1. **Description**

This project is only useful for a discord app user. The purpose of making this as my project is to make notification from git hub for people working in a group and posting their work on git hub. Other than that, it is for people to store their friends data as people may change their username often.

1. **Design**
2. **Explanation**

* Main file (pypy.py)

client = MyClient()

token = **"secret token"**

.

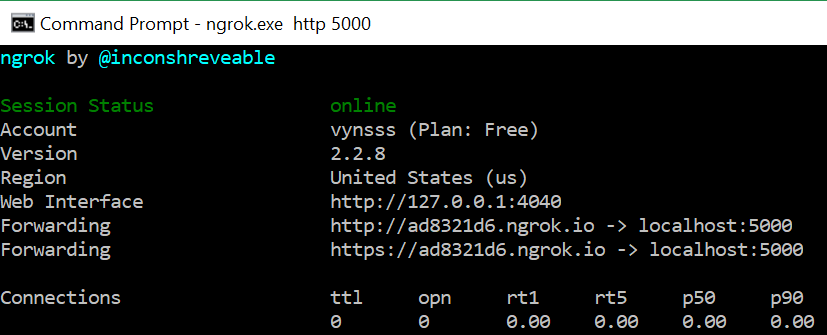
.

.

Client.run(token)

MyClient() is a class from the message\_content.py to handle the command message of the user to the bot. The token cannot be revealed or publicized to a third party as it’s a login for the bot to the discord server, and it is used to run the program using the client.run().

def flask\_app():  
 app = Flask(\_\_name\_\_)  
  
 @app.route(**'/github'**, methods=[**'POST'**])  
 def github\_webhook():  
 jmsg = request.json  
 msg\_send = jmsg[**'sender'**][**'login'**] + **' pushed to github'** loop.create\_task(client.send\_github\_notif(msg\_send))  
 print(jmsg)  
 return **'Successful'** print(**'Flask App Starting'**)  
 app.run()

flask\_app() is what connect the code to the git hub payload URL through the ngrok.

The forwarding link is copied and pasted to the payload URL in the webhook setting of the repository.

The (push) besides the link, indicates that it only update the push events happening in the repository.

jmsg (short for json message) get the data from the request.json. The jmsg[**'sender'**][**'login'**] in the msg\_send is from this part of the request.json:

'sender': {

'avatar\_url': 'https://avatars2.githubusercontent.com/u/43365946?v=4',

'following\_url': 'https://api.github.com/users/vynsss/following{/other\_user}',

'received\_events\_url': 'https://api.github.com/users/vynsss/received\_events',

'node\_id': 'MDQ6VXNlcjQzMzY1OTQ2',

'gravatar\_id': '',

'followers\_url': 'https://api.github.com/users/vynsss/followers',

'site\_admin': False,

'html\_url': 'https://github.com/vynsss',

'login': 'vynsss',

'gists\_url': 'https://api.github.com/users/vynsss/gists{/gist\_id}',

'events\_url': 'https://api.github.com/users/vynsss/events{/privacy}',

'url': 'https://api.github.com/users/vynsss',

'subscriptions\_url': 'https://api.github.com/users/vynsss/subscriptions',

'id': 43365946,

'repos\_url': 'https://api.github.com/users/vynsss/repos',

'type': 'User',

'starred\_url': 'https://api.github.com/users/vynsss/starred{/owner}{/repo}',

'organizations\_url': 'https://api.github.com/users/vynsss/orgs'

}

loop.create\_task(client.send\_github\_notif(msg\_send)) is too loop for every push event that is happening in the repository.

t = threading.Thread(target=flask\_app)  
t.start()

Threading is for the discord app and the flask\_app() to run side by side at the same time. Without the threading, the flask\_app() will not work.

