that makes HTTP requests from NodeJS, and Cheerio which parses markup and provides an API for traversing the resulting data.

Work Breakdown

Figure 0: Work Breakdown Structure



Front End & Middleware

The front end design is primarily focused on one component- the user’s interactions with the bot while in the Discord server. In the design, switch statements are used to evaluate user inputs and execute the statements associated with the case. The front end development includes many customized outputs for each case, which involved sending messages to users based on the feature of the bot they were interacting with.

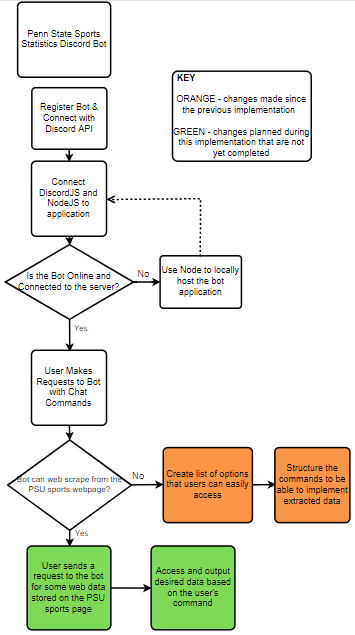
Back End

The back end of our bot consumes the Discord API to access features of the Discord platform. Puppeteer is a Node library which we were able to utilize to make API calls to various websites. The back end data accessed by the Penn State Sports StatBot is retrieved once a user enters a specific command. When these commands are entered, the client application makes a call to a website, scans the value of its HTML element, then outputs the data based on statements programmed into the selected case. The asynchronous functions within these web scraping cases will wait for the declared elements to be found, and then send a message with the output of the value when the function finishes.

How final API requests are carried out

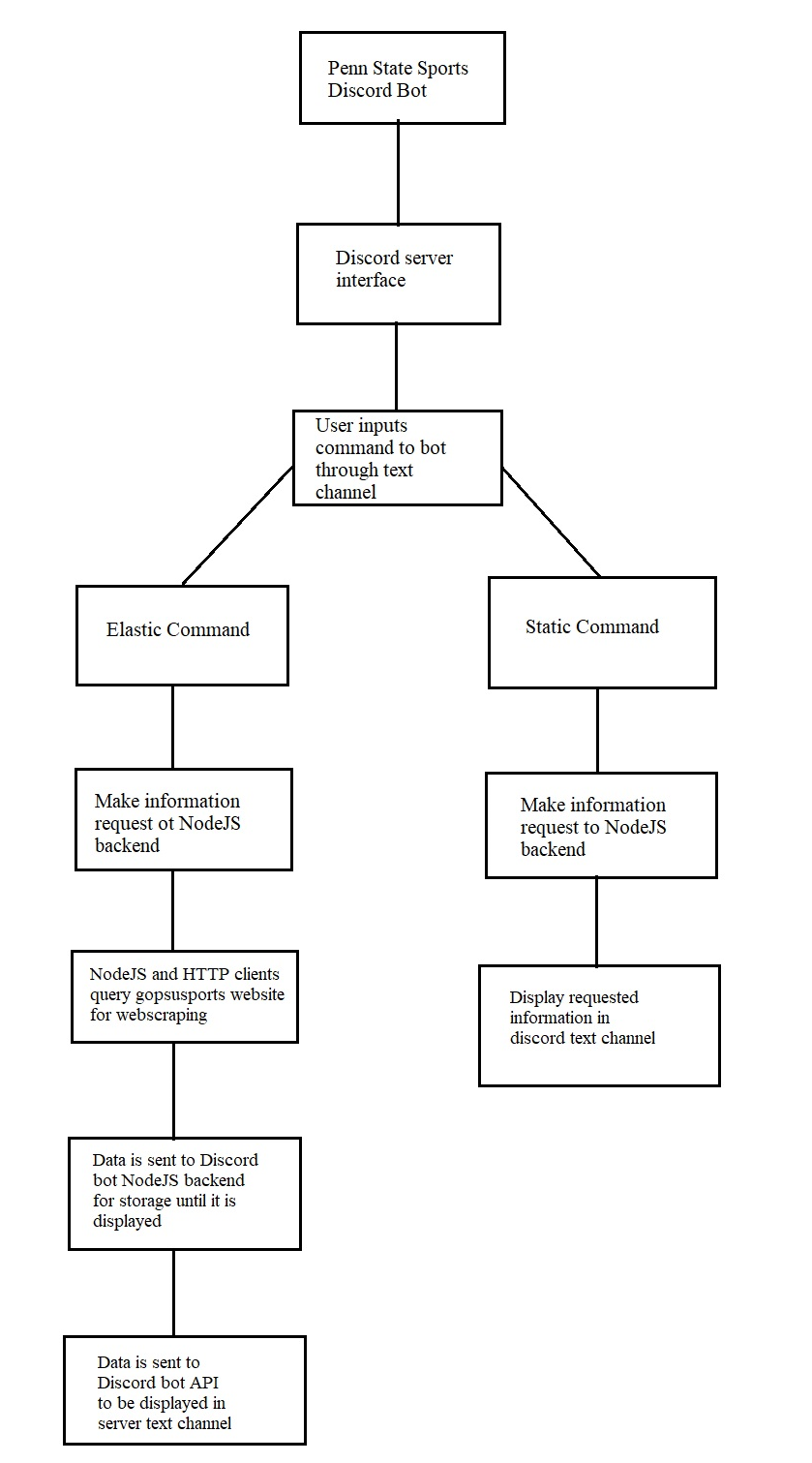
When a user enters a valid command, our bot accesses a web page with Puppeteer. This library provides an API that can run a headless version of the browser, but for our API calls we chose to set this headless feature to false. During the API call, the client opens a browser instance and then awaits the elements of the page that we choose to access for the command. After web scraping from the website the bot’s API call finishes and closes the browser.

