

Pokémon League Manager Bot Test Procedures

By: Andrew Fleming

Overview:

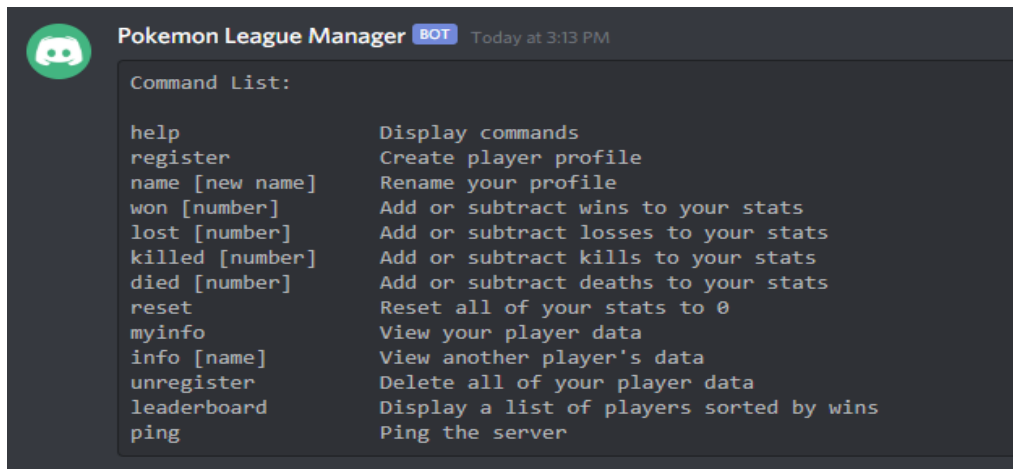
The main focus for the system test procedure is to make sure all of the commands work properly when the user is registered, and throw errors when the user attempts to enter them without registering, first. The commands are meant to be easily understandable and simple to use, so there is a help command that lists every command. Player data and the command list are pulled from a SQL database, so it's important to ensure that the commands properly write to the database.

Hosting:

To begin the testing, we must make sure the bot is being hosted properly and free of errors. To do so, we must navigate to the hosting service's website and use their command line console. Then we can attempt to run the project by typing "node index.js". If the bot starts up successfully and shows that it is online on Discord, that is a good sign it is working correctly. To ensure that the bot can be reached by users, enter the "ping" command to show that the server is sending a response.

Help (Non-Functional Requirement #1):

The "=help" command is meant to allow users to have an easily accessible list of commands. The help command is most often a universal command for Discord bots, so it should be easy for the user to figure it out. If successful, the bot should return a list of the commands along with descriptions of what they do. Below is the expected output:



The screenshot shows a Discord chat window with the header "Pokemon League Manager BOT" and a timestamp "Today at 3:13 PM". The bot's avatar is a green circle with a white robot face. The message content is "Command List:" followed by a list of commands and their descriptions in a two-column format.

Command	Description
help	Display commands
register	Create player profile
name [new name]	Rename your profile
won [number]	Add or subtract wins to your stats
lost [number]	Add or subtract losses to your stats
killed [number]	Add or subtract kills to your stats
died [number]	Add or subtract deaths to your stats
reset	Reset all of your stats to 0
myinfo	View your player data
info [name]	View another player's data
unregister	Delete all of your player data
leaderboard	Display a list of players sorted by wins
ping	Ping the server

Registration (Functional Requirement #1):

To register, users must simply type “=register”. This command records their Discord ID, as well as adds default values to each field. The name defaults to their Discord username and all numerical fields default to 0. If the register operation is successful, the user should be able to type “=myinfo” and view their player profile. If the user has already registered, the bot should return a message informing them that they cannot register again.

Next, we have to test the “=unregister” command. This will clear the user’s information from the database, and make it so they are unable to access any command other than “=help” and “=register”. To test this command, we only have to run it and then check that the user is revoked access to the user-specific commands. Just like the “=register” command, if the player isn’t registered, it should return with an error message informing the user that they cannot run the command.

Further testing regarding registration is testing every command that isn’t “=help” or “=register” returns an error message if the user isn’t registered yet. The user shouldn’t be able to make changes to a profile that doesn’t yet exist.