* message\_content.py

async def send\_github\_notif(self, content):  
 #id from channel id in discord server  
 channel = discord.Object(id=**'508174256920723459'**)  
 await self.send\_message(channel, content)

This line of the code connect to the flask\_app() in the main python file, it send the notification message to the channel based on the channel id.

if message.content.upper().startswith(**".CURRENCY"**):  
 url = **"http://www.apilayer.net/api/live?access\_key=df1fa21b42994013fed11d8454508658&format=1"** response = requests.get(url)  
  
 spl = message.content.split(**" "**)  
 # the currency to convert from  
 convert\_from = str(spl[1]).upper()  
 # the currency to convert to  
 convert\_to = str(spl[2]).upper()  
 amount = float(spl[3])  
  
 #only need e.g. IDR as the input, not e.g. USDIDR  
 convert\_from = **"USD"** + convert\_from  
 convert\_to = **"USD"** + convert\_to  
  
 #https://stackoverflow.com/questions/44766282/accessing-json-api-with-python?rq=1  
 #checking for error  
 if response.status\_code != 200:  
 await self.send\_message(message.channel, **"error {}"**.format(response.status\_code))  
 else:  
 data = json.loads(response.text)  
  
 if convert\_from and convert\_to in data[**"quotes"**]:  
 from\_number = float(data[**"quotes"**][convert\_from])  
 to\_number = float(data[**"quotes"**][convert\_to])  
 total1 = **"%.2f"** % round(to\_number / from\_number \* amount, 2)  
 await self.send\_message(message.channel, **"total converted amount: {}"**.format(total1))  
 else:  
 await self.send\_message(message.channel,**"Sorry, the currency you are inputting are false or not available in the library"**)

One of the side feature of this program is the currency converter. For example, the input message content/input is .currency IDR USD 10000, the input is separated by “ “ to identify each word as part of the data. The IDR is the currency of the 10000 which the user wanted to convert it to USD. .upper() in the command is to make sure that if the input is in upper case or not, the data still can be read. The response.text is the library which contain the converted USD to other currency. (you can check the library here <http://www.apilayer.net/api/live?access_key=df1fa21b42994013fed11d8454508658&format=1>)

from user\_data import user  
  
userdata = {}

file = open(**"data.txt"**, **"r"**)  
data = file.read().split(**'**\n**'**)  
for f in data:  
 new = f.split(**","**)  
 userdata[new[0]] = user(new[0], new[1], new[2], new[3], new[4], new[5], new[6])  
file.close()  
  
def view(key):  
 d = userdata[key]  
 return(**"userID: {}, name: {}, username: {}, usernumber: {}, age: {}, school: {}, major: {}"**.format(d.userid, d.name,d.username ,d.usernumber, d.age,d.school,d.major))  
  
def write():  
 temp = **''** file = open(**"data.txt"**, **"w"**)  
 for d in userdata.values():  
 temp += d.userid + **","** + d.name + **","** + d.username + **","** + str(d.usernumber) + **","** + str(d.age) + **","** +d.school+**","**+ d.major + **'**\n**'** file.write(temp)  
 file.close()

This code is to make a function to access the data.txt. def view(key) is to view the data of people using discord, while def write() is to store the data that may have been removed, edited or added by the user.

if message.content.lower() == **".viewdata"**:  
 for d in userdata.keys():  
 await self.send\_message(message.channel, view(d))  
  
#search data from data.txt based on the userID  
if message.content.lower().startswith(**".searchdata"**):  
 search\_data = message.content.split(**" "**)  
 data = search\_data[1]  
 if data in userdata:  
 await self.send\_message(message.channel, view(data))  
  
#to add or edit data to userdata dictionary  
if message.content.lower().startswith(**".editdata"**):  
  
 add\_data = message.content.split(**","**)  
  
 userid = add\_data[1]  
 name = add\_data[2]  
 username = add\_data[3]  
 usernumber = add\_data[4]  
 age = add\_data[5]  
 school = add\_data[6]  
 major = add\_data[7]  
  
 userdata[userid] = user(userid, name, username, usernumber, age, school, major)  
 await self.send\_message(message.channel, **"The data successfully added or editted!"**)  
  
 write()  
  
#to remove data from data.txt  
if message.content.lower().startswith(**".remove"**):  
 remove\_data = message.content.split(**" "**)  
 number = str(remove\_data[1])  
 if number in userdata.keys():  
 userdata.pop(number)  
 await self.send\_message(message.channel, **"The data have been removed."**)  
 else:  
 await self.send\_message(message.channel, **"The data/user number u inputted is not available or an error occured"**)  
  
 write()

.viewdata is the command needed to view the data, it loop every data and print the returned value in def view(key). While the search data is to find a specific data based on the user ID which is the unique number of a user. (e.g. .searchdata 481443152964878347), it is also the key for the dictionary, the reason for making the user ID as a key is because the user ID is the unique number that only each person have, so even if the data of the person is the same, the data will not clash or change due to containing the same information.

