Wild Card Bot

Sprint 2 Test Cases

Team 25: Konstantin Liehr, Arun Thirumavalavan, Nikolas Damalas, Kyle Arrowood, Michael Gu, Matthew Wang

Test Case 0001
System: codeDocs Phase: 2
Python search check
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Type in “!python str isupper”

Expected result

1. Bot responds with an embedded card that gives information on what the string.isUpper() command is in python from google search. Clicking the card takes you to the google search.

Test Case 0002
System: codeDocs Phase: 2
Python Documentation search check
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Type in “!pydoc str isupper”

Expected result

1. Bot responds with an embedded card that gives information on the functionality of str.isUpper from the python documentation. Clicking the card takes you to the documentation page.

Test Case 0003
System: codeDocs Phase: 2
Python Documentation Incorrect Search Check
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Type in “!pydoc xyz xyz” in discord

Expected result

1. Bot responds with a red card saying the module does not exist and is not found in the documentation.

Test Case 0004
System: codeDocs Phase: 2
Java Search Check
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!java arraylist sort” in discord.

Expected result

1. Bot responds with a blue card with information on java arraylists sort capabilities. Clicking the card takes you to the google search.

Test Case 0005
System: googleModule Phase: 2
Default Search Check
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!google Purdue” in discord.
3. Enter “!google When was Purdue founded” in discord.

Expected result:

1. Bot responds with a green card that will give a description, or if a question is asked will give an answer. In both cases there will be a link to the search results.

Test Case 0006
System: googleModule Phase: 2
Image Search Check
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!google images Purdue” in discord.

Expected result:

1. Bot responds by sending the user the top 8 images from the google images search.

Test Case 0007
System: googleModule Phase: 2
Video Search Check
Severity: 2

Instructions

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!google videos dogs” in discord.

Expected result:

1. Bot responds by sending the top 5 youtube results with hyperlinks to videos.

Test Case 0008
System: googleModule Phase: 2
Search Results Check
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!google 20 Purdue” in discord.

Expected result:

2. Bot responds by sending the user the top 10 search results from Google.

Test Case 0009
System: music module Phase: 2
Song queue check
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Join a voice channel in discord
3. Enter “!play <valid youtube link”
4. Enter “!play <another valid youtube link”

Expected Result:

1. The bot will join the voice channel
2. After each call of “!play” the bot will notify the user that the song was added to the queue.
3. The bot will play both youtube videos in the voice channel, one at a time.

Test Case 0010
System: music module Phase: 2
Song missing input
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Join a voice channel in discord
3. Enter “!play” in chat

Expected Result:

1. The bot will respond and say “No video link provided!”

Test Case 0011
System: music module Phase: 2

Song bad input
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Join a voice channel in discord
3. Enter “!play https://abc123.com” in chat

Expected Result:

1. The bot will respond and say “Error in youtube link provided!”

Test Case 0012
System: music module Phase: 2
Testing list queue function
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Queue a song with “!play <url>”
4. After the song starts, enter “!queue”

Expected result:

1. The bot will list out all queued songs, including the one currently playing.

Test Case 0013
System: music module Phase: 2
Empty queue list
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Type “!queue”

Expected result:

1. The bot will respond saying the queue is empty, since no songs have been added.

Test Case 0014
System: music module Phase: 2
Skip function
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Queue a song with “!play <url>”
4. Queue a second song with “!play <url>”
5. Enter “!skip”

Expected result:

1. The bot will stop playing the current song, and after a few seconds it will play the second song.

Test Case 0015
System: music module Phase: 2
No song to skip
Severity: 1
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Type “!skip”

Expected result:
1. The bot will notify the user there is no song to skip

Test Case 0016
System: music module Phase: 2
Pause music
Severity: 1
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Play a song with “!play <url>”
4. Enter “!pause”

Expected result:
1. The song will pause

Test Case 0017
System: music module Phase: 2
Pause music without music playing
Severity: 2
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Enter “!pause”

