Wild Card Bot

Sprint 1 Retrospective

Team 25 - Wild Card Bot

Konstantin Liehr, Arun Thirumavalavan, Nikolas Damalas,
Kyle Arrowood, Michael Gu, Matthew Wang

What went well?

In general, we fully finished the modular function system which is the core feature of the bot. We also finished the administration tools and got a start on the music functions. Meeting three times a week and also meeting with our coordinator weekly has helped ensure we are always on the same page and working towards the same goals.

User Story #1: As a developer, I would like the bot to be able to interact with the Discord API.

#	Description	Estimated Time	Owner
1	Make connection to Discord API	1 Hr	Nikolas
2	Add to the server for testing	0.5 Hr	Nikolas
3	Create a function to generate a list of all of the users of the server.	2 Hrs	Kyle
4	Create a function to get all of the permissions and roles of all of the users of the server.	3 Hrs	Kyle
5	Create a function to create a list to store all of the commands.	2 Hrs	Kyle

Completed: The bot connects to Discord and our test server with no issues. The command to list all users also gives their roles/permissions. The ‘help’ command gives a list of all the commands available.

User Story #2: As a developer, I would like to be able to dynamically load functions to the bot from separate files

#	Description	Estimated Time	Owner
1	Create function to load data from external function files	2 Hrs	Nikolas

2	Create listener to check for commands in chat for all external functions	3 Hrs	Nikolas
3	Determine the format that external functions need to be in to load in correctly.	3 Hrs	Nikolas
4	Create function to send command arguments to the respective function and the execute it	3 Hrs	Nikolas

Completed: We have a command class that holds all the information for each of the commands loaded from external files. The files are imported as modules and then we append the command list of each file to the core command list. Because the modules are loaded into the main bot, we can easily pass arguments and extra information to the commands. A function checks each message to see if it contains a recognized command, and if it does it runs the corresponding function.

User Story #3: As a Server Administrator, I want to add and remove functions from the bot via a function folder.

#	Description	Estimated Time	Owner
1	Create a directory locally that the bot can access labelled “Modules”	0.5 Hr	Nikolas
2	Create a function to reload the function list in the bot when the function folder is modified	2 Hrs	Nikolas

Completed: The modules folder holds all of the external functions for the bot. This folder is what the main bot will check when it wants to load or reload the commands. We implemented multithreading to have a second process that checks the modules folder every few seconds for a change in the present files, if a change is found all the modules are reloaded. THis allows users to drag and drop functions directly to the modules folder if they please.

User Story #4: As a Server Administrator, I want to add functions to the bot using Discord's file sharing feature.

#	Description	Estimated Time	Owner
1	Determine how the bot can access files uploaded via discord chat	1 Hr	Nikolas
2	Create an admin level function to scan for files uploaded via discord and download them to the function folder	4 Hrs	Nikolas
3	Create function to check for command/alias collisions before installing the function	3 Hrs	Nikolas
4	Call the reload function to make sure the commands are loaded	1 Hr	Nikolas

Completed: Users can drag files into their discord window and use the add command as a comment on the upload and the bot will grab the file, check to make sure it is formatted correctly, and then add it to the modules folder. This allows approved users to add modules without having to directly access the modules folder. If the user is not approved the file will not be added and they will be notified.

User Story #5: As a Server Administrator, I want to remove functions from the bot with a command.

#	Description	Estimated Time	Owner
1	Create a function to delete a command file given the name of the module or the filename.	2 Hrs	Nikolas
2	Call reload function to make sure commands are not still accessible.	1 Hr	Nikolas

Completed: Using the del command, users can specify a module or filename and the bot will delete it, given it exists and the user has permission. If neither of those conditions is met the user is notified accordingly. After the file is deleted, the functions are reloaded to make sure the deleted module is not accessible.

User Story #6: As a Server Administrator, I want to be able to edit the list of banned words.