The .editdata input (e.g. .editdata,12423584392,amar,amartyaa,1244,18,bi,cs) is separated by “,”, to make sure that the data like their full name that is separated by “ “ is not counted as a different data. The edit and add data use the same command as long as the key is the same, the data in the dictionary will automatically changed and added at the same time. As for the .remove command (e.g. .remove 12423584392) it will remove based on the key, so all the data in the dictionary which value is using the key inputted will be removed. The write() function is directly applied as the data is changed, so it will automatically saved the data.

* The class (user\_data.py)

class user():  
 def \_\_init\_\_(self, userid, name, username, usernumber, age, school, major):  
 self.userid = userid  
 self.name = name  
 self.username = username  
 self.usernumber = usernumber  
 self.age = age  
 self.school = school  
 self.major = major

This class is to help access the data.txt in the message\_content.py

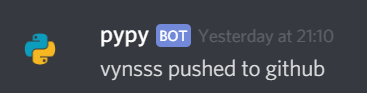
* data.txt

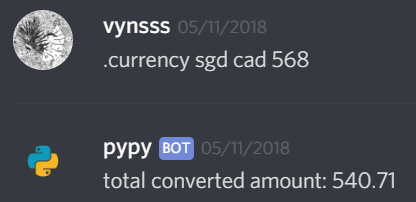
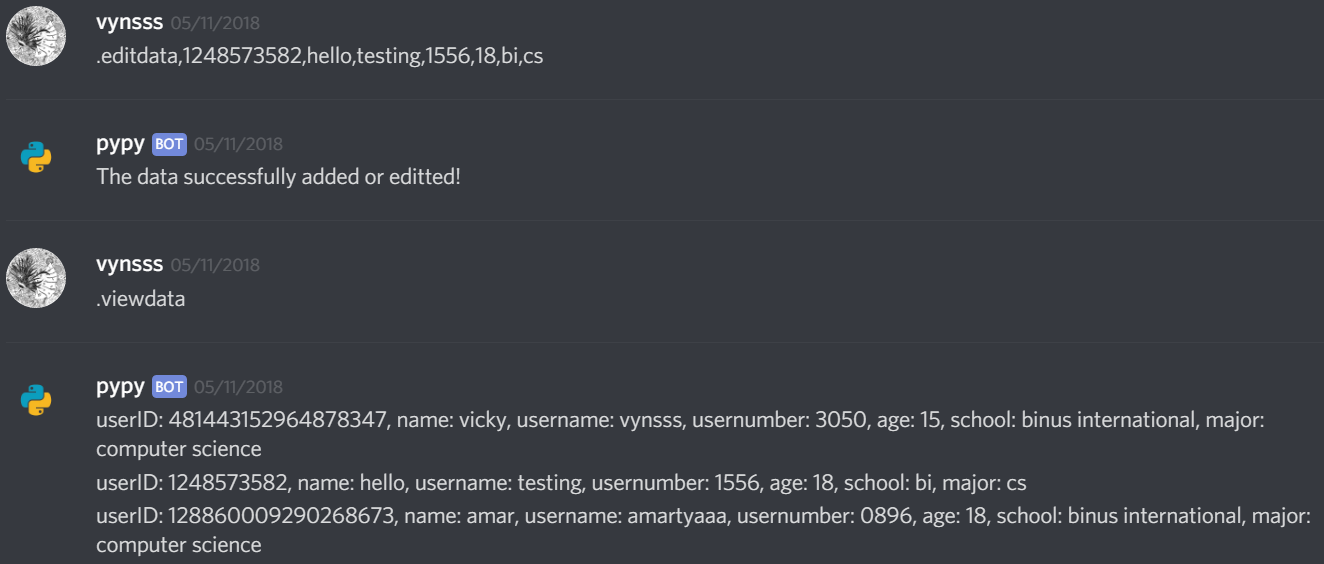
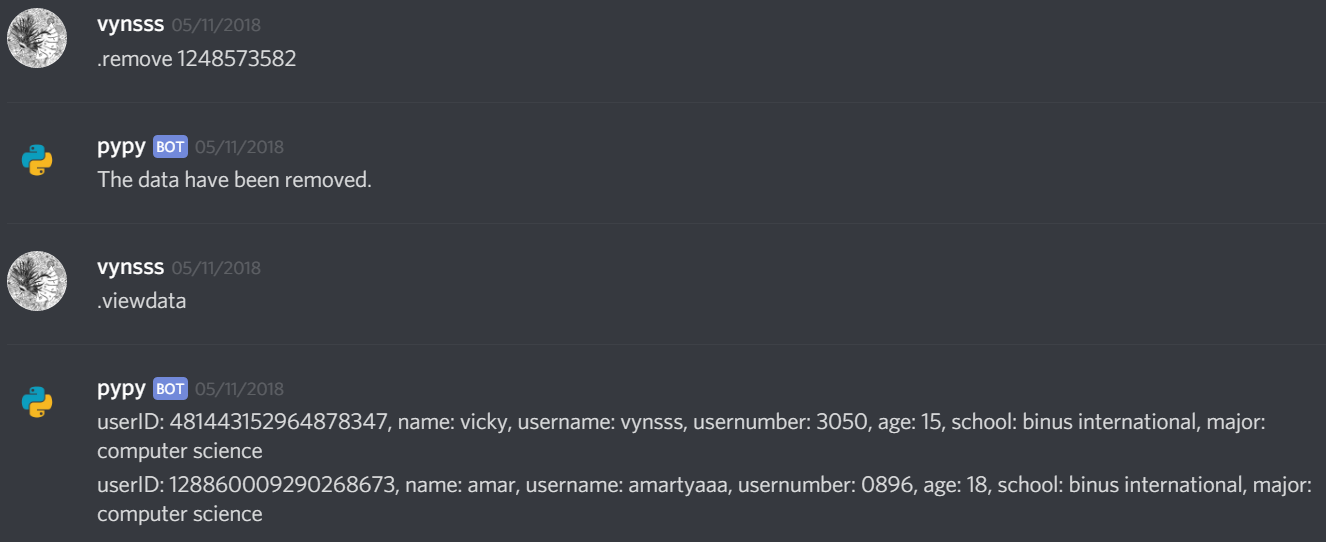
481443152964878347,vicky,vynsss,3050,15,binus international,computer science  
128860009290268673,amar,amartyaaa,0896,18,binus international,computer science