Design Flow Chart



Functional Decomposition

Figure 0: Functional Decomposition

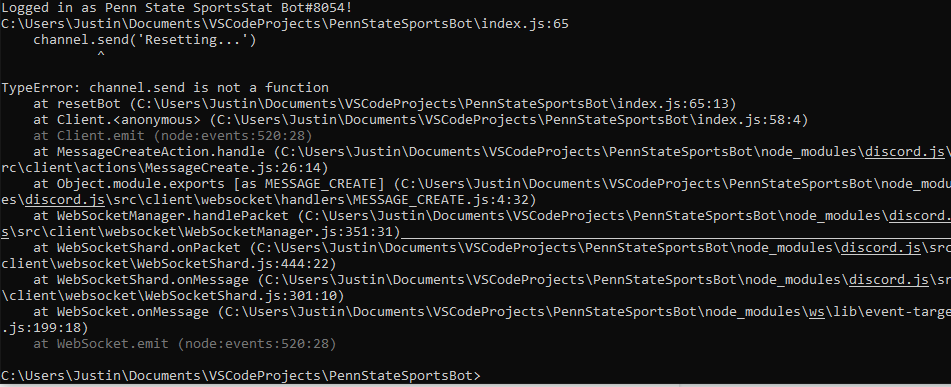


**Testing**

Case One

The first test case involves user input as they are interacting with the Discord bot via chat. This testing case was helpful to determine whether or not the client would be able to stay online after experiencing a particular type of error. The outcome of the test led to the bot’s client crashing and needing to be restarted. By changing the syntax of the code, the function then worked without error. If the bot encounters a command that is not recognized, it does not crash but will not respond to the user.

Figure 0: command interpreter flagging syntax error



Case Two

The second test case arose when we noticed problems with a required project dependency. The package manager for Node.js found and flagged a security vulnerability within one of the imported dependencies. In addition to this, a different dependency could not be found by the client application. We explicitly declared the dependency for the module that could not be found, and then NPM was used to correct both of these package issues. After this testing case, our bot did not encounter any more runtime errors or security vulnerabilities.

