Wild Card Bot

Sprint 2 Retrospective
Team 25 - Wild Card Bot
Konstantin Liehr, Arun Thirumavalavan, Nikolas Damalas,
Kyle Arrowood, Michael Gu, Matthew Wang

What went well?

User Story #1:

As a user, I want to access code documentation

#	Description	Estimated Time	Owner
1	Determine how the python documentation website is formatted	2 Hour	Nikolas
2	Create a function to scrape the python documentation website on a requested page	3 Hours	Nikolas
3	Determine how the java documentation website is formatted	2 Hour	Nikolas
4	Create a function to scrape the java documentation website on a requested page	3 Hours	Nikolas
5	Create a module for the bot that will search for a requested item in the python or java documentation and return the documentation information plus a link.	3 Hours	Nikolas
6	Create manual test cases for both python and java documentation	30 minutes	Nikolas

Completed:

The base python and java documentation search works, pulling the first google result related to the corresponding language and search query. In addition, the pydoc command pulls information straight from the python documentation if given the correct package and function names. All of them are hyperlinked to the results if the user wants more information.

User Story #2

As a user, I want to queue multiple songs to be played

#	Description	Estimated Time	Owner
1	Create a queue to hold youtube links for the music module, with automatic resizing	2 Hour	Nikolas
2	Learn how to find the end of a song in the bot and then Create loop to play from queue until empty	2 Hours	Nikolas
3	Make the module process the next song during the current song so that it is ready to play once it finishes	2 Hours	Nikolas
4	Create manual test case	15 minutes	Nikolas

Completed:

Using ‘play’ with multiple songs will queue the songs together and play each, with each playing as the previous one ends. If multiple songs are queued, the bot will go ahead and preemptively download the song so there is no wait between songs. However, a double linked list was used instead of a traditional queue because of the need to be able to print from the song list and remove elements from the list in other user stories.

User Story #3

As a user, I want to add and remove songs from the queue

#	Description	Estimated Time	Owner
1	Create function to remove a song from the queue given the index in the list	2 Hours	Nikolas
2	Modify the !play command to add to queue instead of immediately playing the song	2 Hours	Nikolas
3	Create manual test case	15 minutes	Nikolas

Completed:

The removeSong command works by giving the index of the song in the queued songs list. If the index is invalid it warns the user and does not cause any errors. Using play with a song will automatically queue it up if a current song is already playing.

User Story #4

As a user, I want to view the queue

#	Description	Estimated Time	Owner
1	Create function to list out the queue with index numbers for reference	2 Hours	Nikolas
2	Create manual test case	15 minutes	Nikolas

Completed:

Using the queue command will print out a list of all the songs in the queue. At the top the currently playing song is displayed, and below each song is printed in order with an index number next to it as a reference in case a user wants to remove that song later.

User Story #5

As a user, I want to clear the queue.

#	Description	Estimated Time	Owner
1	Create a function to clear all songs from the queue	2 Hours	Nikolas
2	Create manual test case	15 minutes	Nikolas

Completed:

Using the clear command will clear all songs from the queue, but not disrupt any currently playing song.

User Story #6

As a user, I want to skip a song.

#	Description	Estimated Time	Owner

1	Create a function to skip the current song and go to the next one in the queue	1 Hour	Nikolas
2	Create manual test case	15 minutes	Nikolas

<u>Completed:</u>

Using the skip command will end the currently playing song and queue up the next one if there is one. If there is no song playing, then the user is warned that there is nothing to skip.

User Story #7

As a user, I want to get a google search result in the server with a command.

#	Description	Estimated Time	Owner
1	Create a function to send search query to google search engine.	1 Hour	Kyle
2	Create a function to display information received from google.	1 Hour	Kyle
3	Create an option to search for images.	1 Hour	Kyle
4	Create an option to search for videos.	1 Hour	Kyle
5	Create manual test case	15 Minutes	Kyle

<u>Completed:</u> To keep all of the google commands simple to use, I added all of the different functionalities of the google module with just the keyword !google. The bot will then perform based on the arguments given. I used web scraping to accomplish most of the tasks here, but had to use another youtube specific API in order to implement the google videos search. The bot now correctly displays images videos or just a search depending on what is asked.

