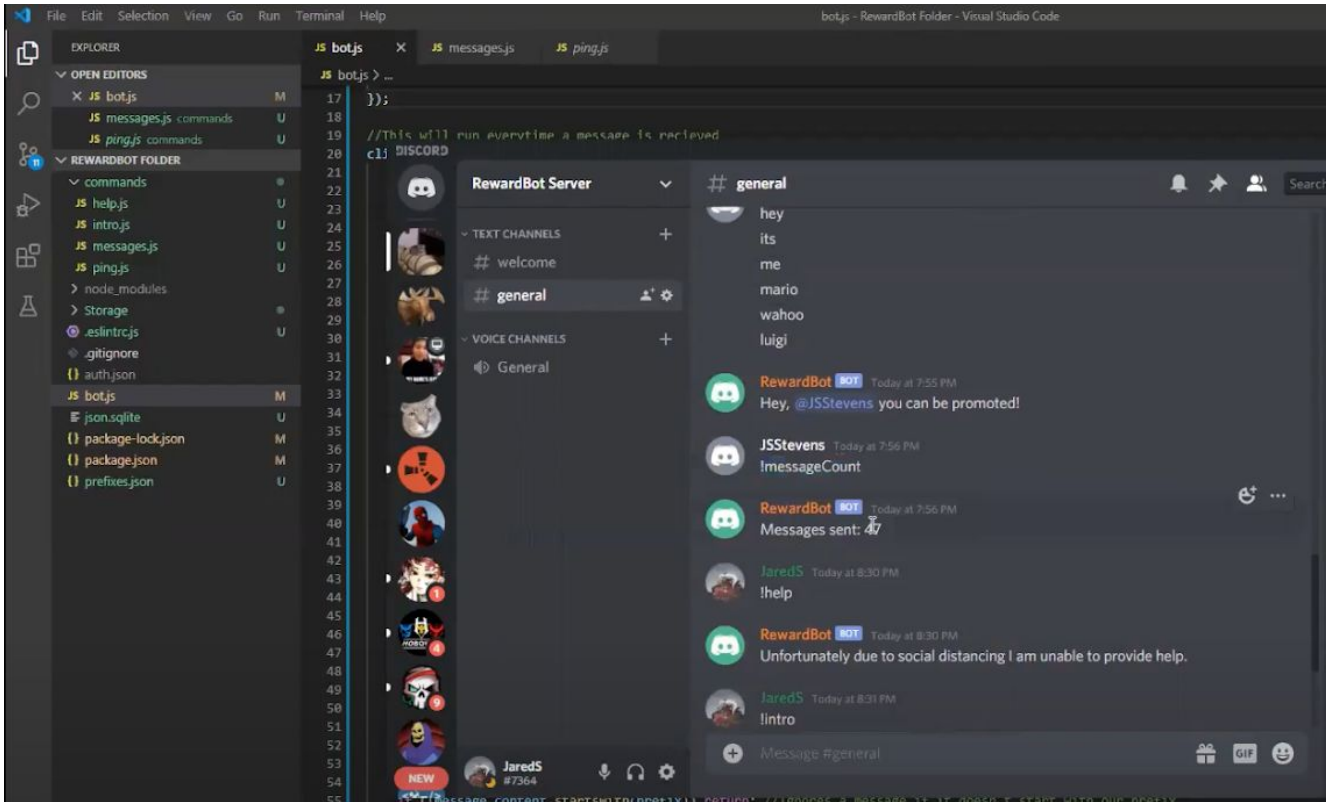
RewardBot

→ *Milestone: Report* ←



```
});

//This will run everytime a message is recieved
```

general

hey
its
me
mario
wahoo
luigi

RewardBot BOT Today at 7:55 PM
Hey, @JSStevens you can be promoted!

JSStevens Today at 7:56 PM
!messageCount

RewardBot BOT Today at 7:56 PM
Messages sent: 47

JaredS Today at 8:30 PM
!help

RewardBot BOT Today at 8:30 PM
Unfortunately due to social distancing I am unable to provide help.

