Project Statement

My project idea is to make a Discord bot to help breathe new life into dying discord servers in a silly way. Effectively, what I’ve been thinking about is creating a “Drama” bot, or a text channel bot that will claim that they witnessed something embarrassing or childish that was done or happened to a user in the server. So, for example if Kevin is an unsuspecting user, I could prompt the Drama Bot with a command along the lines of **#drama @Kevin**, and a Mad Libs style message should be generated claiming Kevin did something silly whether it’s true or not. The #drama command should also be prepared for random input of a user as well (utilizing different constructors) to randomly choose a victim for the Mad Libs message. So, the #drama command alone would randomly select a user in the server to be subject to silly fake drama, and specifying a single user would guarantee that person as the victim in the resulting message.

The message would be a random template that I will create from a first person perspective of the Drama Bot so that they can appear to be a person at first glance. Then, in the template there will be random subject matter variables that will be populated with random nouns from an established list. Finally, there should also be a random action/verb from another list to maximize the silliness from the result. I plan on having at least 5 templates for the overall message structure, and at least 10 nouns and 10 verbs to be picked at random for filling out the template.

At the very least, I plan to use this occasionally to get my friends to peek at the server again.

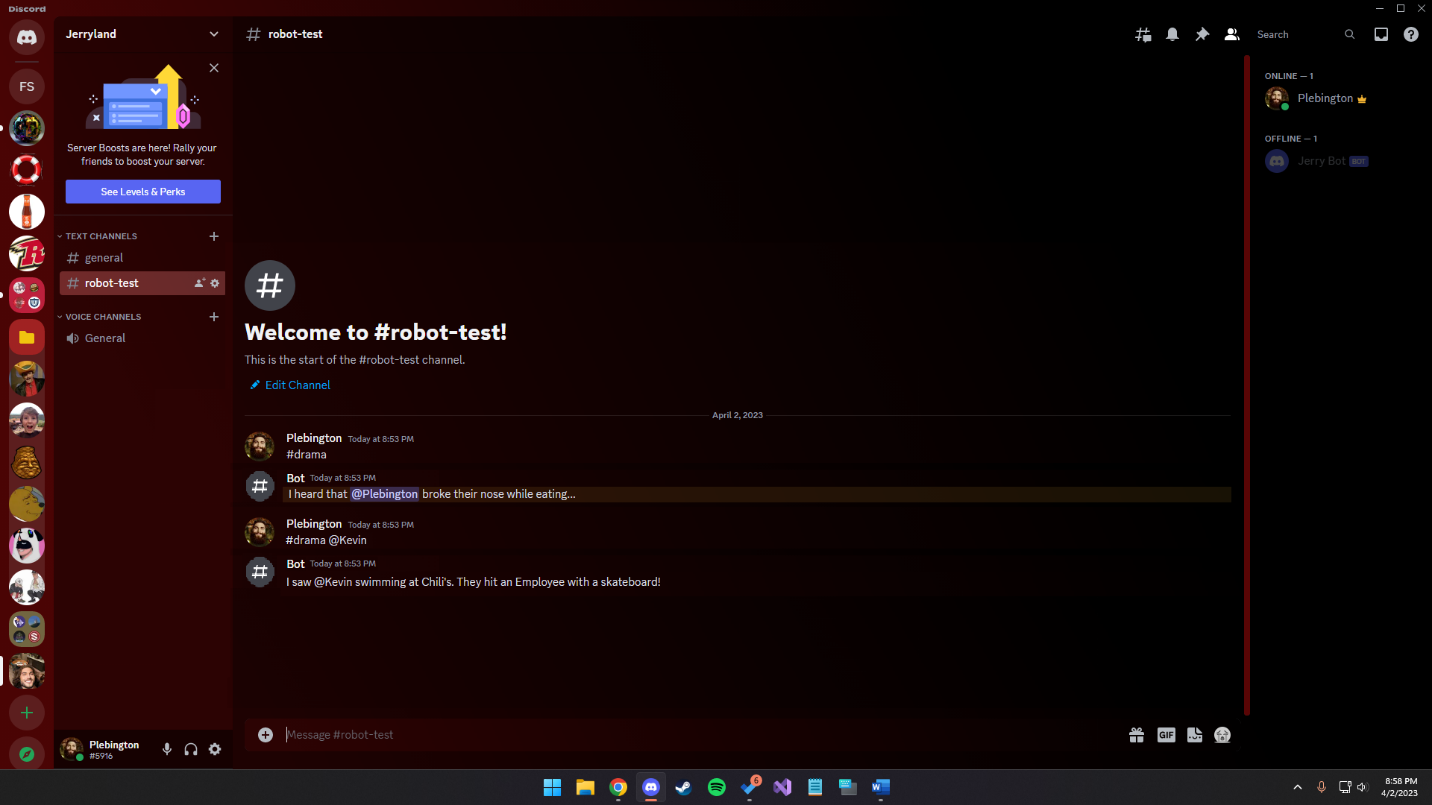
Project Plan

1. Flesh out the requirement list. (30 minutes dedicated, and more as they appear.)
2. Set up the development environment. (30 minutes) (Visual Studio, include necessary libraries and establish planned classes.)
3. Set up GitHub for the Project (30 minutes)
4. Invite Professor Jenkins to the GitHub project (5 minutes)
5. Writing Code Week 1 (3 hours)
6. Test the bot Week 1 (2 hours)
7. Maintenance Combing Week 1 (2 hours)
8. Writing Code Week 2 (3 hours)
9. Test the bot Week 2 (2 hours)
10. Maintenance Combing Week 2 (2 hours)
11. Writing Code Week 3 (3 hours)
12. Test the bot Week 3 (2 hours)
13. Maintenance Combing Week 3 (2 hours)
14. Writing Code Week 4 (5 hours)
15. Test the bot Week 4 (3 hours)
16. Maintenance Combing Week 4 (3 hours)
17. Make class diagrams. (3 hours) (UML models.)
18. Write a document regarding how requirements are met (3 hours)
