Algorithm & Programming
Final Project

Discord BOT

Chellshe Love Simrochelle
Student ID: 2502043040
COMP6047001
17th January 2022
Presented to: Mr Jude Joseph Lamug Martinez

Project Specification

Discord is an app used for real-time text chat, Discord servers are set up by people and generally are focused on a certain community, many groups and organizations have Discord servers for people to chat with each other. A Discord server often has many channels which are different rooms that you can chat in. For this final project, I developed a friendly Discord bot that makes people feel welcomed and encouraged. Using Python, the bot runs completely in the cloud. This means that I do not need to install anything on my computer and I do not need to pay anything to host my bot. I used a number of tools to make this project come to life, including the Discord API, some Python libraries and a cloud computing platform called Replit to launch my Discord Bot. A Discord Bot is treated as a user in servers on the platform that is controlled by a program instead of a person. However, you can still code a Discord Bot to make it do whatever you please. There are many things Discord Bots are often used for such as chatting, responding to messages, managing servers, playing music etc. I also used webhooks with my Discord Bot to give real-time updates on a specific GitHub repository.

Input

- User types in a specific command provided in a list by the Discord Bot.

Output

- The Discord Bot will respond to the commands imputed by the user with the programmed words.

Solution Design

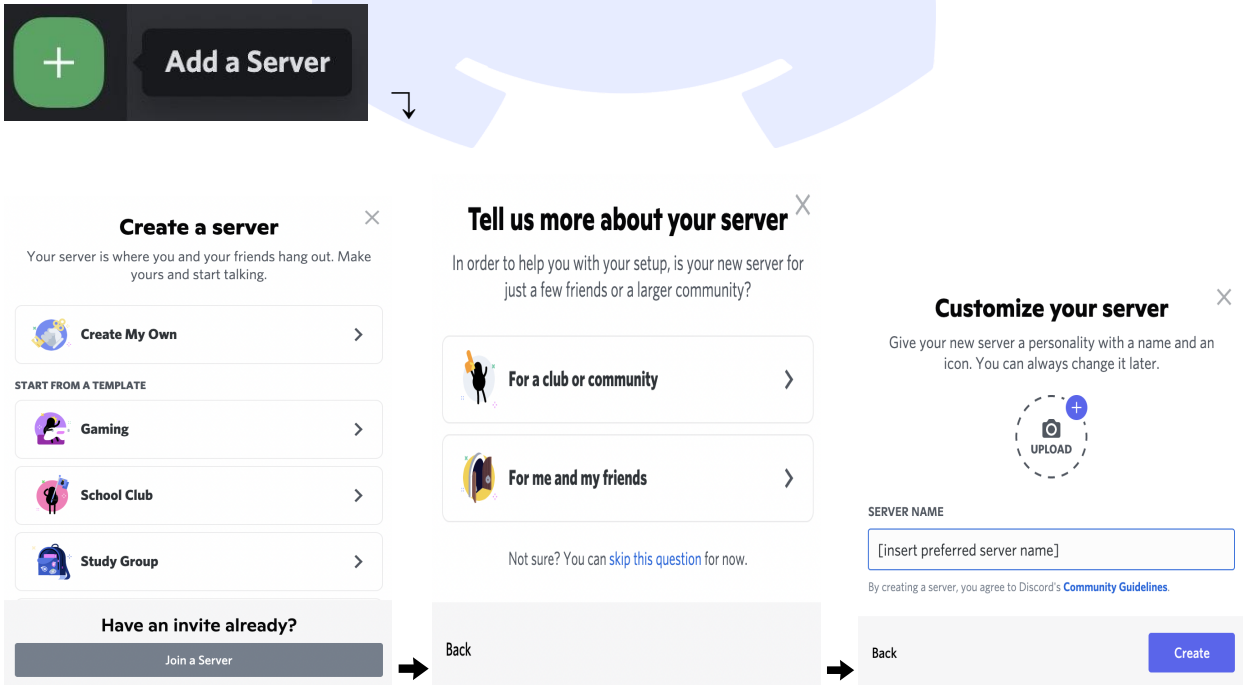
- My personal Discord server page with the Discord Bot I made embedded in the server.

Discord Server Page

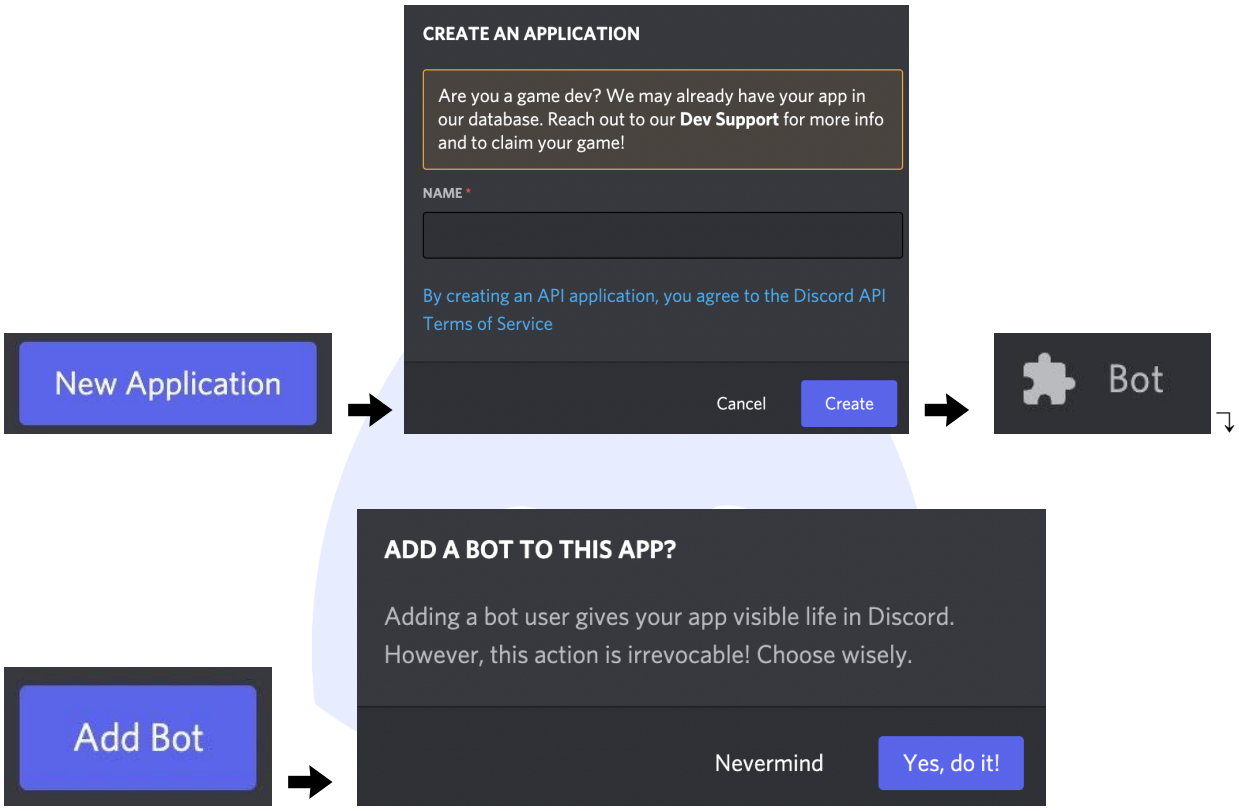
You can either use the Discord desktop app or open Discord through the web to log into your account and access your personal server. You can add the bot that I made by clicking the invite link that has been shared with you from the “OAuth2 URL Generator” by Discord Developer. When a new member joins the server they will automatically be greeted by the Discord Bot and will be given pointers on how to get to the list of commands presented by the Bot in a table. Lastly, you can use any of the commands presented to you as you please.

