BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:		**Surname**	**Given Names**	**Student ID Number**
	1.	**Vanessa**	**Vicky**	**2201807791**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1BC**	**Name of Lecturer(s)**	: Monica Hidajat
Major	**: CS**		
Title of Assignment (if any)	: Discord Bot		
Type of Assignment	**: Final Project**		
Submission Pattern			
Due Date	**: 20 – 11 - 2018**	**Submission Date**	**: 19 – 11 -2018**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

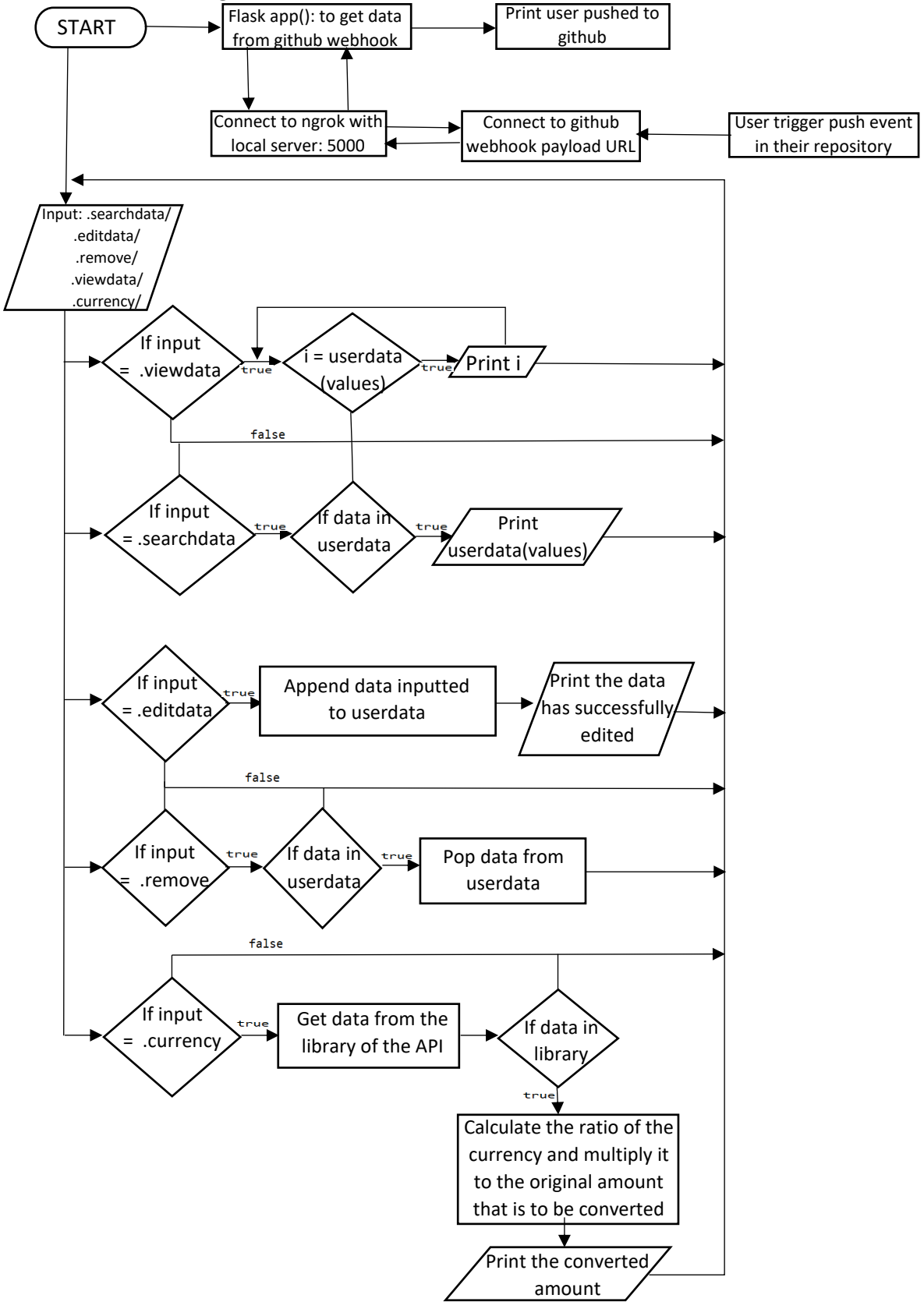
By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)
Vicky Vanessa

I. 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.

II. Design



III. Discussion

Implementation:

To make this project run, it highly rely the API such as discord.py, asyncio, flask, as well as the others. It also use ngrok to make it possible to run it.

Discord.py, as the name suggest, this API is for python use. This API allow the bot to take control of the discord application and the server which it is in. It have a lot of function which help the discord bot to control the server or the member.

Asyncio was used to handle the message in the server which the discord bot is in. This is to make sure that the respond of the discord bot is not overlapping with it's other respond.

Request was used to get/request data based on the URL given to them. The json file is to store the requested data and help the user to access it's data.

Flask is used to handle the webhook request, similar to request, it gets the data from the webhook payload URL through ngrok.

Ngrok is a program to connect the local server to the online server in the internet. It helps this project to connect the local server to the webhook payload URL in git hub.