User Story #8

As a user, I want there to be an option to jump to the search in my browser from an extra command.

#	Description	Estimated Time	Owner
1	Create a function to display a link to the search.	1 Hour	Kyle
2	Create a function to display links of the top results when requested.	1 Hour	Kyle
3	Create manual test case	15 Minutes	Kyle

Completed: As a part of the google module, the bot when given an integer argument in the form !google <int> <query> will give the user the top n results where n is the integer given. This way users can look at search results from within discord.

User Story #9

As a user, I want there to be an option for the bot to give me a short answer rather than extra unneeded information.

#	Description	Estimated Time	Owner
1	Create a function to display the google short answer to the user.	2 Hours	Kyle
2	Create a function to create a short answer for the user when google doesn’t provide one.	4 Hours	Kyle
3	Create manual test case	15 Minutes	Kyle

Completed: Using beautiful soup for web scraping, when finding a short answer for the query, I would scrape the short answer given from google. If there was no short answer then I would scrape the description of the first search result which would alway give a good description of the search.

User Story #10

As a user, I want to be able to run a stopwatch on the server.

#	Description	Estimated Time	Owner
1	Create a function to start a stopwatch functionality. It will start at time equal to zero and count up.	2 Hours	Kyle
2	Create a function to display how long the stopwatch has been running.	3 Hours	Kyle
3	Create a function to reset the stopwatch.	1 Hour	Kyle
4	Create a function to stop the stopwatch.	1 Hour	Kyle
5	Create manual test case	15 Minutes	Kyle

Completed: I created a command called !stopwatch that when given no arguments would dynamically display a stopwatch in discord. The optional arguments that were allowed and performed functions on the stopwatch were “stop”, “unpause”, “reset”, and “delete”. All of these worked the way that they were intended.

User Story #11

As a user, I want to be able to set a timer for a specific length of time on the server.

#	Description	Estimated Time	Owner
1	Create a function to start a timer functionality. It will start at the time given and count down to zero.	2 Hours	Kyle
2	Create a function to display how much time is left on the timer.	3 Hours	Kyle
3	Create a function to notify the user that created the timer when it is done.	1 Hour	Kyle
4	Create a function to stop the timer.	1 Hour	Kyle
5	Create a function that deletes the timer.	1 Hour	Kyle

6	Create manual test case	15 Minutes	Kyle

<u>Completed:</u> I created a command called !timer that is used to create a timer. It takes a float as an argument and that is the amount of minutes the timer will run, for example using “!timer 0.5” will give the user a 30 second timer. The other arguments that were created with the timer were “pause”, “unpause”, and “delete”. All of these worked as intended and the timer displayed the amount of time left dynamically to the user.

User Story #12

As a user, I want to type a command to give a timezone in a certain city.

#	Description	Estimated Time	Owner
1	Create a function that will fetch the timezone of any given city.	1 Hour	Kyle
2	Create a function that displays the local time in the city given	1 Hour	Kyle
3	Create a function that displays the time difference between the city given and the local time zone.	1 Hour	Kyle
4	Create manual test case	15 Minutes	Kyle

<u>Completed:</u> I created a command called !timezone, at first when I was implementing this I was using a couple of different time modules in python to get a location of a given place and then using that to get the timezone, but I changed my mind and figured it’d be easier to just use web scraping for this since all timezones are easily web scraped. This also made it easy to calculate the time difference. The timezone function is very simple and works very well.

User Story #13

As a user, I want to display information and stats from my Riot account.