Expected result:
1. The bot will notify the user there is no song to pause

Test Case 0018
System: music module Phase: 2
Resume music
Severity: 1
Instructions:
1. Start the bot, play some music, and pause it (See test case 0021 instructions)

2. Enter “!resume”

Expected result:

1. The song that was paused will resume playing

Test Case 0019
System: music module Phase: 2
Resume music
Severity: 1
Instructions:

1. Start the bot in the console with python3 main.py
2. Enter a voice channel
3. Enter “!resume”

Expected result:

1. The bot will notify the user there is nothing to resume

Test Case 0020
System: Music Module Phase: 2
Remove song from queue
Severity: 1
Instructions:

1. Start the bot, enter a voice channel, and start playing two songs (See test case 009)
2. Enter “removeSong 1”

Expected Result:

1. The bot will notify the user the song has been removed, and the song will be removed from the queue

Test Case 0021
System: Music Module Phase: 2
Remove song from queue bad input
Severity: 1
Instructions:

1. Start the bot, enter a voice channel, and start playing two songs (See test case 009)
2. Enter “removeSong abc”

Expected Result:

1. The bot will notify the user that there was an issue removing the song, that perhaps the index is incorrect or the queue is empty.

Test Case 0022
System: Music Module Phase: 2
Clear queue
Severity: 1
Instructions:

1. Start the bot, enter a voice channel, and start playing two songs (See test case 009)
2. Enter “!clear”

Expected result:

1. The bot will notify the queue has been cleared.
2. The queue of upcoming songs will be cleared minus the currently playing song.

Test Case 0023
System: clockModule Phase: 2
Run Timer Test
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!timer 0.5” to start a timer for 30 seconds
3. Wait for timer to complete

Expected Result:

1. The bot will output a timer that dynamically counts down from 30 seconds.
2. When it completes, it will send a message mentioning the user that created the timer and tell them that their timer is done.

Test Case 0024
System: clockModule Phase: 2
Timer continues even while other commands are being used
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!timer 2” to start a timer for 2 minutes.
3. Perform any other command on the bot such as “!help”

Expected Result:

1. The bot will give the user a timer for 2 minutes.
2. The bot will be able to give the user the help page meanwhile the timer is still counting down showing that the bot is still able to perform other tasks while the timer is still working.

Test Case 0025
System: clockModule Phase: 2
Delete Timer
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!timer 2” to start a timer for 2 minutes.
3. Enter “!timer delete” to delete the timer.

Expected Result:

1. The bot will give the user a timer for 2 minutes.
2. When prompted the bot will delete the timer from the chat.

Test Case 0026
System: clockModule Phase: 2
Pause and Unpause Timer
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!timer 1” to start a timer for 1 minute.
3. Enter “!timer pause” to pause the timer.
4. Enter “!timer unpause” to unpause the timer.

Expected Result:

1. The bot will give the user a timer for 1 minute.
2. When prompted the bot will pause the timer and it will stop changing time.
3. When prompted the bot will unpause the timer and it will finish as expected.

Test Case 0027
System: clockModule Phase: 2
Timer Not Found
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timer delete” to attempt to delete a timer.
3. Enter “!timer unpause” to attempt to unpause the timer.
4. Enter “!timer pause” to attempt to unpause the timer.

Expected Result:

1. The bot will tell the user for all three of these commands that there is no timer found.

Test Case 0028
System: clockModule Phase: 2
Timer can’t unpause before being paused
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timer 1” to start a timer for 1 minute.
3. Enter “!timer unpause” to unpause the timer.

Expected Result:

1. The bot will give the user a timer for 1 minute.
2. When prompted the bot will tell the user that the timer cannot be unpaused since it is already running.

Test Case 0029
System: clockModule Phase: 2
Robustness test
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timer abc” to test robustness.

Expected Result:

1. The bot will tell the user to enter a time in order to use the !timer command.

Test Case 0030
System: clockModule Phase: 2
Run Stopwatch
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch” to start a stopwatch.