Setting up Discord

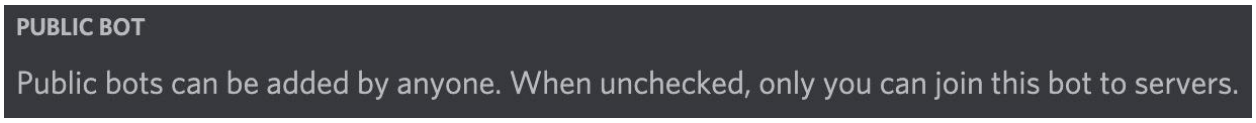
The first step in creating a Discord Bot is to make sure you have a Discord server to add the Bot to. You need to go to discord.com and log in to Discord and click the plus button to add a server. Then you can choose which category you want your server to be for. I went with “for me and my friends” and I called it ‘chellshe love’ server. Now we have created our server

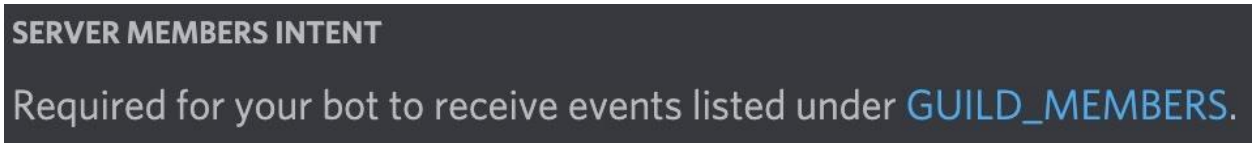


Next, you should make a Discord bot account right within Discord by going to this link "https://discord.com/developers/applications". Once you are on the applications page, you need to click on the "New Application" button and create a unique name for your app. After you created your application, go to the "Bot" tab and click on "Add Bot" then press "Yes, Do It!".