JaredS Today at 8:31 PM
!intro

Team Members: Ruthy Levi, Madeleine Daniell, Jared Sirios

Date: *May 15th, 2020*

Problem:

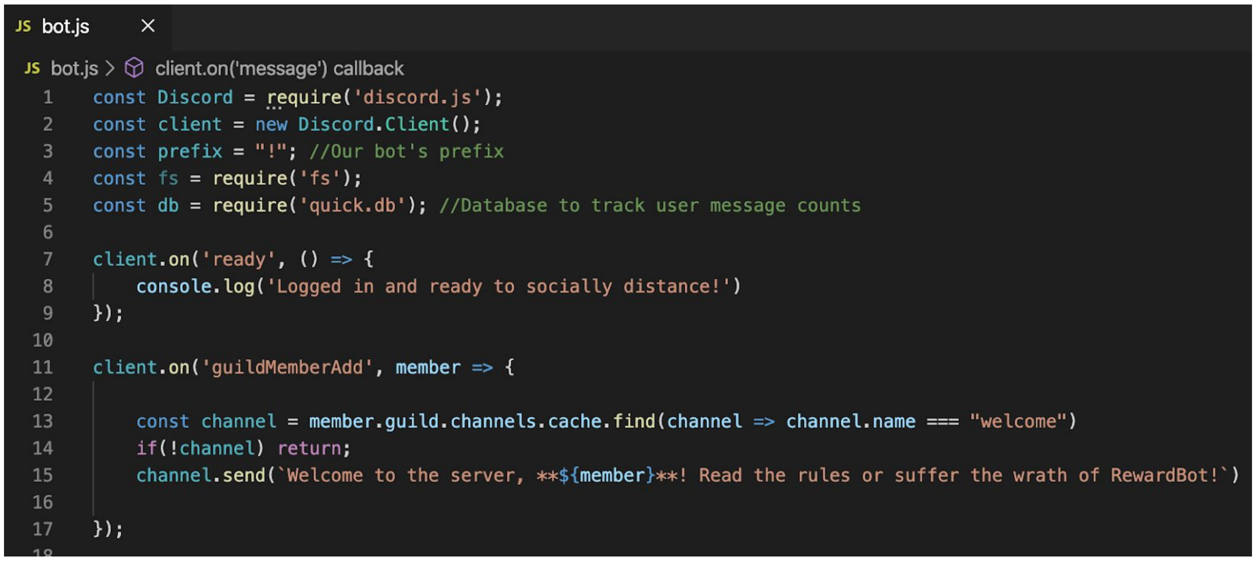
For our final project, the group has decided to create a “RewardBot” in a Discord Server. We utilized Node.js, a Javascript runtime, with discord.io, a Discord library. The problem this bot addresses is collaboration -- in particular, promoting contribution. Oftentimes in a crowdsource environment, engagement between users can be difficult. There is a certain anonymity in a crowdsource environment -- (it’s in the title, ‘crowd’) -- and many users do not engage with others. The majority of users simply gather whatever information they need and typically don’t contribute. This type of environment without contributing users will not allow the domain to grow and reach new heights. Anybody that looks into the domain will see only a small group of users contributing while all other users simply “mooch” off of the others. Once this becomes a standard, this domain and environment will die in no time.

Primary features and screenshots:

The proposed bot to act as a solution with this problem is a RewardBot. The RewardBot is the catalyst that will encourage the users to become more active. RewardBot is simple in design, and thus relies heavily on interactions with users. This bot’s main purpose is to be the user’s tracker for all of their stats. It will track everything from contributions, account status, user-to-user interactions and multiple other status. Once certain thresholds are met, for example 25 contributions to a project, the bot will “reward” the user with some set amount of fake points and a badge. Even though the points and the badge are fake, the idea of being rewarded will give the user the desire to be rewarded again, because everyone loves being rewarded.