Expected Result:

1. The bot will create a stopwatch for the user and it will dynamically display the elapsed time.

Test Case 0031
System: clockModule Phase: 2
Stopwatch runs while other commands are being used
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py
2. Enter “!stopwatch” to start a stopwatch.
3. Perform any other command on the bot such as “!help”

Expected Result:

1. The bot will give the user a stopwatch.
2. The bot will be able to give the user the help page meanwhile the stopwatch is still counting showing that the bot is still able to perform other tasks while the timer is still working.

Test Case 0032
System: clockModule Phase: 2
Delete stopwatch
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch” to start a stopwatch.
3. Enter “!stopwatch delete” to delete the stopwatch.

Expected Result:

1. The bot will create a stopwatch for the user and it will dynamically display the elapsed time.
2. The bot will delete the stopwatch and tell the user it has successfully been deleted.

Test Case 0033
System: clockModule Phase: 2
Stop and Unpause Stopwatch
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch” to start a stopwatch.
3. Enter “!stopwatch stop” to stop the stopwatch.
4. Enter “!stopwatch unpause” to unpause the stopwatch.

Expected Result:

1. The bot will create a stopwatch for the user and it will dynamically display the elapsed time.
2. The bot will stop the stopwatch when told to do so and it will display the time elapsed.
3. When prompted the bot will unpause the stopwatch and it will continue from where it left off.

Test Case 0034
System: clockModule Phase: 2
Reset Stopwatch
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch” to start a stopwatch.
3. Enter “!stopwatch reset” to reset the stopwatch

Expected Result:

1. The bot will create a stopwatch for the user and it will dynamically display the elapsed time.
2. The bot will reset the stopwatch when prompted and the stopwatch will start again from zero.

Test Case 0035
System: clockModule Phase: 2
Stopwatch Not Found
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch delete” to attempt to start a stopwatch.
3. Enter “!stopwatch stop” to attempt to stop a stopwatch.
4. Enter “!stopwatch unpause” to attempt to unpause a stopwatch.
5. Enter “!stopwatch reset” to attempt to reset a stopwatch.

Expected Result:

1. The bot will tell the user for all fourof these commands that there is no stopwatch is found.

Test Case 0036
System: clockModule Phase: 2
Cannot Unpause a running Stopwatch
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch” to start a stopwatch.
3. Enter “!stopwatch unpause” to unpause the stopwatch.

Expected Result:

1. The bot will give the user a stopwatch.
2. When prompted the bot will tell the user that the stopwatch cannot be unpaused since it is already running.

Test Case 0037
System: clockModule Phase: 2
Robustness Test
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!stopwatch abc” to test robustness.

Expected Result:

1. The bot will tell the user to enter a valid argument when using the !stopwatch command.

Test Case 0038
System: clockModule Phase: 2
Timezone Test
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timezone Indianapolis” to see the local time of Indianapolis.

Expected Result:

1. The bot will tell the user the local time of Indianapolis as well as the time difference between where the bot is being run from and Indianapolis.

Test Case 0039
System: clockModule Phase: 2
Timezone Test
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timezone Salt Lake City” to see the local time of Salt Lake City.

Expected Result:

1. The bot will tell the user the local time of Salt Lake City as well as the time difference between where the bot is being run from and Salt Lake City.

Test Case 0040
System: clockModule Phase: 2
Timezone Test
Severity: 1
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!timezone” to see the local time of the bot.

Expected Result:
1. The bot will tell the user the local time of the timezone that the bot is being run from.

Test Case 0041
System: Music Module Phase: 2
Severity: 2
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter a voice channel
3. Enter “!play <songname>” where song name is any audio you want to listen to.

Expected result:
1. The bot will notify the user the music is being prepared.
2. After a short delay, the desired music will start playing.

Test Case 0042
System: decisionMaker Phase: 2
Severity: 2
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter “!decide” without specifying arguments.

Expected result:
1. The bot should notify the user that the format for calling the command is incorrect.
2. The bot should tell the user where to find the proper format.

Test Case 0043
System: decisionMaker Phase: 2
Severity: 1
Instructions:
1. To start the bot, at the console, enter: python3 main.py
2. Enter “!decide number” to pick an integer between 0 and 9.