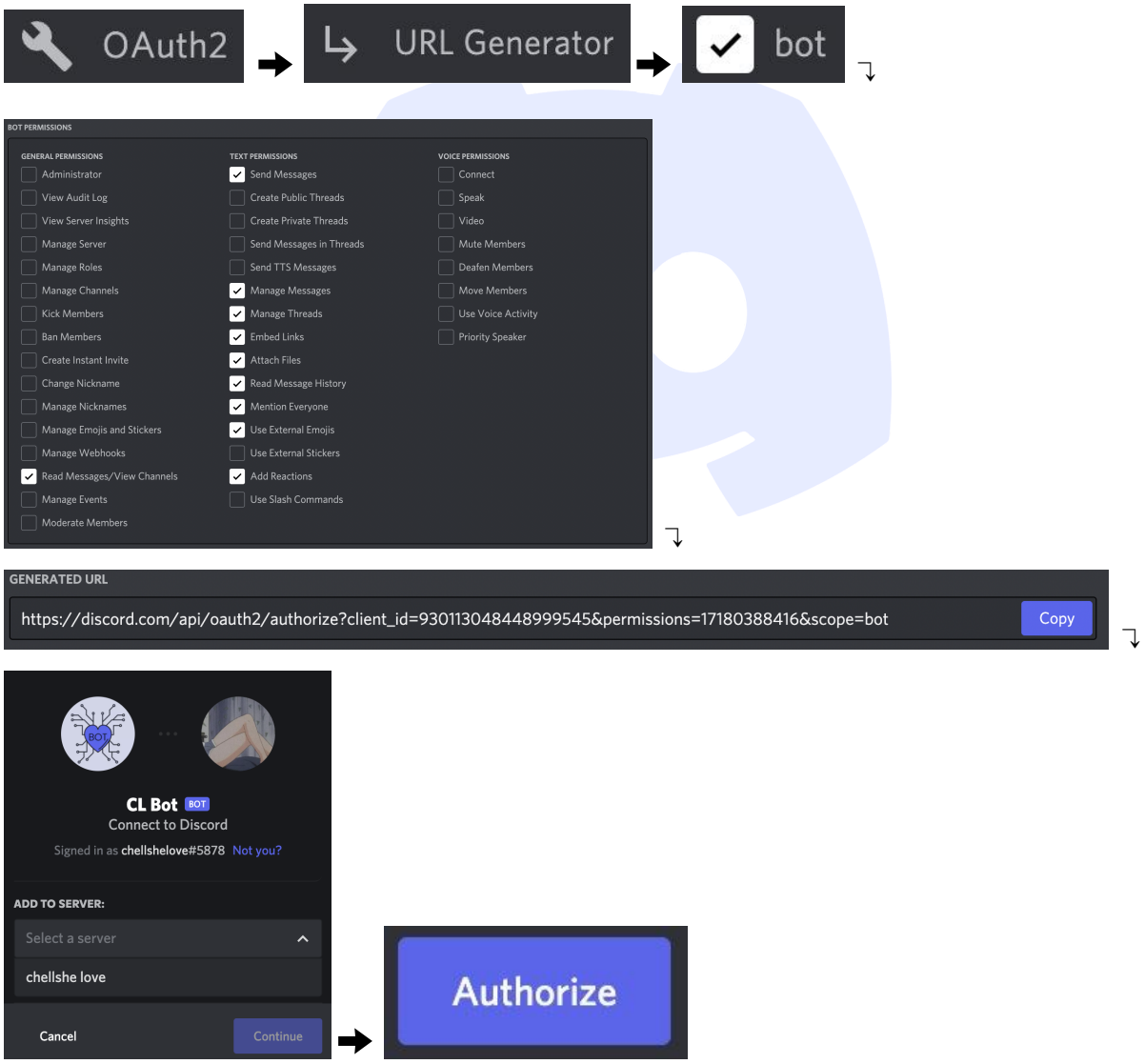


After successfully creating the Bot, you need to make sure that both of these settings are toggled on as they need to be up and running for the code to work later on.





Now we have to invite the Bot we just made into the server. In order to do that, we need to create an invite URL. Go to the “OAuth2” tab then go to the “URL Generator” and there you can see the scopes section. Tick the “Bot” box and the Bot permissions section will appear below. Tick all the permissions we want for our Bot. The Bot I made is mainly going to be used for text messages ergo I did not put a lot of permissions. After selecting the appropriate permissions, scroll to the bottom and copy the generated URL. Open a new tab and paste the URL and choose the server you want to add your Bot to. Press “Continue”, and “Authorize”. After the Bot is authorized, you can go back to your server and see that your bot has slid into the server. However, the Bot is still offline as we have yet to create a Python code that creates the Bot and runs it.



Implementation & Explanation of the Code

For this project, I used quite a lot of modules that I am somewhat new to: discord.py, os, requests, JSON and random. It was quite complicated and confusing when I first started doing it as I needed to learn what each module does for the code. I did not import all 5 modules at the same time in the very beginning. Instead, every time I think there needs to be a module imported, I will only then add it to the top with the rest mid code. There are a total of 275 lines on the main Discord_Bot.py file and 15 lines on the keep_alive.py file which I will be explaining further in this report.

```
import discord # I used the discord.py library therefore i need to import it by typing in "import discord"
import os # used so that the token can run in the main file
import requests # this module allows the code to make a HTTP request to get data form the API
import json # the API will then return JSON so that it will make it easier to work with the data when it's returned
import random # import a random module because the bot will choose a random starter encouragement
from replit import db # replit's built in database to store user submitted messages
from keep_alive import keep_alive # to make sure our bot keeps running even though we close our tab
# create an .env file on replit to save the token to prevent it from being shared
# to get the token, go to the discord developer website, go to your application and press the Bot tab then copy the token
my_secret = os.environ['token']
```

Since I enabled server member intent, I will therefore need to establish it in my code as well. You are going to get the default intents then I edited the members attribute. It is similar to saying “Hey let me get the information on who is joining my server and pass them through the client”.

```
intents = discord.Intents.default() # got to pass the intent after it is turned on from the bot setting in the discord developer website
intents.members = True
# create an instance of a client and it is part of the discord.py library, this is going to be the connection to discord
client = discord.Client(intents = intents)
```

I created a python list that contains sad words that the bot will respond to. Create a variable called sad_words that holds a list of my preferred sad words. Next, create a variable called starter_encouragements that holds a list of my preferred encouraging words that will be used to respond to the sad_words variable. It is called starter encouragements because users can add more encouragements to the list that is stored in the database.

```
sad_words = ["sad", "depressed", "unhappy", "miserable", "depressing", "lost", "rough"] # create a variable and put it in sad words

# make a variable and put it starter encouragements to be used to repsond to sad words
starter_encouragements = ["Cheer up!", "Hang in there :)", "You got this!", "You are a great person :)", "We are here for you <3", "Care to
```

Create a new key-value pair in the database. If “responding” is not in the database, we are going to add it and it is going to start off as being true.

```
22    if "responding" not in db.keys(): # new key in the database
23      db["responding"] = True # start off as true
```

I added a helper function called get_quote. This is a function we can call to return a quote from the API. First, I used the request module to request data from the API URL. I stored the response from the API in a variable called “response”, I made a get request for the zenquotes.io URL which will return a random quote. Then I converted that response to JSON. The response is going to have .text so that I can load it into JSON. Now we have to get the quote out of the JSON data, “q” in line 29 stands for a quote. Then I added the author of the quote after inserting the hyphen to put a space, “a” in line 29 stands for the author. Finally, you can return the quote.

```
25  def get_quote(): # helper function that we can call to return a quote from the API
26    # zenquotes.io API will generate random inspirational quotes, used the request module to request data from the API URL
27    response = requests.get("https://zenquotes.io/api/random")
28    json_data = json.loads(response.text) #convert the response in JSON
29    quote = json_data[0]["q"] + " -" + json_data[0]["a"] # getting the quote out from JSON
30    return quote # return the quote as a message on discord
```