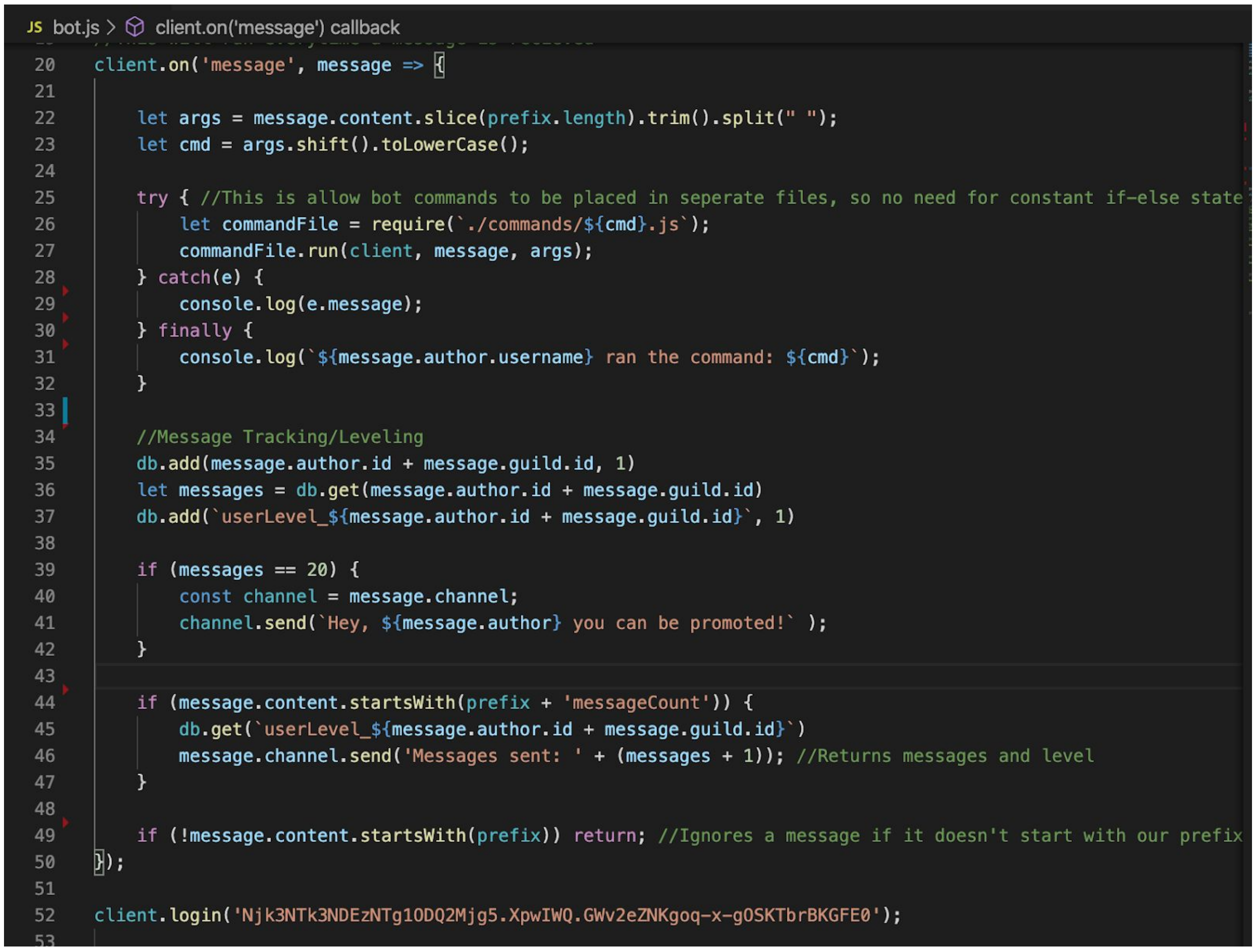
Once the user has that desire, it becomes easy to implement more and more rewards to get users even more involved. These badges and points could also be publicly displayed on the user's profile. You could specifically make rewards in areas that you want users to interact in more. And, obviously, actual rewards can be given to users after they gather a certain amount of badges and points. For example, after a user has gathered 100 badges they could be granted certain privileges other users wouldn't have; these users would be Trusted Users and could be granted certain permissions.