#	Description	Estimated Time	Owner

1	Determine how the Riot API works and learn how to extract data for a user about champion skill orders, various item paths, matchups, lane/jungle tips, etc.	2 Hours	Konstantin/Michael/ Arun
2	Create a function that displays information about a Riot account.	2 Hours	Konstantin/Michael/ Arun
3	Create a function that displays a user's League profile.	6 Hours	Konstantin/Michael/ Arun
4	Create a function that displays a user's Valorant profile	4 Hours	Konstantin/Michael/ Arun
5	Create a function that displays a user's TFT profile	3 Hours	Konstantin/Michael/ Arun
6	Create a function that displays a user's Legends of Runeterra Profile	3 Hours	Konstantin/Michael/ Arun
4	Create manual test case	15 minutes (each)	Konstantin/Michael/ Arun

<u>Completed:</u> A user can request League and tft profiles using the !league and !tft commands. When these commands are called, the bot will return information on each profile including level, profile picture, and ranks in different modes. The profile functions request data directly from the Riot API on each call and parse the returned data before outputting. Additionally in order to know which tags to add to the commands a !regions function can be called that shows the tags for all of the different regions.

User Story #14

As a user, I want to get statistics on a live game and display a user's match history.

#	Description	Estimated Time	Owner
1	Create a function that displays the match history of a user.	3 Hours	Konstantin/Michael/ Arun
2	Create a function that displays all champion masteries for a user.	2 Hours	Konstantin/Michael/ Arun

3	Create a function that searches and displays information on live League games.	4 Hours	Konstantin/Michael/Arun
4	Create manual test cases	15 minutes (each)	Konstantin/Michael/Arun

<u>Completed:</u> A user can request information on a live League match using the !live call. This includes all champions, usernames, summoner spells, and runes taken by each player. A user can also request champion masteries for a player using the !mastery call. This includes mastery points of the top 3 most played champions and total mastery points. The !matchhistory command displays information on the last 5 matches of a player. When these commands are called, the bot requests data from the Riot API and parses the returned data before outputting.

User Story #15

As a user, I want to get relevant information about each champion, recommended builds, and gameplay tips.

#	Description	Estimated Time	Owner
1	Research Item order, skill orders and laning tips for general lane matchups for all 150+ characters (winrates are in the API, but tips and ordering are obtained through playing the game, not found in the API)	6 Hours	Michael/Konstantin
1	Research how to use the database of items and characters	1 Hour	Michael/Konstantin
2	Create a function that recommends items builds for each champion.	6 Hours	Michael/Konstantin
3	Create a function that recommends level-up orders for each champion	4 Hours	Michael/Konstantin
4	Create a list of gameplay tips. Create a function that displays relevant ones for each lane.	6 Hours	Michael/Konstantin
5	Create manual test cases	45 minutes	Michael/Konstantin

<u>Completed:</u> Using the static datafiles from data dragon, we loaded them to the path of the Wildcard bot. Important datafiles such as the JSON file containing each champion information were read in using the pandas package into dataframes. Much of the data was then unnested to easily extract useful information. Most image names of images displayed using Riot functions are pulled from the champion dataframe which are then used by functions to grab them in the datafiles before outputting. The !champ function displays information on a certain champion which includes skill descriptions and images of each skill. The !tips function displays tips on playing with or against a certain champion. The outputted information from these two functions are taken from the champion dataframe. The skill order(!skillorder) and recommended item build(!items) commands were web scraped from the u.gg and op.gg websites respectively. When !skillorder is called, the bot will return the recommended skill order of the specified champion. When !items is called, the bot will return the recommended item build of the specified champion including the starting, core, and movement(boots) items.

User Story #16

As a user, I want the bot to recommend me the jungle pathing depending on lane matchup, jungle matchup, and most popular jungle routes.