The update_encouragements function is going to update the encouragements on the database. First, it is going to check if encouragements is a key in the database so if encouragement is in db.keys it just returns a list of the keys from the database. Therefore, if encouragement is already in the keys, it is going to a value from the database that is stored under a certain key. Now we just have to add the new encouragement to this list by using .append which is going to add something new to this list. Then after we append the new encouraging message to the old encouraging messages, we have to save it again into the database. If there is not already encouragements in the database, we are going to have to make it by making lines 38 and 39.

```
32  def update_encouragements(encouraging_message): # make a function that will update encouragments to the database
33    if "encouragements" in db.keys(): # check if encouragements is a key in the database
34      encouragements = db["encouragements"] # get the value form the database stored under a certain key
35      encouragements.append(encouraging_message) # add new encouraging message to the list
36      # after we add the new encouraging message to the old encouraging message we need to save it to the database
37      db["encouragements"] = encouragements
38    else:
39      db["encouragements"] = [encouraging_message]
```

I then created a new helper function that will delete an encouraging message and it is going to take an index as an argument. First, we need to get a list of the messages from the database in line 42. Now we have to check if the length of encouragements is more than the index. If the length of the encouragements is more than the index then we need to delete the encouragements from the index and we are going to save it in the database again.

```
41  def delete_encouragements(index): # a fucntion that will delet any added new encouragements
42      encouragements = db["encouragements"] # list all the messages from the database
43      if len(encouragements) > index: # check if the lenght of encouragements is more than the index
44          del encouragements[index] # if it is more then we are going to delete
45          db["encouragements"] = encouragements # save it into the database again
```

I used a client.event decorator to register an event; discord.py is an asynchronous library so things are done with callbacks. A callback is a function that is called when something else happens. In the code below, I used the on_ready event. This event is going to be called when the bot is ready to start being used. When the bot is ready, it is going to print line 51 to the console.

```
47  @client.event # used a client.event decorator to register an event, this client uses events to make it work
48  # this is also an asynchronous library on discord.py so things are done in call backs, a callback is a function that is called when something
49  # so the on_ready event is going to be called when the bot is ready to be used
50  async def on_ready():
51      print(f"{client.user.name} has connected to Discord!") # when the bot is ready it's going to print in the console
```

Register another client.event, I made an event that will greet the new members that have just joined the server. Then I established what channel I want to welcome the new member in. I did this by getting the server-id which can be obtained by going to the server setting, and then clicking the widget tab and copying the server-id. The reason as to why we need to obtain the server-id is due to the fact that bots are many servers and so when someone joins a server, the bot has to know which server to welcome them in (line 55). Next, we need to get a channel from the server, so essentially which channel we want to welcome them in (line 56). Right-click on the channel and click copy id. Then the last thing you need to do is to create the message you would like to send to the newly joined member through the server and private message (line 58 and 60).

```
53  @client.event # register event
54  async def on_member_join(member): # name of the event
55      guild = client.get_guild(930561768974073936) # get the server id so the bot knows which server its gonna welcome the new member in
56      channel = guild.get_channel(930561768974073940) # get the channel id so the bot knows which channel its gonna welcome the new member in
57      # the message that will be sent in the server
58      await channel.send(f"Welcome to the server {member.mention} ! :partying_face: Please type in 'help' so that I may assist you.")
59      # the message that will be sent through private message
60      await member.send(f"Welcome to {guild.name}'s server, {member.name}! :partying_face:")
```

The next event would be if the bot senses a message, it will then write in the discord server. This on_message event triggers each time a message is received. However, I do not want it to reply to anything if the message is from the bot itself, and so I made it check if the message is from ourselves (line 64). The next section of code is used to count the number of members that are in the particular server by using the same get_guild function on the previous section of code.

```
62  @client.event # register an event
63  async def on_message(message): # this on_message event triggers each time a message is recieved
64    if message.author == client.user: # we don't want it to do anything if the message is from ourselves, so we check using this line
65      return
66
67    guild = client.get_guild(930561768974073936) # get the server id
68    if message.content.startswith("users"):
69      # it will count the number of users in this particular server
70        await message.channel.send(f"Number of Members in this Server: {guild.member_count}")
```

The next thing I made is a table that embeds all the commands into one table. I made a command called “help” which will make the embedded table pop up whenever the user types it down. Inside the brackets in line 74, it has a title and a simple description of the contents of the table. The list of commands I made is what is going to be called fields. For the field it takes a name and a value, name as in the commands and value as in the output

```
72  # embedding all the commands into one clean table
73    if message.content.startswith("help"): # if the user types help a list of commands will be presented in an embedded table
74          embed = discord.Embed(title = "Help on BOT", description = "Some useful commands (please type with all lowercase or uppercase letters)
75          embed.add_field(name = "hello, hi, hey, hai, yo", value = "Greets the user")
76          embed.add_field(name = "wassup", value = "Bot tells you what's up")
77          embed.add_field(name = "how are you, how r u, hru", value = "Bot replies with how they're feeling")
78          embed.add_field(name = "aw", value = "Bot loves you")
79          embed.add_field(name = "users", value = "Prints out the number of people in the server")
80          embed.add_field(name = "thanks, thx", value = "The bot is glad to be of help")
81          embed.add_field(name = "i love you, i love u, i luv u, ily, ilu", value = "The bot will send love")
82          embed.add_field(name = "wish me luck", value = "The bot bids you good luck")
83          embed.add_field(name = "lol, lmao, haha, lmfao", value = "The bot laughs along")
84          embed.add_field(name="inspire", value = "Prints out random inspirational quotes")
85          embed.add_field(name="congrats", value = "Congratulates the user")
86          embed.add_field(name = "responding true", value = "Turns on encouraging bot response")
87          embed.add_field(name = "responding false", value = "Turns off encouraging bot response")
88          embed.add_field(name = "new", value = "Used to add new encouragements")
89          embed.add_field(name = "list", value = "Shows a list of newly added encouragements")
90          embed.add_field(name = "del", value = "Used to delete newly added encouragements")
91          await message.channel.send(content=None, embed=embed)
```