#	Description	Estimated Time	Owner
1	Determine the most efficient way to store the list of banned words.	3 Hrs	Kyle
2	Implement the list of banned words into the main module.	3 Hrs	Kyle
3	Create a function that adds a word to the list of banned words via a command.	2 Hrs	Kyle
4	Create a function that removes a word from the list of banned words via a command.	2 Hrs	Kyle
5	Create a function that displays the list of banned words.	1 Hr	Kyle

Completed: To keep the banned word commands easy to use I created only one command called !bannedWords. When given no arguments it would display the list of bannedWords. When given add and more arguments it would add those words to the banned words list. When given remove and more arguments it would remove those words and notify the user if the word did not already exist in the banned words list. When given the argument clear it would remove all of the words from the banned words list. The banned words were stored in a text document to enable the bot to remember words after being started and stopped.

User Story #7: As a Server Administrator, I want the bot to automatically delete messages containing banned words.

#	Description	Estimated Time	Owner
1	Create a function to detect if a	3 Hr	Matthew

	banned word has been used in a message efficiently.		
2	Create a function that deletes the message with the banned word before it is sent.	3 Hr	Matthew

Completed: From the implementation of banned words, there was a list object that held all the banned words. This was populated upon starting the bot by parsing the text file that stored the banned words. A function was created that checks each message for banned words. It first split the message into words, stripped all punctuation, and then made the words lowercase for case insensitive comparison. It would return a boolean value that indicates whether the message has a banned word. Using this output, an if statement was added to the on_message() function that calls the function to check for banned words and delete the message if there was a banned word.

User Story #8: As a Server Administrator, I want the bot to warn users if they try to use a banned word.

#	Description	Estimated Time	Owner
1	Research key word detection and implementation in python using discord api	2	Arun
2	Create a function to detect which user sent a banned word.	3 Hr	Arun
3	Research withholding/deleting messages specified to user that sent through the use of discord api/permissions	2	Arun
4	Create a function that warns the user of the word that they tried to use and tell them it is a banned word.	3 Hr	Arun
5	Create a function that restricts users from sending messages for a period of time specified by the server administrator when they send too many banned words in a short period	4 Hr	Michael

	of time.		
6	Create a function that allows the server administrator to change the timeout period.	3 Hr	Michael
7	Test Key word detection and banned word functions of bot and fix any implementation problems	1 Hr	Arun

Completed: This user story is closely related to user story 7 and much of it was done through collaboration. In addition to maintaining a banned words list and deleting messages from automatically timed out users, server admins can manually timeout users using the !mute function.

User Story #9: As a Server Administrator, I want the bot to kick/ban users with a command.

#	Description	Estimated Time	Owner
1	Research discord api/permission for kicking and banning users based on certain criteria	2 Hr	Arun
2	Research standard formats for different variations of bans and kicks as a result of different actions	1 Hr	Arun
3	Create a function to kick a user from the server.	3 Hrs	Arun
4	Create a function to ban a user from the server.	3 Hrs	Arun
5	Create a banned list of all users that are banned from the discord server.	3 Hrs	Michael
6	Test kicking and banning functions and fix any implementation problems as well as the different criteria and notification messages along with this	1 Hr	Arun

<u>Completed:</u> Two commands that allow admins to kick a user in the server by specifying their name and their tag number using the format !kickMember and another command to ban a user using the format !banMember. The key difference between kicking and banning a member is that when you ban a member they get on a list of banned users from which an admin in the server has to manually unban before inviting them to the server again, while after you kick a member they can be invited almost immediately again through an invite link. A list of all the banned members from a server are tracked by the bot and can be displayed using the !bannedMembers command.

User Story #10: As a Server Administrator, I want to assign and change user roles with a command.

#	Description	Estimated Time	Owner
1	Research Role creation/deletion/assigning	1Hr	Arun
2	Research standard format for permissions and commands for role related actions	1 Hr	Arun
2	Research Role permissions and api for shortcuts in changing roles through commands	1 Hr	Arun
3	Create a function to create a role with a specific name.	3 Hr	Arun
4	Create a function to remove a role with a specific namer.	3 Hr	Michael
5	Test that the Wildcard Bot is correctly manipulating roles	3 Hr	Arun
6	Test role permission through commands and shortcuts and fix any implementation problems	2 Hr	Arun

<u>Completed:</u> Created a function for listing all current roles in the server whether a user has it or not by using the command !roles. Users and admins can also create new roles in discord using the !createRole command and can similarly delete an existing role with the !delRole command.