#	Description	Estimated Time	Owner
1	Research the most popular first clear jungle routes done by high elo junglers	3 Hours	Konstantin/Michael/ Arun
2	Create a database of jungle routes done by high elo junglers.	6 Hours	Konstantin/Michael/ Arun
3	Research jungle matchups, item builds and jungle paths relative to matchup	5 Hours	Konstantin/Michael/ Arun
4	Create a function that identifies the jungle matchup and assigns a recommended jungle path.	8 Hours	Konstantin/Michael/ Arun
5	Research lane matchups, cc per champ and recommended gank styles and paths depending on enemy and players team	6 Hours	Konstantin/Michael/ Arun

6	Create a function that determines the laners and indicates which path to take based on lane matchups.	8 Hours	Konstantin/Michael/ Arun
7	Create a function that takes #4 and #6 into account and determines the best route.	1 Hour	Konstantin/Michael/ Arun
8	Create manual test cases	45 minutes	Konstantin/Michael/ Arun

Completed: The !topjg command outputs the top five players of a specified champion including links to their profiles. When called, the bot will web scrape data from leagueofgraphs.com and parse it before outputting. A list of jungle champions and their respective jungle clear paths was manually added to the module. The !firstclear command returns the recommended clear path of a specified champion. When called, the bot will search the jungle clear list for the champion and output. The !database command returns a link to an external datasheet containing in-depth information on jungle champions and their clears.

User Story #17

As a user, I want to receive messages in game telling me where to go on the recommended jungle pathing

#	Description	Estimated Time	Owner
1	Research how to privately message a user using discord API and turn on their discord overlay for the game, (giving them a notification for this).	4 Hours	Arun/Michael
2	Message the user during the loading screen of which path they should take as a discord private message dependent on champion, matchup, and lane matchups.	5 Hours	Arun/Michael
3	Automatically turn on discord overlay for League of Legends. Message the user during the game of which camp they should be at (i.e. “Go to Red,	6 Hours	Arun/Michael

	raptors, krugs, Can Gank Top, Reset).		
4	Create manual test cases	45 minutes	Arun/Michael

Completed: The !pmpath command is similar to the !firstclear command. Instead of outputting to a text channel, !pmpath directly messages the user using the function the jungle clear path of a specified champion. It utilizes the jungle champion clear paths list in the module. The !overlay command directly messages a user real time data on jungle first clears. When called, the function will begin outputting messages at the start of a match, guiding users on which jungle camps they should be on. These messages use the Discord overlay function and are displayed in game in the top left or right corner.

User Story #18

As a user, I want to have the bot to randomly generate a number between a range of values.

#	Description	Estimated Time	Owner
1	Research and learn about the usages of the Python random library.	1 Hour	Matthew
2	Create a function that will use the random number function to generate a random number in the range specified by the user if applicable.	1 Hour	Matthew
3	Create a function that will display the picked number to the user.	30 Min	Matthew
4	Create unit tests to make sure the bot is correctly generating a number in the given range.	1 Hour	Matthew

Completed: Created the command !decide that has the following format: “!decide <picks> <number/custom> <args>” where <picks> and <args> are optional. The user can request random number generation by specifying “number”. A function was created that generates a random integer or float within a range determined by the parameters. The user specifies the range via arguments. If no arguments are given, a random integer is generated in the range from 0 to 9. If one argument is given, a random number is generated in the range from 0 to the argument. If two arguments are given, a random number is generated in the range from the smaller argument to the larger argument. If any argument specified is a float, then a random float is generated. Otherwise, a random integer is generated. The random number is then displayed to the user in all cases.

User Story #19

As a user, I want to have the bot randomly pick between a set of options to help me make decisions.

#	Description	Estimated Time	Owner
1	Determine the best format for users to pass in the set of options via arguments.	1 Hour	Matthew
2	Create a function that will parse the options given by the user into a list and randomly generate a number in the list index range to pick an option.	2 Hours	Matthew
3	Create a function that will display the picked option to the user.	30 Min	Matthew
4	Create unit tests to make sure the bot is correctly generating an index and properly parsing the set of options even if the options include spaces.	1 Hour	Matthew

Completed: Created the command !decide that has the following format: “!decide <picks> <number/custom> <args>” where <picks> and <args> are optional. The user can specify custom options by indicating “custom”. Spaces are the default delimiter between custom options, but the user can wrap options in quotes to include spaces in their custom options. Users specify the custom options via arguments. A function was created that parses the arguments given by the user, returning a list of custom options. Another function was created to choose a custom option by randomly generating a list index. This option is then displayed to the user.