Here I am making up commands for the users to type in and what the Discord Bot will respond in return.

```
# make up commands for the users to type in and what will the discord bot respond in return
  if message.content.startswith("hello"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("HELLO"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("hi"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("HI"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("hey"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("hai"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("HAI"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("yo"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("YO"):
    await message.channel.send("Hello there, how are you doing ?")

  if message.content.startswith("wassup"):
    await message.channel.send("The sky lol")

  if message.content.startswith("WASSUP"):
    await message.channel.send("The sky lol")

  if message.content.startswith("how are you"):
    await message.channel.send("Good now that you're here")

  if message.content.startswith("HOW ARE YOU"):
    await message.channel.send("Good now that you're here")
```

```
  if message.content.startswith("how r u"):
    await message.channel.send("Good now that you're here")

  if message.content.startswith("HOW R U"):
    await message.channel.send("Good now that you're here")

  if message.content.startswith("hru"):
    await message.channel.send("Good now that you're here")

  if message.content.startswith("HRU"):
    await message.channel.send("Good now that you're here")

  if message.content.startswith("aw"):
    await message.channel.send(":smiling_face_with_3_hearts:")

  if message.content.startswith("AW"):
    await message.channel.send(":smiling_face_with_3_hearts:")

  if message.content.startswith("inspire"):
    quote = get_quote()
    await message.channel.send(quote)

  if message.content.startswith("INSPIRE"):
    quote = get_quote()
    await message.channel.send(quote)

  if message.content.startswith("congrats"):
    await message.channel.send("Congratulations! 🎉")

  if message.content.startswith("CONGRATS"):
    await message.channel.send("Congratulations! 🎉")

  if message.content.startswith("thanks"):
    await message.channel.send("The pleasure is all mine")

  if message.content.startswith("THANKS"):
    await message.channel.send("The pleasure is all mine")
```

```
171     if message.content.startswith("thx"):
172       await message.channel.send("The pleasure is all mine")
173
174     if message.content.startswith("THX"):
175       await message.channel.send("The pleasure is all mine")
176
177     if message.content.startswith("wish me luck"):
178       await message.channel.send("Good Luck! :muscle: :star_struck:")
179
180     if message.content.startswith("WISH ME LUCK"):
181       await message.channel.send("Good Luck! :muscle: :star_struck:")
182
183     if message.content.startswith("i love you"):
184       await message.channel.send("I love you more!!")
185
186     if message.content.startswith("I LOVE YOU"):
187       await message.channel.send("I love you more!!")
188
189     if message.content.startswith("i luv you"):
190       await message.channel.send("I love you more!!")
191
192     if message.content.startswith("I LUV YOU"):
193       await message.channel.send("I love you more!!")
194
195     if message.content.startswith("i love u"):
196       await message.channel.send("I love you more!!")
197
198     if message.content.startswith("I LOVE U"):
199       await message.channel.send("I love you more!!")
200
201     if message.content.startswith("ily"):
202       await message.channel.send("I love you more!!")
203
204     if message.content.startswith("ILY"):
205       await message.channel.send("I love you more!!")
206
207     if message.content.startswith("ilu"):
208       await message.channel.send("I love you more!!")
209
210     if message.content.startswith("ILU"):
211       await message.channel.send("I love you more!!")
```

```
213     if message.content.startswith("lol"):
214       await message.channel.send(":rofl:")
215
216     if message.content.startswith("LOL"):
217       await message.channel.send(":rofl:")
218
219     if message.content.startswith("lmao"):
220       await message.channel.send(":rofl:")
221
222     if message.content.startswith("LMAO"):
223       await message.channel.send(":rofl:")
224
225     if message.content.startswith("lmfao"):
226       await message.channel.send(":rofl: :rofl:")
227
228     if message.content.startswith("LMFAO"):
229       await message.channel.send(":rofl: :rofl:")
230
231     if message.content.startswith("haha"):
232       await message.channel.send(":rofl:")
233
234     if message.content.startswith("HAHA"):
235       await message.channel.send(":rofl:")
```

If database responding is true then it will respond to the sad words. Below is the code that made it possible to use the starter encouragements with the new encouragement that has been added to the database. And all of that is going to be inside the if statement of responding since if it is true, it will respond to the sad word, if not it will not respond to the sad word.

```
237     if db["responding"]: # if its true then it will respond to the sad words
238       options = starter_encouragements # create a new variable
239       if "encouragements" in db.keys(): # if there is any encouragement in the database
240         options = options + list(db["encouragements"]) # add them to the options of starter encouragements
```

If any word in the sad_words list is mentioned then it's going to respond with random starter encouragements that I have made.

```
242     if any(word in message.content for word in sad_words): # if the bot detects any of the words listed in sad words appear
243       await message.channel.send(random.choice(options)) # send a random starter encouragements
```

I made a new command called "new" that will allow the user to add in a new encouragement to the starter encouragement list. I split the message at new and added the 2nd element to the array because the 2nd element of the array is going to be the newly added message. Then I updated the encouragements with the newly added encouraging message. The next thing I made is a delete command that will delete the newly added encouragements if the user wanted to. Lastly for this section, I made a command that will show a list of all the newly added encouragements by the user so that it will be easier for them to see which one's they added and which ones are from the started encouragements.

```
245     if message.content.startswith("new"):
246       encouraging_message = message.content.split("new ", 1)[1]
247       update_encouragements(encouraging_message)
248       await message.channel.send("New encouraging message added.")
249
250     if message.content.startswith("del"):
251       encouragements = []
252       if "encouragements" in db.keys():
253         index = int(message.content.split("del", 1)[1])
254         delete_encouragements(index)
255         encouragements = db["encouragements"]
256       await message.channel.send(encouragements)
257
258     if message.content.startswith("list"):
259       encouragements = []
260       if "encouragements" in db.keys():
261         encouragements = db["encouragements"]
262       await message.channel.send(encouragements)
```

Now I added the feature of changing whether the bot is going to respond to sad words or not. The idea is that the user is going to type in “responding true” or “responding false” and that’s the discord message that the bot is going to receive as a command, the bot is going to respond by turning on or off the sad word function.

```
264     if message.content.startswith("responding"):
265       value = message.content.split("responding ", 1)[1] # get the value that the user typed in
266
267       if value.lower() == "true": # if its true then the bot will repsond to sad words
268         db["responding"] = True
269         await message.channel.send("Responding is on.")
270       if value.lower() == "false": # if its true then the bot will not repsond to sad words
271         db["responding"] = False
272         await message.channel.send("Responding is off.")
```

This is the line where we run the bot and where we keep the bot alive and running even though we might close our tabs. First, you need to make an .env file that will store your token for the bot, anything that we put in a .env file will not be made public and nobody will be able to see anything inside the .env file. The keep_alive command is used to run the server.