Info (Functional Requirement #2):

The “=info” and “=myinfo” commands are essential to this bot, because the bot’s main purpose is to allow users to track their statistics, as well as others. These commands are meant to display different users’ player profiles. The “=myinfo” command allows players to view their own profiles, listing their name, wins, losses, win/loss average, kills, deaths, and kill/death average. By entering “info” followed by a valid username, the user can view other player’s statistics. This command is one of the main tools used in testing, as it allows us see the values that we are attempting to update.

Name (Functional Requirement #2):

The “=name” command allows the user to further change the name associated with their player profile from the default value. The command requires that the user type the command, followed by their new name on the same line, such as “=name Kyle Johnson”. The name command is meant to help the user easily customize and personalize their profile if a different name is desired. To check that the command was successful, you can enter “=myinfo” to see that the name has changed correctly to the new one.

Player Statistics (Functional Requirement #2):

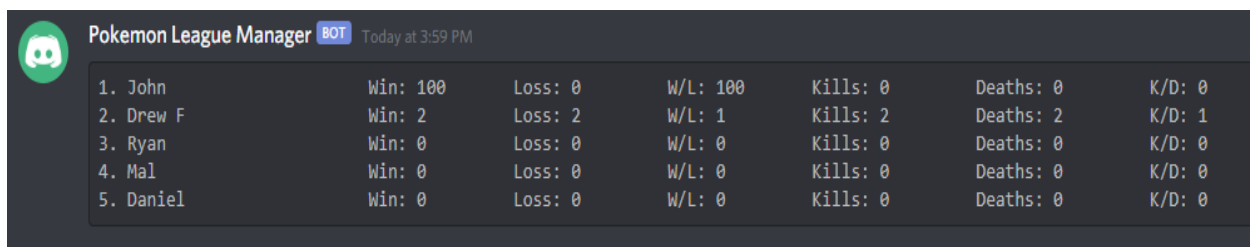
There are several commands to be tested that are meant to manipulate player statistics. If the operations are successful, they should be reflected in the statistics shown by the “=myinfo” command. The player statistic commands are: =won, =lost, =killed, and =died. By putting a positive number next to the commands, you can add that many to the command’s respective player stat. On the other hand, putting a negative number will subtract from the designated player stat. If the user enters a value that isn’t an integer, the bot will return an error message.

Behind the scenes, every time a value has changed, either K/D or W/L should update based on the changed value. If kills or deaths are changed, K/D should automatically be averaged and updated. This is the same with wins and losses triggering W/L to be updated. If either losses or deaths’ values are 0, the averaging operation should just return the value of the opposite statistic to prevent division by zero. It is crucial to prevent division by zero errors, as they will crash the entire bot.

Another command that manipulates the player statistics is “reset”. As you can probably infer, this allows users to revert all of the statistics back to 0, to start anew. This command is only meant to change wins, losses, W/L, kills, deaths, and K/D back to their default values. The name should remain the same. This change should be reflected in the player profile displayed by “myinfo”.

Leaderboard (Functional Requirement #4):

The “=leaderboard” function is meant to allow users to see a complete list of their fellow competitors sorted by their wins, from greatest to least. In addition, the sorting function displays the number ranking of the user based on their place in the list, as well as each user’s statistics. The leaderboard isn’t stored in a database and is only created and sorted when the “=leaderboard” command is run. This command is also meant to be exclusive to registered users. The following is the expected output:



1. John	Win: 100	Loss: 0	W/L: 100	Kills: 0	Deaths: 0	K/D: 0
2. Drew F	Win: 2	Loss: 2	W/L: 1	Kills: 2	Deaths: 2	K/D: 1
3. Ryan	Win: 0	Loss: 0	W/L: 0	Kills: 0	Deaths: 0	K/D: 0
4. Mal	Win: 0	Loss: 0	W/L: 0	Kills: 0	Deaths: 0	K/D: 0
5. Daniel	Win: 0	Loss: 0	W/L: 0	Kills: 0	Deaths: 0	K/D: 0

That concludes testing.