How it work:

- Main file (pypy.py)

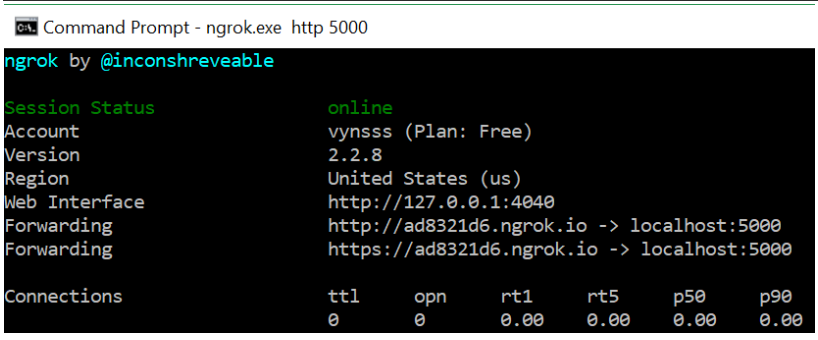
```
client = MyClient()
token = "secret token"
.
.
.
Client.run(token)
```

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 is 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 ngrok. Ngrok is used for connecting the local sever to the online server which is the discord.

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. So every time someone pushed to git hub, the flask_app() will get the data from the git hub and through the payload URL it will send the data to the local host which will be sent to the discord server.

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 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, this function send the notification message to the channel based on the channel id in the server which the discord bot is in.

```
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 currency. .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 contain the library which contain the data of 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 and save 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
```

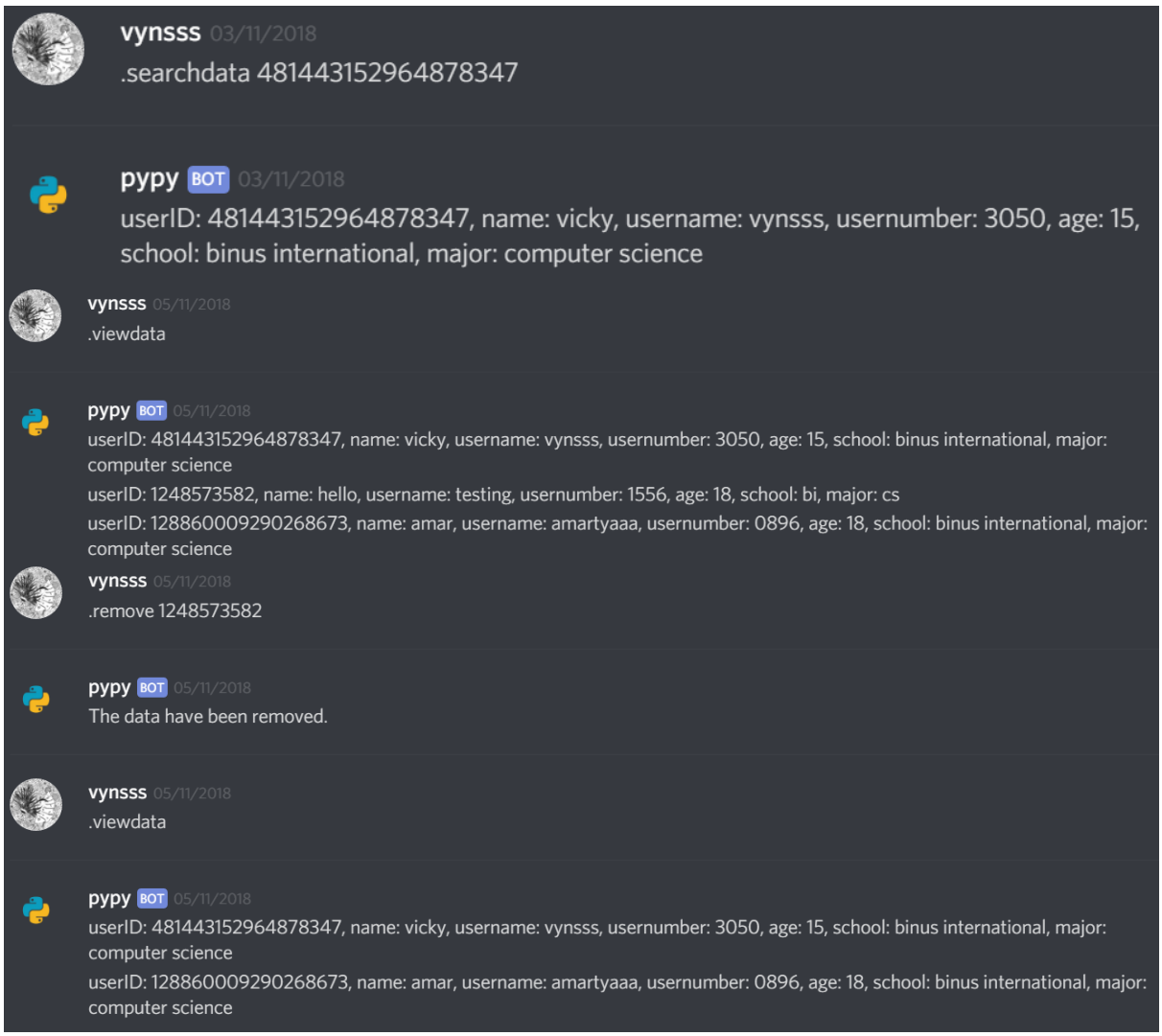
The content is separated by “,” so using the same reason as .editdata, it will not confuse “ “ as it’s separation value and can store multiple words data.

IV. Evidence

- Git hub notification for push events

- Currency converter

- Accessing the data



V. 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