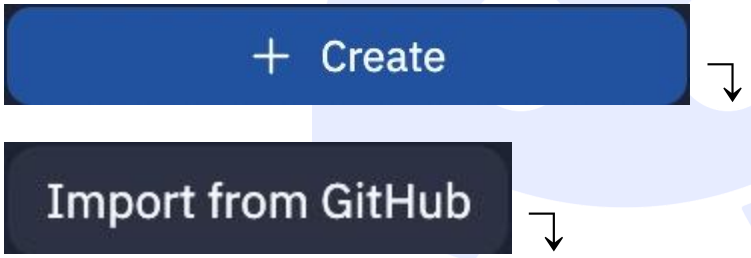
```
274   keep_alive() # this will run our web server
275   client.run(os.getenv("token")) # using import os we are running our bot using the token from the .env file through replit
```

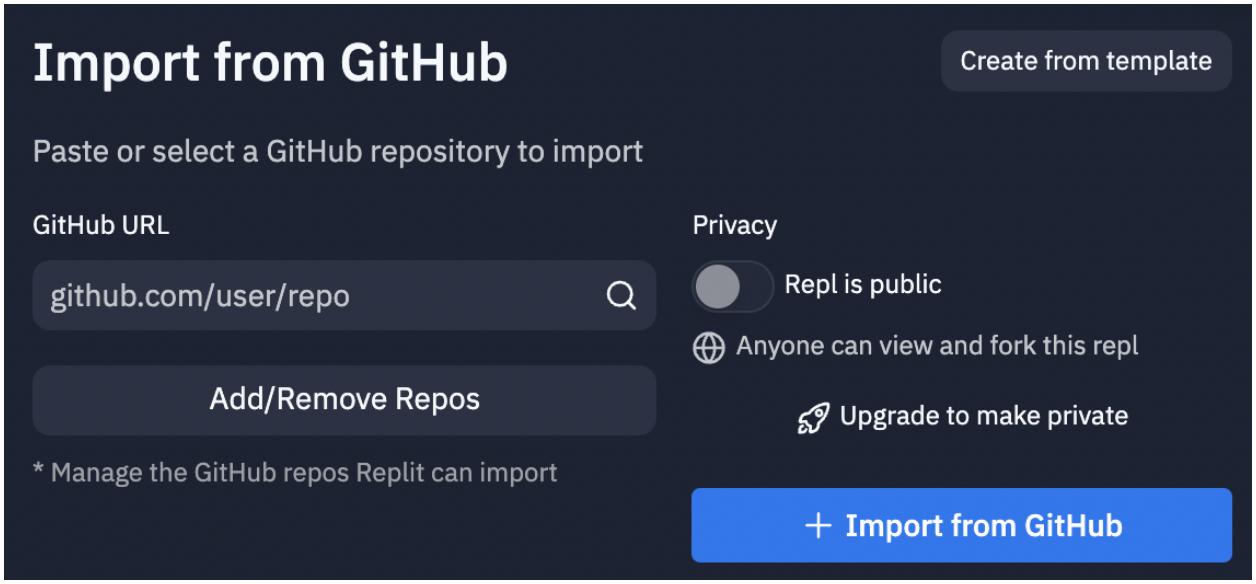
As for the keep_alive.py file, I made a web server on replit and set up uptime robot which I will explain what it is in the next section. I pasted the code from the provided web server. It’s going to return “hello, I am alive” to anyone who visits the server, the server will run on a separate thread from our bot so they can both be running at the same time. Essentially we just need the bot to run this webserver.

```
1   from flask import Flask # use flask as the web server
2   from threading import Thread
3
4   app = Flask('')
5
6   @app.route('/')
7   def home():
8       return "Hello. I am alive!" # return "hello i'm alive" to amyone who visits the server
9
10  def run():
11    app.run(host='0.0.0.0',port=8080)
12
13  def keep_alive():
14      t = Thread(target=run)
15      t.start()       You, seconds ago • Uncommitted changes
```

Setting up Replit

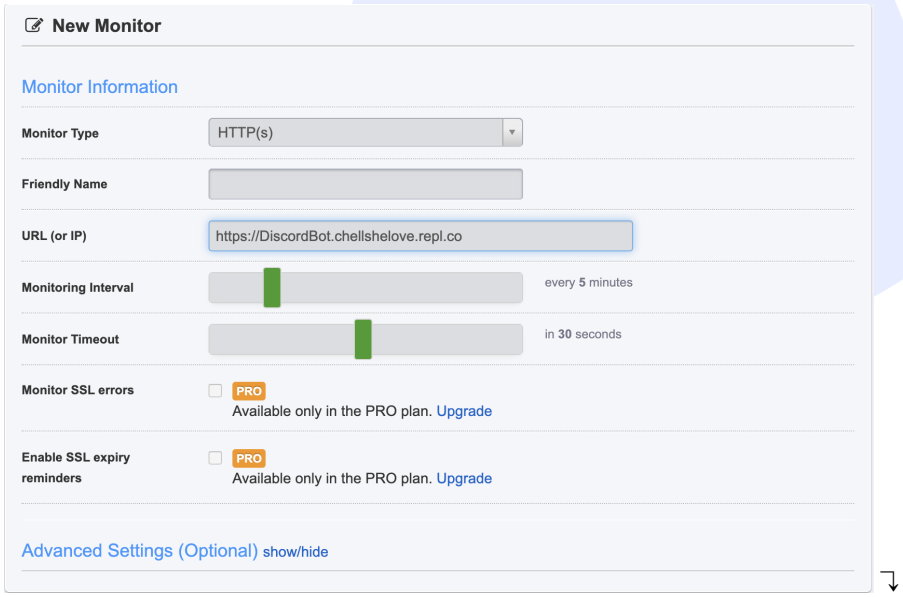
I imported the whole Discord Bot in the cloud on “Replit” (https://repl.it) as the bot will be completely hosted in this cloud. I can Replit is an online IDE that you can use in your web browser, so it makes it a lot easier to code things if you do not have to install any special software on your computer. Moreover, you can log into it from multiple browsers and computers as well as easy access to your code and your programs no matter what computer you are using, or even run the Bot on this website so that we do not have to run it on our local computer. This will make it so that we can keep the Bot running even if our computer shuts down or we close our browser tab. It can also clone GitHub Repositories which was what I did after typing the code on “Visual Studio Code”. Make sure you have logged into your account, press the tab button and press the “+ Create”. Then you would want to press the “Import from GitHub” button and choose the repository that you would like to clone. Finally, press the “+ Import from GitHub” button and your repository has been successfully cloned.