User Story #20

As a user, I want the bot to pick more than one of the possible choices or numbers.

#	Description	Estimated Time	Owner
1	Create a function that detects if the user has specified that more than one of choices should be picked.	1 Hour	Matthew
2	Create a function that will generate unique random numbers in a range and return a list of with size equal to the number of choices.	1 Hour	Matthew
3	Enhance the functions for numbers and options to account for multiple picks and display all picks accordingly.	1 Hour	Matthew
4	Create unit tests to make sure that the bot is correctly picking more than one possible choice or number.	1 Hour	Matthew

Completed: Created the command !decide that has the following format: “!decide <picks> <number/custom> <args>” where <picks> and <args> are optional. The user can specify more than one picks by specifying the number of picks in <picks>. A function was created to check if the number of picks given was more than zero and not more than the total number of possible options or numbers. The functions for random number generation and selecting a custom option were modified to accept another parameter for the number of picks and return a list of options or numbers with a size equal to the number of picks. The Python sample function was used to guarantee that all picked options or numbers were unique. The picked options or numbers are then displayed to the user.

User Story #21

As a user, I want to have the bot generate a poll that members of the server can vote on.

#	Description	Estimated Time	Owner
1	Determine the information the user needs to provide in order to create a	1 Hour	Matthew

	poll such as the title and time limit.		
2	Create a class that represents a poll and has variables that correspond with the necessary information.	1 Hour	Matthew
3	Research how reactions work in discord and determine how to implement them with a poll.	1 Hour	Matthew
4	Create a function that given the necessary information for a poll will create a poll object with that information.	1 Hour	Matthew
5	Create a function that given a poll object creates and formats a message to represent to the poll. It will also add the proper reactions so that members can vote on the poll.	2 Hours	Matthew
6	Create manual tests to make sure that the bot is correctly creating the poll and that the members can vote on it.	1 Hour	Matthew

Completed: Created a class called Poll that includes fields for the title, question, options, reactions, time limit, text channel, and other overhead required for poll creation via direct messages. Created a function that takes a Poll object and sends a message to the text channel specified by the Poll object. The message is formatted to represent a poll. It contains the title and question specified by the Poll object as well as the options and which reaction corresponds to which option. The bot will then react with all the reactions so that users on the server can vote on the options. Another function was created that checks if the Poll object has a time limit. If it does, this function will continuously edit the message corresponding to the poll to display the time remaining in seconds for the poll.

User Story #22

As a user, I want to create a poll using direct messages to not reveal the question and topic beforehand.

#	Description	Estimated Time	Owner
1	Research how direct messages are handled by the bot and how the bot can directly message a user.	2 Hours	Matthew
2	Create a command that will indicate the user wants to create a poll and immediately direct the message the user.	1 Hour	Matthew
3	Create a function that will continue to ask users for information such as the poll title, questions, and reactions for the poll using direct message and parse the given information for the creation of the poll.	3 Hours	Matthew
4	Create a function that will send the poll to a channel in the server specified by the user.	1 Hour	Matthew
5	Create manual tests to make sure that the bot is correctly direct messaging the user and obtaining required information.	1 Hour	Matthew

Completed: Created the command !createpoll, where the bot will message the user and provide instructions on poll creation upon calling the command. The user can then directly respond to the bot via direct messages, providing the necessary information such as the title, question, options, and time limit. Upon providing all the necessary information, the Poll object is passed into the function specified in the user story above. In order to achieve this, a dictionary was created in the Main module that keeps track of the last command each user called. When the user directly messages the bot, if the last command requires direct message interaction, it will be called with the direct message content being the input. A dictionary was created that keeps track of all the polls corresponding to each user. A function was created to create a Poll object for the user calling !createpoll and add the Poll object to the user's list of polls in the dictionary. Another function was created to detect if a user has an unfinished poll. If they do, the function will find the field the user was specifying and update it to the value provided by the user.