User Story #11 : As a Server Administrator, I want the bot to have the ability to remove messages with links in certain channels.

#	Description	Estimated Time	Owner
1	Create a function to recognize links and remove messages with them.	2 Hrs	Matthew
2	Create a command to allow users to choose which channels should and should not be monitored for links	2 Hrs	Matthew
3	Test that the bot can recognize links and only removes messages with links in channels requested.	2 Hrs	Matthew

Completed: A list object was created that holds the channel id values of the text channels that should be monitored for links. Created a function along with several helper functions that would allow for manipulation and traversal of this list, allowing the user to choose which channels they want monitored for links as well as see what channels are currently being monitored. These functions are invoked by the command !removelinks. To view the channels currently being monitored, a user will need one required argument, which is 'view' to specify viewing the list. To add or remove a channel currently being monitored, a user will need two required arguments, which are 'add' or 'remove' to specify the action and the name or id value of the text channel, and one optional argument, which is required if the second argument is a channel id value.

Another function was created to check if a message in a monitored channel contains a link. It checks to see if http:// or https:// is in the message and the text channel the message was sent from is in the list. It would return a boolean value that indicates whether the message has a link and is sent from a monitored channel. Using this output, an if statement was added to the on_message() function that calls the function to check for invalid links and delete the message if necessary. A warning was also sent to the user.

User Story #12: As a Server Administrator, I want to create text and voice channels with a command.

#	Description	Estimated Time	Owner
1	Create a function that can create a	2 Hrs	Matthew

	text channel.		
2	Create a function that can delete a text channel.	2 Hrs	Matthew
3	Create a function that can create a voice channel.	2 Hrs	Michael
4	Test using a Discord server that a text channel or voice channel is properly created	4 Hrs	Michael

Completed: Created a function along with helper functions for creating and deleting a text channel, which can be utilized by the !createtc and !deletetc commands respectively. Another function was created for the creation of a voice channel functionality. This can be utilized using the command !createvc. Commands !createtc and !createvc feature one required argument, which is the name of the channel that is created, and a optional argument, which is the category that the channel should be put under. If the category doesn't exist, then a new category is created. The command !deletetc featured one required argument, which was the name or id value of the channel being deleted, and one optional argument, which is required if the first argument is a channel id value.

User Story #13: As a developer, I want to know what format to code commands in.

#	Description	Estimated Time	Owner
1	Determine the standard format for commands to be written in	2 Hrs	Michael
2	Include the formatting in the README.md	1 Hr	Michael

Completed: Included standard format for commands in the readme file. This includes essential imports and data structures users need to get started with adding commands. An example function is provided which shows the basic syntax of a command. For more examples on how functions work, users can also look in the serverAdministrator.py file.

User Story #14: As a developer, I want to know what languages I can code commands in.

#	Description	Estimated Time	Owner
1	Research compatibility of external modules written in other languages (e.g. JavaScript) with the python base bot.	2 Hrs	Matthew
2	Create a prototype module in each potential compatible language and test how it links with the base bot.	2 Hrs	Matthew
2	Create a function that displays compatible languages and steps needed to do so.	2 Hrs	Matthew

Completed: We wanted Wild Card Bot to be compatible with external modules written in languages other than Python. Upon doing research, I found that it was incredibly difficult to integrate external modules in other languages without severely restricting the functionality, and some languages would be almost impossible to integrate without directly converting it to Python. I created a prototype module which was written in C++ and a module called cppModuleRunner that would run the prototype module. It would take the stdout output from the C++ module and create a message and send it to the user, which was invoked by the !cpp command after specifying the file name and command as arguments. I then created another module called developer that features the !developer command. This command displays the sections of the developer manual, which includes the compatible languages and steps needed to create a compatible Python or C++ module.

User Story #15: As a Server Administrator, I want to know if the function I am adding uses the same command as a currently implemented command.