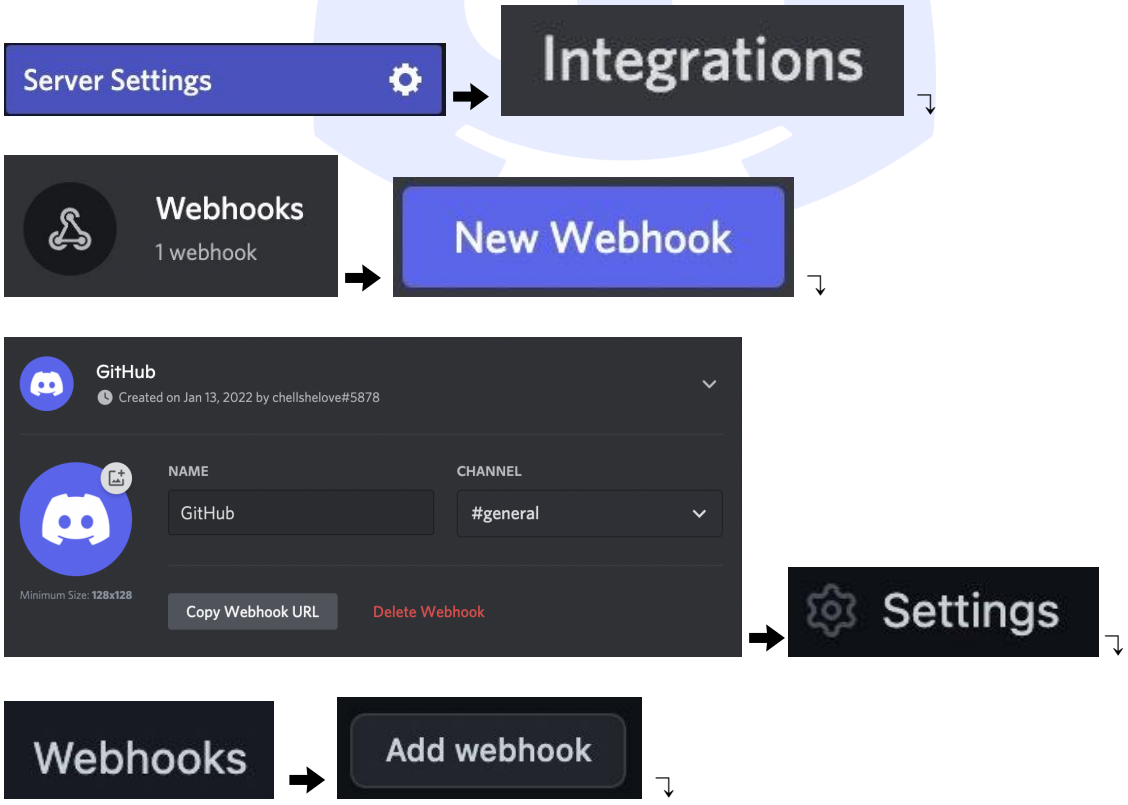
Setting up Uptime Robot

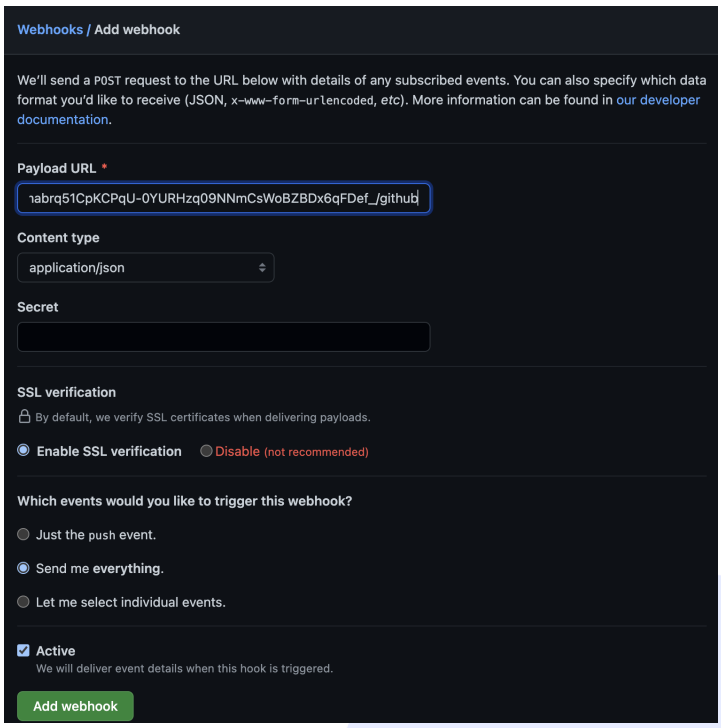
After the webserver on replit is up and running, you will then get a link on the top right corner of your replit window, copy the link. Then go to the uptime robot website and click the “+ Add New Monitor”. Then you select the monitor type to “HTTP(s)”, add a friendly name, and paste the link that you copied from replit. Click “Create Monitor” and “! Create Monitor (with no alert contact selected). Your bot should be up and running continuously so people can always interact with it on replit after you execute all the steps.