Figure 0: Case two dependency security vulnerability

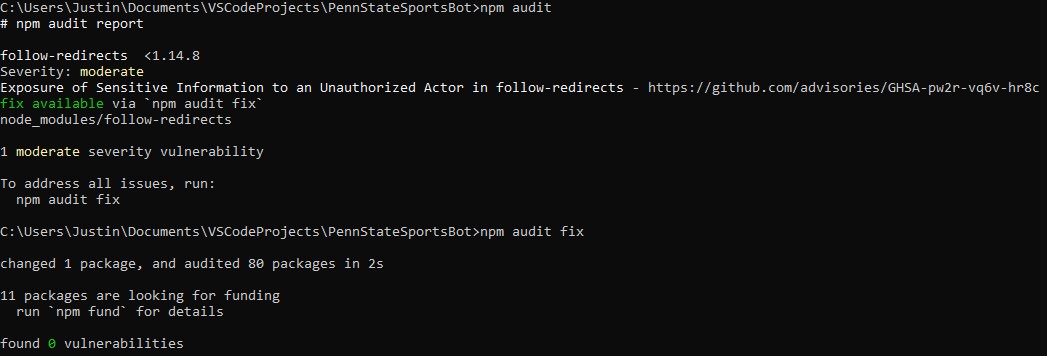
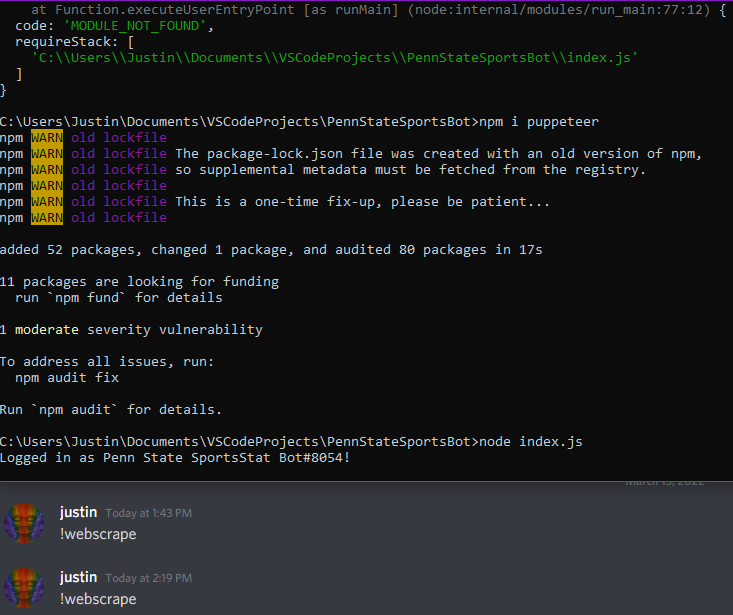


Figure 0: Case two required dependency error



Back End Unit Testing

The back end unit testing was integral in successfully accessing data from the websites. Most of the data came from the official Penn State sports webpage, and some supplementary data came from Sports Reference. In both cases, back end testing was done to ensure we could retrieve accurate statistics at an optimized speed. In each case that accesses statistical data, the bot navigates to one of the website urls, and then gets the back end data via web scraping. Testing this was a significant time constraint, and we hope to continue this testing in the future for the bot’s additional functionality.

Front End Unit Testing

While developing our Discord bot, much of the front end unit testing was done to ensure that the user’s experience with the bot was intuitive and smooth. Without a traditional front end view, like a website or an application page, the main focus of our front end development and testing was on the design of the user-to-bot interaction. Many of the original ideas for the bot’s output were tweaked slightly to improve the user experience. One example of the frontend design testing is the options command, which used to have labels that were hardly descriptive and slightly redundant. A small change to the interface added clarity and consistency which helped improve the experience of bot interactions.

Other Application Crash Avoidances

To avoid application crashes, strict cases were programmed into the bot. If none of the cases are triggered by user input then the bot will not respond in any way, rather than crashing and needing to be reset. Syntax errors were checked during the unit testing phases, and our second test case addressed a package issue that would previously have caused the bot to crash. With the current design, we have established cases for user input that ensure the bot is stable during runtime.

**Conclusion**

The design process for our project focused on the creation of a solution that promotes the spread of information for all Penn State sports. This major goal pushed our group to create a Discord Bot that can provide statistics for any of Penn State’s Division 1 team sports. We utilized Discord’s API with JavaScript frameworks to access this statistical data from Penn State's athletics webpage (gopsusports.com) and Sports Reference (sports-reference.com). This process led to the creation of our Penn State Sports StatBot for Discord.

Changes in development

Initially our team wanted to access statistical data strictly from Penn State’s official athletics website. Unfortunately, we encountered issues with web scraping from the site. Penn State’s gopsusports.com is likely generated by artificial intelligence, which brought challenges in the readability and accessibility of its data elements. Most features of the Penn State Sports StatBot come from accessing the official athletic website and we gathered the statistics for specific games from Sports Reference. With Sports Reference, we were able to more easily find statistics based on the sporting event’s specific date. This solution allows the bot to dynamically provide statistics to the user for a particular game.

Challenges

One major problem our team ran into was difficulty in predicting how long our bot would be to create given our general lack of coding experience within the team. With only one out of three members fully comfortable with coding the others attempted to learn as much as they could over this time period. An example of this challenge was a team member working over a two week period to learn the Puppeteer API within javascript only to not be able to utilize the API as hoped which led to scrapping use of the API to web scrape game scores as originally intended.

Another challenge encountered by the team was the timeframe we had to complete our goal. While we were able to create much of what we originally had set out to, we could not make some of the original functionality we intended to such as a full game stats display for any game the user requests. Another feature we did not have time to implement was the ability to webscrape game scores for all Penn State sports. We could only find a working database to web scrape the scores for Penn State men’s basketball and men’s football. While we could not complete all intended functionality we did have enough time allotted to complete the base of our Discord bot from which we can add whatever functionality we wish to in the future.