#	Description	Estimated Time	Owner
1	Create a function that scans the newly added module for commands and compares the commands to the current command list.	4 Hrs	Kyle

2	Create a function that prompts the user telling them that there are overlapping commands.	2 Hrs	Kyle
3	Create a function to rename the commands that were named the same.	2 Hrs	Kyle
4	Create a function to call the functions from external modules with the new given command name.	4 Hrs	Kyle

Completed: Initially this task didn't include creating a command to rename commands, but we went ahead and added one in as it just made sense. The rename command can be called using !rename and two arguments, one giving the old name and the other giving the new name. The main reason that renaming commands was important was because of collisions on loading in modules. A function was created that prompted the user when modules were loaded with colliding command names. When the user would input a new name for the command it would change it in the internal python file so that the collision would not occur again. To call any given command there is a function as a part of the Command class called callCommand() that can be used on any command to call it.

User Story #16 : As a Server Administrator, I want to specify and change which roles can use a command.

#	Description	Estimated Time	Owner
1	Create a function that can limit specific commands to specific roles.	2 Hrs	Matthew
2	Create a function that would allow specific roles to use specific commands.	2 Hrs	Matthew
3	Create a function that will automatically determine which commands a role can use based on permissions.	4 Hrs	Matthew

Completed: A dictionary object was created that has each role's id value as keys and has a list containing the permissions of the role and commands names of commands a role cannot use as values. Created a function along with multiple helper functions that would allow manipulation or traversal of this dictionary, enabling users to limit specific commands or allow specific commands for the given role as well as display the currently allowed commands for a role. These functions can be utilized by the !rolecommands command. To view the allowed commands for a role, the user needs two required arguments, which are 'perms' to indicate viewing the permissions and the name of the role. To allow or block a role from using a command, a user with admin privileges needs three arguments, which are 'allow' or 'block' to indicate the action, the name of the role, and the command that should be allowed or blocked.

Another function was created that automatically determines all commands a role can use based on the role's permissions and the permissions required for each command, which is a variable in the Command Class. It is called every time a user uses a command and traverses all roles on the server. If a new role is found, since it is not in the dictionary, a key value entry is created for it in the dictionary based on its permissions. If the permissions of a role were updated, which can be determined by comparing the permissions stored in the list with the current permissions, then the existing key value entry is deleted and rebuilt to reflect the new permissions.

User Story #17 : As a Server Administrator, I want to know where I can download additional modules.

#	Description	Estimated Time	Owner
1	Research searching functionality and linking external module functionality	2 Hrs	Konstantin
2	Create a command that provides the infrastructure for links to other external modules.	3 Hrs	Konstantin
3	Use a Google Search functionality with the command alongside the links and display top results.	8 Hrs	Konstantin
4	Test that the command using placeholder links as well as the associated Google Search result.	2 Hrs	Konstantin

<u>Completed:</u> Initially there was some issue in implementing the Google Search due to the limited use of it in Discord Bots already, therefore it took some time to choose a method with enough support as some ways had very little developer information, however the current implementation uses the command !downloadModules <search> in which when the user only types the command the project GitHub is linked to the user and if the user wishes to search for modules the search function will search Google Using web scraping and returns the top 5 results to the user as links and embeds.

User Story #18 : As a user, I want to know what commands are available to use with the bot.

#	Description	Estimated Time	Owner
1	Research creation of discord API lists and how to display with command.	2 Hrs	Kyle/Konstantin
2	Create or add on to a command that displays the command list.	3 Hrs	Kyle/Konstantin
3	Tests that the command list required call works as intended and shows an accurate list of update commands.	2 Hrs	Kyle/Konstantin

<u>Completed:</u> The user story was implemented and addressed in the !help function as having multiple functions that served the same purpose would be confusing and pointless for the user, so only one !help function was made to encompass many roles in helping the user. The !help function with these requirements prints out a list of all available commands on the server to the user, stays updated as modules are added to the command list which the user can search by, as well as specific commands, and provides the user the roles for each command that can use that command.

User Story #19: As a user, I want to know what a command does and ways to invoke it.