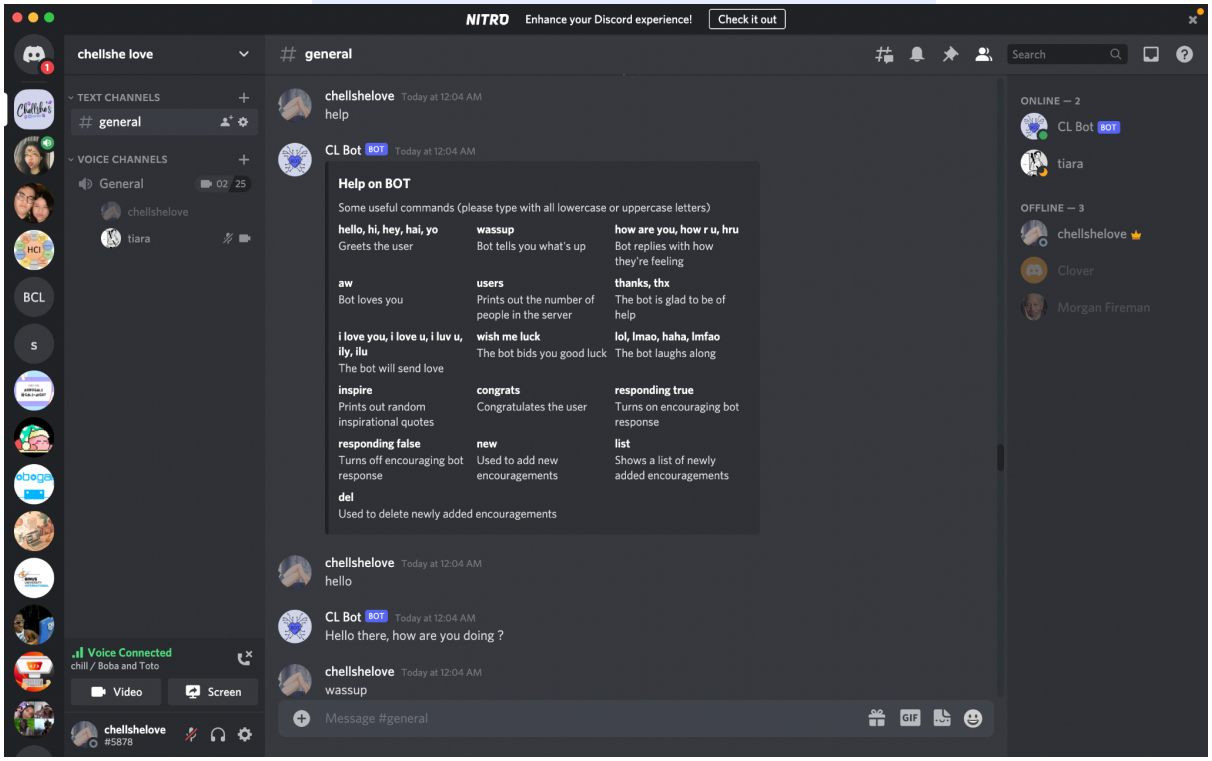
Bonus Bot

A Discord bot that works with webhooks. I connected the bot to GitHub so that the bot will post a message to your server whenever a certain GitHub repository is updated and for this GitHub bot, you don't even need to code anything or host it anywhere. To add this GitHub bot, we will go to our server, then we'll go to the server settings and then we'll see integrations and then there is a webhooks tab, click create webhook. I named mine GitHub and I put it in the general channel, then I'll just copy the webhook URL, make sure to save changes. Then I went over to the GitHub repo that I want to connect with this bot, we'll go to settings and then on the side tab it says webhooks, I'm going to click add webhook, then I'm just going to paste in the URL that I copied from Discord, here's an important part you need to add "/github" at the end of the URL and then for the content type I changed it to Applications/Json, then scroll down and for what event would you like to trigger this webhook, I choose send me everything so I'll add the webhook and now I can go back right over to my Discord channel to test if the bot works by updating the specific repository.





Proof of Working Program



Reflection & Experiences

At first glance, this project was quite tedious for me as I have no experience in programming before. It took me a full week to do research on what I wanted to do for this final project because I was having a hard time finding a doable idea that matched my current skillset. After days of contemplating I finally decided on making a Discord Bot. Prior to University, I can say that I have never used discord at all. I chose this idea for the sole reason that ever since I started University at Binus, I have been using Discord on a daily basis due to the fact that most of my friends use Discord as their main app to communicate and interact with each other.

I realized that making a Discord Bot is really confusing and complicated because I ran into some issues regarding creating the bot itself on Discord developer, importing some modules, finding errors, learning the commands from the discord.py library, making the bot run on the cloud and setting it up on uptime robot. I find myself getting a bit lost during the whole coding process whenever I encounter an error or if the code does not seem to make sense to me. Nonetheless, I did not lose hope and I kept trying to figure out what went wrong or kept re-reading the code until I am able to understand it well. Youtube played a huge role for me in making this project as it helped me figure the code out when I came across errors and it also shows how I can debug them.

Overall, this project has put me through a lot of ups and down but I still pulled through and managed to finish this whole project on time. I learned a lot of new things from making this project and hopefully, this will be beneficial for my future projects as I will be more efficient in doing it. Moreover, there are some things that I would change when I do my next project such as starting earlier and having better time management so that I don't feel the immense pressure looming over me because the due date closing in.

To summarize my experiences in creating my own Discord bot, I find it really difficult and time-consuming. A lot of issues might be encountered during the whole creating process of the Discord Bot but with a lot of persistence and determination, I was able to get this Discord bot to function according to the required specifications that I wanted.