Expected result:
1. The bot should pick an integer between 0 and 9 and display it to the user.

Test Case 0044

System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide number <bound>” where bound is any integer value.

Expected result:

1. The bot should pick an integer between 0 and <bound>, even if <bound> is negative.

Test Case 0045
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide number <bound1> <bound2>” where bound1 and bound2 are both integers.

Expected result:

1. The bot should pick an integer between <bound1> and <bound2> regardless of which one is larger.

Test Case 0046
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide number <bound>” where bound is any float value.

Expected result:

1. The bot should pick a float between 0 and <bound>, even if <bound> is negative.

Test Case 0047
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide number <bound1> <bound2>” where bound1 is a float and bound2 is an integer or bound1 is a integer and bound2 is a float.

Expected result:

1. The bot should pick a float between <bound1> and <bound2> regardless of which one is larger.

Test Case 0048
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.

2. Enter “!decide number <bound1> <bound2>” where bound1 and bound2 are both floats.

Expected result:

1. The bot should pick a float between <bound1> and <bound2> regardless of which one is larger.

Test Case 0049
System: decisionMaker Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide <bound>” where bound is any float or integer.

Expected result:

1. The bot should notify the user that the format for calling the command is incorrect.
2. The bot should tell the user where to find the proper format.

Test Case 0050
System: decisionMaker Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide number <text>” where text is any string that contains alphabetic letters.

Expected result:

1. The bot should notify the user that there was an error with the arguments given to the command.

Test Case 0051
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide custom <option1> <option2> <option3>” where the options can be any alphanumeric word with no spaces.

Expected result:

1. The bot should parse all the options specified.
2. The bot should pick one option from the three specified options.
3. The bot should display to the user the picked option and all possible options.

Test Case 0052
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide custom <option1> <option2> <option3>” where the options are an alphanumeric phrase with spaces surrounded by quotes (such as “option one”).

Expected result:

1. The bot should parse all the options specified.
2. The bot should pick that one option from the three specified options.
3. The bot should display to the user the picked option and all possible options.

Test Case 0053
System: decisionMaker Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide <option1> <option2> <option3>” where the options can be any alphanumeric word with no spaces.

Expected result:

1. The bot should notify the user that the format for calling the command is incorrect.
2. The bot should tell the user where to find the proper format.

Test Case 0054
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide 2 custom <option1> <option2> <option3>” where the options can be any alphanumeric word with no spaces.

Expected result:

1. The bot should parse all the options specified.
2. The bot should pick two options from the three specified options. The options picked cannot be the same option.
3. The bot should display to the user the picked options and all possible options.

Test Case 0055
System: decisionMaker Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!decide 2 number” to pick two unique integers between 0 and 9.

Expected result:

1. The bot should pick two integers between 0 and 9 and display it to the user. These two integers cannot be the same.

Test Case 0056
System: decisionMaker Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.

2. Enter “!decide 11 number” to pick eleven unique integers between 0 and 9, which is not possible.

Expected result:

1. The bot should notify the user that there was an error with the arguments given to the command.

Test Case 0057
System: pollModule Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot the name of a text channel in the server.

Expected result:

1. The bot should send the poll as a formatted message to the text channel specified by text channel name. It should have the correct title, question, options, and reactions. The time remaining should also be indicated.
2. After the time limit, the bot should remove the poll and send the poll results. It should have the total votes, who voted for what, and total votes for each option. The bot should also direct message the user notifying them that the poll has ended.

Test Case 0058
System: pollModule Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot “none”. This is the poll time limit.
8. Direct message the bot the name of a text channel in the server.
9. Direct message the bot “done 0”. This concludes the poll.

Expected result:

1. The bot should send the poll as a formatted message to the text channel specified. It should have the correct title, question, options, and reactions.
2. Upon messaging “done 0”, the bot should remove the poll and send the poll results. It should have the total votes, who voted for what, and total votes for each option. The bot should also direct message the user notifying them that the poll has ended.

Test Case 0059
System: pollModule Phase: 2
Severity: 1
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. Direct message the bot “auto”.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot the name of a text channel in the server.