One last challenge as detailed above was our inability to use the Penn State athletics website to obtain all of the data we wished to web scrape and display for the user. The website was set up in a way in which web scraping the data was extremely difficult to do, and the API used by the website to obtain the data is private and therefore also not an option. Because of this we needed to find different websites to scrape the data from which left us unable to perform web scraping for a majority of Penn State sports.

Lesson Learned

Our group has learned a lot through the process of this project. Some of the main things we have learned is how to see the project through from start to finish, how to build a Discord bot, and how to web scrape. We began with a topic that we were all interested in and an idea we were mostly unfamiliar with, the Discord bot and web scraping. Our group had some experience using Discord for work and leisure purposes but none of us had ever built a Discord bot. This was a learning curve and took some trial and error to get it working correctly. Our group was successful in learning how to code the bot.

Another thing that we learned was how to web scrape. We used multiple JavaScript libraries including Puppeteer to try web scrape data from sports websites. Learning this process took the most time and we had to use our resources to figure out the correct process. At first we were not successful in getting the web scraping to work, but after continuing to try different things we got it to work. Unfortunately, due to some roadblocks with the website’s HTML, it took us longer than expected to successfully implement the web scraping feature and we were not able to accomplish all of the original functionality we had talked about with the bot.

Overall, We were able to take our idea from its conception to the finish and learned many things along the way from our successes and failures.

**Future Work**

User Interface and Optimization

The next step for our Discord bot is further developing and optimizing our web scraping algorithm to be able to display full game stats for individual games. Currently we can only obtain the score information for games to display the winners and losers of individual games. Our expanded web scraping features would allow the user to obtain full player and team stats for any game they choose. Our team would also like to expand our web scraping functionality to include more sports as currently it can only obtain information for men’s football and men’s basketball. To do this we will need to find databases for the other sports that dynamically upload the results and statistics of games from which we could scrape the necessary data from.

Our team would also like to work towards making our Discord bot more user friendly in regards to the commands which the user inputs to get the different scores and statistics that they want, and with how the bot outputs the requested information to the user. We would like to add a full set of options commands for users so they can see how to input each command and what these commands are used for. Another feature we would like to improve upon is the formatting of our bot’s outputs to make them more visually appealing and easy to understand for the user.

One last piece of optimization we would like to make in the future is optimizing our backend code for faster command processing. Currently some commands take a good bit of time for our bot to process and output, we would like to make our code leaner so that it can process user requests more efficiently for a better user experience. This will likely involve making less complicated loops for our web scraping processes so as to cut down the amount of work it takes to perform each task.

Real World Testing

Throughout the development of our project we have tested the Discord bot many times for general usability and to see areas where we can improve. With the testing we have been able to discover some bugs and errors that we would have not known about if we didn’t perform the tests.

While we have done all of this testing, we have only been able to test with the members of our group for certain scenarios. We would like to expand the testing to our targeted demographic because it will give us the chance to get feedback, so we can improve the bot. The most important part of the project is user satisfaction, so it is vital that our target groups are able to interact with bot without fault. Our future plans include inviting avid Discord users and Penn State fans to test the bot and give us feedback that way we can ensure the best user experience.

Expanding to Other Schools

We believe that our project has potential to inspire similar solutions at other schools. A more generalized version of our project could be something like a Big Ten Sports statistics bot. After our future work concludes, and the bot’s code is more robust, other schools and organizations would have the opportunity to reference our code during the development of their own solution. This can help fulfill our original goal of bringing awareness to all of the sports a school has to offer.

**References**

*2019-20 report: Amazing College football popularity highlighted by impressive ratings and attendance data*. National Football Foundation. (n.d.). Retrieved January 26, 2022, from https://footballfoundation.org/news/2020/5/27/2019\_Attendance\_and\_Ratings.aspx

*Athletics and Recreation at penn state*. Athletics and Recreation at Penn State - Undergraduate Admissions. (n.d.). Retrieved January 26, 2022, from https://admissions.psu.edu/life/athletics/

Pilcher, K. J. (2020, April 24). *Iowa wrestling sets NCAA Season Attendance Record*. Log in. Retrieved January 26, 2022, from https://www.thegazette.com/iowa-hawkeyes/iowa-wrestling-sets-ncaa-season-attendance-record/#:~:text=Penn%20State%20was%20second%2C%20averaging,high%2015%2C995%20against%20Ohio%20State.

*Statistics*. College Hockey | USCHO.com. (2020, March 4). Retrieved January 26, 2022, from https://www.uscho.com/stats/attendance/division-i-men/