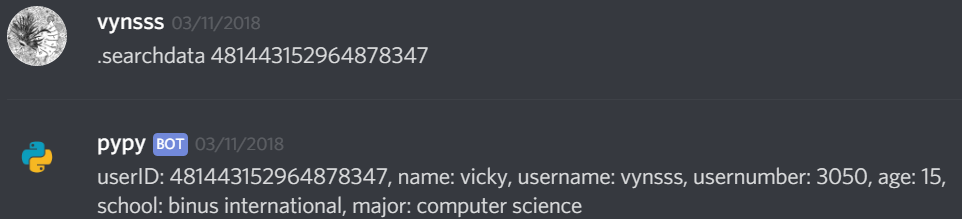
The content is separated by “,” so using the same reason as .editdata, it will not confuse “ “ as it’s separation value.

1. **Evidence**

* Git hub notification for push events



* Currency converter
* Accessing the data



1. **Resource**

* Documentation
  + <https://media.readthedocs.org/pdf/discordpy/latest/discordpy.pdf>
* Tutorial
  + <https://www.youtube.com/watch?v=_0LXIvLDhBM&t=356s>
  + <https://github.com/FoggyIO/DiscordPythonBots>
  + Flask - <https://www.youtube.com/watch?v=YMBzb_RBDAA>
* Currency converter
  + <https://currency-api.appspot.com/>
  + <https://stackoverflow.com/questions/44766282/accessing-json-api-with-python?rq=1>