Expected result:
1. Upon messaging “auto”, the bot should react to the message that obtains the user’s reactions with the specified number of options.
2. The bot should send the poll as a formatted message to the text channel specified. It should have the correct title, question, options, and reactions. The time remaining should also be indicated.
3. After the time limit, the bot should remove the poll and send the poll results. It should have the total votes, who voted for what, and total votes for each option. The bot should also direct message the user notifying them that the poll has ended.

Test Case 0060
System: pollModule Phase: 2
Severity: 2
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot any word with alphabetic characters excluding “none”. This should be the poll time limit.

Expected result:

1. The bot should direct message the user back indicating that the time limit specified is not valid and cannot be parsed.
2. The bot should indicate that the time limit must be an integer.

Test Case 0061
System: pollModule Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot the name of a text channel that does not exist on the server.

Expected result:

1. The bot should direct message the user back indicating that the text channel does not exist on the server.

Test Case 0062
System: pollModule Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot the name of a text channel that corresponds to multiple text channels on the server.

Expected result:

1. The bot should direct message the user back indicating that multiple text channels were found with the provided name.
2. The bot should then display the channel IDs of all such text channels and instruct the user to use the channel ID to specify a text channel.

Test Case 0063
System: pollModule Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot “cancel”.
5. Direct message the bot any phrase.

Expected result:

1. Upon messaging “cancel”, the bot should indicate that the poll creation process has been cancelled.
2. The bot should not respond with prompts used for poll creation for any direct message after the “cancel” direct message.

Test Case 0064
System: pollModule Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Enter “!createpoll” again.

Expected result:

1. Upon messaging “!createpoll” while the poll creation process is already taking place, the bot should notify the user that poll creation is already in progress and you cannot create two polls at once.

Test Case 0065
System: pollModule Phase: 2
Severity: 2
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot “cancel”.
4. Enter “!createpoll” again.

Expected result:

1. Upon messaging “!createpoll” after poll creation has ended, the bot should direct message the user for instructions on poll creation and allow the user to create a poll.

Test Case 0066
System: pollModule Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric word without phrases. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot “<text-channel-id> id” where <text-channel-id> is the channel ID of a text channel in the server.

Expected result:

1. The bot should send the poll as a formatted message to the text channel specified by the channel ID. It should have the correct title, question, options, and reactions. The time remaining should also be indicated.
2. After the time limit, the bot should remove the poll and send the poll results. It should have the total votes, who voted for what, and total votes for each option. The bot should also direct message the user notifying them that the poll has ended.

Test Case 0067
System: pollModule Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Direct message the bot an alphanumeric phrase. This is the poll title.
4. Direct message the bot an alphanumeric phrase. This is the poll question.
5. Direct message the bot “<option 1> <option 2>” where each option is an alphanumeric phrase with spaces surrounded by quotes. These are the poll options.
6. The bot should indicate a message to react to. React with two different reactions.
7. Direct message the bot an integer value. This is the poll time limit.
8. Direct message the bot the name of a text channel in the server.

Expected result:

1. The bot should send the poll as a formatted message to the text channel specified by text channel name. It should have the correct title, question, options, and reactions. The time remaining should also be indicated.

2. After the time limit, the bot should remove the poll and send the poll results. It should have the total votes, who voted for what, and total votes for each option. The bot should also direct message the user notifying them that the poll has ended.

Test Case 0068
System: pollModule Phase: 2
Severity: 1
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!createpoll” to initiate poll creation process. You should receive a direct message from the bot.
3. Enter an alphanumeric phrase to the same text channel you sent “!createpoll”. This is the poll title but specified in the wrong location.

Expected result:

1. The bot should not parse the poll title sent to the text channel. During poll creation, it should only parse and respond to direct messages sent to the bot by the user.

Test Case 0069
System: riot Phase: 2
Severity: 1
Display League profile
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!league <region> <in-game-name>” in discord.

Expected Result

1. The bot should respond with the League profile of the user in a region. This includes their profile icon, ranked stats, and level.