User Story #23

As a user, I want the bot to announce the result in the text channel the poll was sent to with voting statistics.

#	Description	Estimated Time	Owner
1	Create a function that will count the reactions on the poll after the specified time limit and parse the poll results into a list.	1 Hour	Matthew
2	Create a function that will obtain and parse the list with the poll results and create a message that pings all users and displays the poll results with desired voting statistics.	1 Hour	Matthew
3	Create manual tests to make sure that the bot is correctly announcing the voting statistics in the text channel the poll was sent to after the correct amount of time.	1 Hour	Matthew

Completed: There are two situations in which a poll concludes. If a time limit is specified, it will conclude when the time limit runs out. If no time limit is specified, it will conclude when the user directly messages “done <id>” to the bot, where <id> is given to the user when they have finished creating the poll. A function was created to check if the user directly messaged “done <id>” and conclude the poll accordingly. Another function was created that concludes the poll by deleting the message corresponding to the poll and sends another message to the same text channel displaying the results of the poll. It formats the message with the poll results to include the winning option or options, the number of votes for each option, which user voted for which option, and the total number of votes.

User Story #24

As a user, I want to be able to pause and resume the currently playing music.

#	Description	Estimated Time	Owner

1	Create a function to pause currently playing audio	0.5 Hour	Nikolas
2	Create a function that will resume currently paused audio	0.5 Hour	Nikolas
3	Create four manual tests to make sure both functions work as intended	1 Hour	Nikolas

Completed:

Using the pause command while music is playing will pause it. If the pause command is called on a paused song or no song is playing, it will tell the user there is nothing to pause. The resume command will resume the song, and calling the resume command while a song is playing will tell the user that there is nothing to resume.

User Story #25

As a user, I want to be able to play a song from youtube using the title

#	Description	Estimated Time	Owner
1	Use the google search library to get the link of the most relevant youtube video given a title	1.5 Hour	Nikolas
2	Modify the play command so that it will accept both titles and links	1 Hour	Nikolas
3	Create two manual tests to make sure it works as intended	0.5 Hour	Nikolas

Completed:

Using the play function with a search term/title instead of a link pulls the most relevant search result from youtube and plays it. The song that is selected will be linked as it is added to the queue so the user can see what is about to play based on their search. If a youtube link is given, it will play the youtube link. If a non-youtube link is given, the user will be warned an improper link was provided.

What did not go well?

This sprint we had some issues regarding overlapping libraries as well as issues with the Riot API and accessing files. For our modules that required a google search, some members were using one google search library and others were using another, but both used the same package name which caused many errors until we determined that the overlapping package names and mixed usage was the issue. We chose one and converted the few functions that used it to the correct format and usage.

The Riot API key must be refreshed every 24 hours, and we have no way to update the key in the code automatically. This meant that each time we wanted to make edits and tests, we needed to go back through the process of generating a new API key and push it to our code base. In addition the way discord was accessing the various Riot files differed across computers, so files would load for some team members but not others. After some digging we found that sometimes the discord functions will automatically use the file path given but directly inputting it to the correct variable is more consistent.

We had no incomplete user stories this sprint.

How should you improve?

One thing that we can improve on during sprint 3 will be communication. We felt that during sprint 1 it was easier to have better communication, because the work that we were all working on was very close to each other with different features dependent on each other. During sprint 1 we were building the foundation of the bot so that it could be easily added onto with different features, but during sprint 2 most of us were working on completely different features that did not really relate to each other at all. This caused some minor issues with APIs we were using that clashed since they were using the same names. This leads into another thing we can improve on.

Another thing that we can improve on for sprint 3 is the testing. Although we felt that individually we all did fine testing our own features, it was the issues of running all of the features together at the end for the sprint review where issues arose. These issues dealt mostly with personal setup on each machine, which allowed us to figure them out pretty quickly. The problem was that we were stuck dealing with these issues on the night before the review, so for sprint 3 we should probably make sure we figure these types of issues out well in advance. We did not really procrastinate the sprint as a whole, rather just the testing of certain features on different machines.