Code wise, the bot runs and is controlled in "bot.js", and the commands run in the command folder. Below are functions that run when the bot is on. *'Ready'* logs into the console once the bot is launched, and "*guildMemberAdd'* runs when a member is added to the server. It goes to the welcome channel, and welcomes the user.



```
const Discord = require('discord.js');
const client = new Discord.Client();
const prefix = "!"; //Our bot's prefix
const fs = require('fs');
const db = require('quick.db'); //Database to track user message counts

client.on('ready', () => {
    console.log('Logged in and ready to socially distance!')
});

client.on('guildMemberAdd', member => {

    const channel = member.guild.channels.cache.find(channel => channel.name === "welcome")
    if(!channel) return;
    channel.send(`Welcome to the server, **${member}**! Read the rules or suffer the wrath of RewardBot!`)

});
```

“Message” runs whenever a message is sent in the channel that the bot has access to. It gets rid of the prefix, and turns everything into lowercase to rejadust the words.



```
20  client.on('message', message => {
21  
22      let args = message.content.slice(prefix.length).trim().split(" ");
23      let cmd = args.shift().toLowerCase();
24  
25      try { //This is allow bot commands to be placed in seperate files, so no need for constant if-else state
26          let commandFile = require(`./commands/${cmd}.js`);
27          commandFile.run(client, message, args);
28      } catch(e) {
29          console.log(e.message);
30      } finally {
31          console.log(`${message.author.username} ran the command: ${cmd}`);
32      }
33  
34      //Message Tracking/Leveling
35      db.add(message.author.id + message.guild.id, 1)
36      let messages = db.get(message.author.id + message.guild.id)
37      db.add(`userLevel_${message.author.id + message.guild.id}`, 1)
38  
39      if (messages == 20) {
40          const channel = message.channel;
41          channel.send(`Hey, ${message.author} you can be promoted!` );
42      }
43  
44      if (message.content.startsWith(prefix + 'messageCount')) {
45          db.get(`userLevel_${message.author.id + message.guild.id}`)
46          message.channel.send('Messages sent: ' + (messages + 1)); //Returns messages and level
47      }
48  
49      if (!message.content.startsWith(prefix)) return; //Ignores a message if it doesn't start with our prefix
50  });
51  
52  client.login('Njk3NTk3NDEzNTg1ODQ2Mjg5.XpwIWQ.GWv2eZNKgoq-x-gOSKTbrBKGFE0');
53  
```

It also concerns tracking and leveling, which can be seen starting on line 39. The messages are tracked, and then when a user sends 45 messages, it sends a message to the server saying they are eligible for a promotion. Then , there is a message count which responds to the channel and spits out how many messages that user has sent. After that, it ignores anything that doesn’t start with a prefix. The commands in the command folder can be called upon in Discord. For instance, !MessageCount displays the amount of messages that user has sent.

Reflection:

Throughout the development process of the bot certain problems did occur and almost halted progress. There was a problem with one of the libraries used that had to be changed, as well as certain problems with getting commands to work when necessary. For example, when a user reaches a certain amount of messages, the bot is supposed to immediately send a message letting the person know they reached the next level. However, it often takes the bot some time to send the messages, not usually until 1 or 2 messages later. Although the exact problem behind this isn't fully known it can be improved upon in future work. To touch on a more positive reflection, making the functions of the bot was surprisingly easy; each function of the bot started with client.on().

Limitations and future work:

The primary limitation that was found when completing the bot was that since the bot's authentication is tied to one Discord account and one machine, only that one person can run the bot and make changes to the code the bot uses. Through the use of GitHub collaboration is possible, but the only way to make sure the code actually works is for the one person to update and start the bot. This didn't completely prevent collaboration, though.

As for future work, there are several improvements that can be made. Currently the bot only works in one channel, but since the bot is ultimately designed for a large community, it can be programmed to send it's responses and messages into different channels to prevent completely clogging one channel. There are also multiple updates that can be made to improve the bot’s functionality. The functions the bot already has can be fine tuned more, and other functions can be added to promote user interaction.