#	Description	Estimated Time	Owner
1	Research ways to display discord commands using discord API as examples	2 Hrs	Kyle/Konstantin
2	Create or add on to a function that	3 Hrs	Kyle/Konstantin

	can display what a specific command does.		
3	Create or add on to a function that displays how to and different ways to use a specific command.	3 Hrs	Kyle/Konstantin
4	Test that the function call works as expected and also shows examples of commands accurately.	2 Hrs	Kyle/Konstantin

Completed: As with user story #18, user story #19 felt as though it could all fit within one function the !help function and so the !help function with these requirements when typed prints out a list of all of the available commands to the user, the roles that can use that command, as well as *now* for each command having a description of how to invoke it and what its purpose is. If the user wishes to find a specific command or module specific commands, they can search these by adding them as an extra token after the initial command.

User Story #20: As a user, I want to see basic stock information when I provide a ticker

#	Description	Estimated Time	Owner
1	Learn how to use and call the yahoo finance API from python	2 Hrs	Nikolas
2	Create function to grab the ticker from the user and get the corresponding info from the API	2 Hrs	Nikolas

Completed: Using the stock command and providing a ticker, the stock module gives the logo, company name, and current price of the stock in a few seconds. The module calls the yahoo finance api to get this information. If a stock is invalid or not found the user is notified.

User Story #21: As a user, I want to see expanded stock information and get a link with more info about the stock.

#	Description	Estimated Time	Owner
1	Read in the advanced data from the API call and store it in logical groups	1 Hr	Nikolas
2	Create function to display the advanced info based on extra keywords from the user (for example sending the “short” keyword would provide information about shorted shared).	2 Hrs	Nikolas

Completed: Users can use multiple extra tags to get extra information about daily trading information, volumes, long term trading information, shorts, or the business for a provided stock ticker. The module also provides a link to the yahoo finance page if they want more information. Invalid tags are simply ignored and multiple can be used at once.

User Story #22: As a user, I want the bot to be able to join voice channels and play music through youtube links.

#	Description	Estimated Time	Owner
1	Create function for bot to join and leave voice channels	3 Hr	Michael
2	Create function for bot to play music from youtube links	4 Hr	Michael
3	Create function to adjust volume of bot	4 Hr	Michael

Completed: Successfully implemented basic functionality for users to command it to join and leave voice channels. Music can be played through passing in youtube links. The module will extract the mp4 file from the website, convert it to mp3, and then output audio through the voice channel. It will default to the first voice channel in the server to play audio if it is not already connected to one.

What did not go well?

In general, we had some overlap between a few of our user stories and tasks which created some confusion during the implementation process as well as confusion over who should present what when it was time to test our acceptance criteria. We also overestimated the time it would take for some of our tasks so we ended up having to add a few extra user stories to make sure everyone had enough work to do.

We had no incomplete user stories this sprint.

How should you improve?

One aspect our team can improve upon for the next sprint is our time estimations. During the sprint, some team members ran out of work and had to pull extra user stories because we mistakenly overestimated the time many of our tasks would take. This led to team members being at very different places in the project since some tasks were flying by while some were taking the expected amount of time or even a bit longer. Because it was our first sprint and first time working with a discord bot, we did not really have an idea how much time anything would take. We can solve this for the second sprint because we now have experience with the bot and calling external modules and APIs so we can make much more realistic and accurate estimations of how long our tasks will take. Additionally, the modular function system is fully done so we will have no issues testing our new modules as we work on them.

Another thing we can improve upon is making sure our user stories and acceptance criteria do not overlap too much. We had a few user stories that all covered the same idea, but phrased from a different perspective. This led to our help function being the core of a few user stories, and so the acceptance criteria overlapped heavily. This led to some confusion during our sprint review prep as we realized the criteria overlapped. We can solve this by reviewing our acceptance criterion together before the sprint document is submitted and making sure everyone is fully aware of what we are all working on and our goals each week. This will prevent more overlap and ensure we don’t run into the issue again.

During our practice sprint review, we noticed we had entirely missed some edge cases in our modules that neither the user story nor the acceptance criteria covered. This led to us having to do more work than expected on some of our modules in order to make sure the edge cases would be covered and not cause issues with the bot. We can solve this problem by being more thorough during our sprint planning, as well as having each team member fully review the document and everyone’s user stories to see if they catch something that the rest of us missed.