19. Finalize testing proof document (1 hour)
20. Submit the project (1 hour)

Project Requirements

* Functional Requirements
  + A user should be able to call the #drama command via text with nothing else provided.
  + A user should be able to call the #drama command via text followed by @(user) to specify someone involved.
  + The bot will respond to the #drama command with a randomly selected template.
  + Each bot response will replace at least one noun variable with a randomly selected noun.
  + Each bot response will replace at least one verb variable with a randomly selected verb.
  + Each bot response will @ the user involved (to bring attention to their post).
  + The bot should be functional no matter what kind of device is being used (i.e. mobile, desktop, browser, etc.)
* Non-Functional Requirements
  + The bot will speak from the first person as if they witnessed something.
  + A bot response should be posted within 3 seconds of a user using the #drama command.
  + Every valid instance of the #drama command should be addressed by the bot.
  + The #drama command should not execute unless it is standalone or followed only by a @(user).
  + The bot’s code should have an output blacklist (users who will be drama-free).
  + If the #drama command has a blacklisted victim @ed, it should then treat the #drama command as if there was no @(user) at all and pick someone at random.
  + The bot’s code should have an input blacklist (users who are not allowed create drama).
  + If the #drama command is used by a user on the input blacklist, this should be the only time the bot does not respond with the Mad Libs style message.

User Interface Mockup



Class Design

1. bot
   1. The bot class will be a class mostly dedicated to helping the bot actually function and handle included packages properly.
2. message
   1. The message class will help assemble the message for output, which entails randomly selecting a template, including a user (random or specified), a randomly selected noun, and a randomly selected verb.
3. blacklists
   1. The blacklists class will have a hard-coded list of usernames to ban from using the command, as well as a different list of users the bot can under no circumstances use in a message (instead will use a random user.)
4. wordBank
   1. The wordBank class will be used by the message class to return the noun and verb when methods are used. This will not only be nice to handle anything related to the random selection of nouns and verbs, but also provide the option of adding to the word bank by hardcoding more options.