Test Case 0070
System: riot Phase: 2
Severity: 1
Display TFT profile
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!tft <region> <in-game-name>” in discord.

Expected Result

1. The bot should respond with the TFT profile of the user in a region. This includes their profile icon, ranked stats, and level.

Test Case 0071
System: riot Phase: 2
Severity: 1
Display Riot regions
Instructions:

1. To start the bot, at the console, enter: python3 main.py.

2. Enter “!regions” in discord.

Expected Result

1. The bot should respond with the tags of all valid regions.

Test Case 0072
System: riot Phase: 2
Severity: 1
Displays Summoner Match History
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!matchhistory <region> <in-game-name>”

Expected Result

1. The bot should respond to the user with a list of the specified in-game-name’s recent match history including the last 5 games they played of League of Legends.

Test Case 0073
System: riot Phase: 2
Severity: 1
Displays Champion Mastery
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!mastery <region> <in-game-name>”

Expected Result

1. Bot responds with a list of the top 3 highest mastery champions that the user has and their respective mastery points and level from highest to lowest mastery top to bottom.

Test Case 0074
System: riot Phase: 2
Severity: 1
Live Match info
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!live <region> <in-game-name>”

Expected Result

1. Bot responds with information on a live game of the inputted name. This includes queue type, information on all players in the game, and runes on champions they are playing.

Test Case 0075
System: riot Phase: 2
Severity: 1
Live Match info
Instructions:

3. To start the bot, at the console, enter: python3 main.py.
4. Enter “!live <region> <in-game-name>”

Expected Result

2. Bot responds with information on a live game of the inputted name. This includes queue type, information on all players in the game, and runes on champions they are playing.

Test Case 0076
System: riot Phase: 2
Severity: 1
Recommended champion skill order
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!skillorder <champion-name>” in discord.

Expected Result

1. Bot responds with the recommended skill order of the inputted champion.

Test Case 0077
System: riot Phase: 2
Severity: 1
Recommended champion item builds
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!items <champion-name>” in discord.

Expected Result

1. Bot responds with the recommended item builds of the inputted champion.

Test Case 0078
System: riot Phase: 2
Severity: 1
Champion data
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!champ <champion-name>” in discord.

Expected Result

1. Bot responds with the profile of inputted champion. This includes a champion description, playstyle, resource type, and descriptions on each of their abilities.

Test Case 0079
System: riot Phase: 2
Severity: 1
Check Jungle Database
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!database” in discord.

Expected Result

1. The bot responds with a link to a Google doc containing a list of jungle champion names and their clear paths

Test Case 0080
System: riot Phase: 2
Severity: 1
Jungle Path List Recommendation
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!firstclear <jungle_champion_name>” in discord.

Expected:
1. The bot will respond with a jungle path in the form of a list recommended for that specific jungler.

Test Case 0081
System: riot Phase: 2
Severity: 1
Best Junglers on that Champion
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!topjg <jungle_champion_name>” in discord.

Expected:
1. The bot will respond with a list of the top 3 highest rated junglers on the North American Server on that specific champion with a link to their detailed in-game profiles.

Test Case 0082
System: riot Phase: 2
Severity: 1

Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Enter “!pmpath <champion_name>”

Expected:
1. Bot will private message the user with jungle path of the inputted champion they are playing.

Test Case 0083
System: riot Phase: 2
Severity: 1
Private Message Updates with Overlay
Instructions:
1. To start the bot, at the console, enter: python3 main.py.
2. Check to see if the setting for Discord Overlay is turned on.
3. Enter a League of Legends game.

4. Enter “!overlay <in-game-name>” preferably in the loading screen.

Expected:

1. The bot will private message the user at certain timers in the game messages about the camps they should be doing given ideal circumstances on the character the user is playing.

Test Case 0084
System: riot Phase: 2
Severity: 1
League overlay updates exit
Instructions:

1. To start the bot, at the console, enter: python3 main.py.
2. Assume the overlay function was already called.

Expected:

1. After about 3 minutes, the bot will private message the user with an exit message.