assignment report

I use python 3.7.9 for this assignment

Data structure design

credential

there is a file called credential.txt, which is used for Authentication

the struct is

username1 password1

username2 password2

user data

there is a file called `userData.json`

it records the offline message, blacklist, start time, end time and it's IP and port number

Every we set up a new server , `userData.json` file will be renewed

the struct is

```
    "username": {
        "message": ["", ""],
        "blackList": ["", ""],
        "active_period": [
            {
                "start":
                "end":
            }
        ],
        "clientAddress": [IP, port]
    }
```

program design

server

The server will record

- users be blocked
- online user
- client threads
- port been used

The server can

- Authentication
- Message
- Notifications

- Find users online
- Find online history
- show Blacklist to client

client

client is responsible to receive data from server and communicate with user

Client have two thread

- input thread

  this thread keeps listening to the user's input,

  when the thread receive a input, it will respond to server or to a p2p-connection client

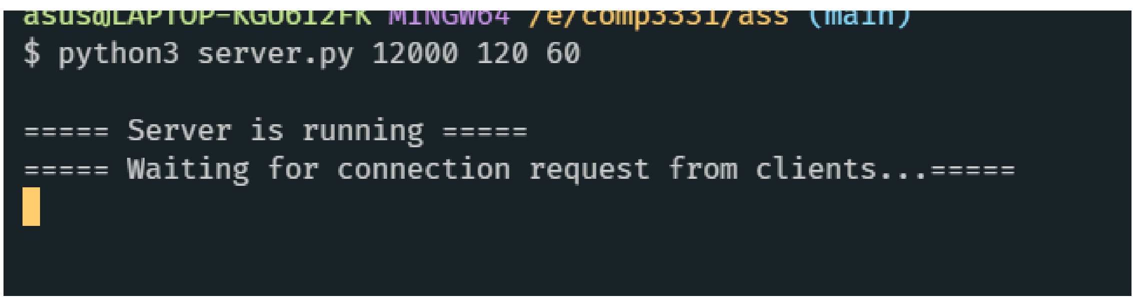
- receive thread

  this thread keeps listening to the server

each p2p connection is connected, then there will have one more thread which keeps listening to the p2p-connection client

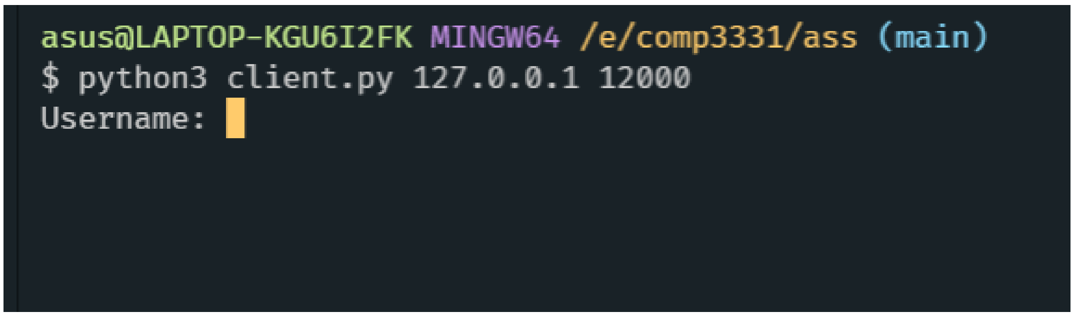
how system works

- set up server



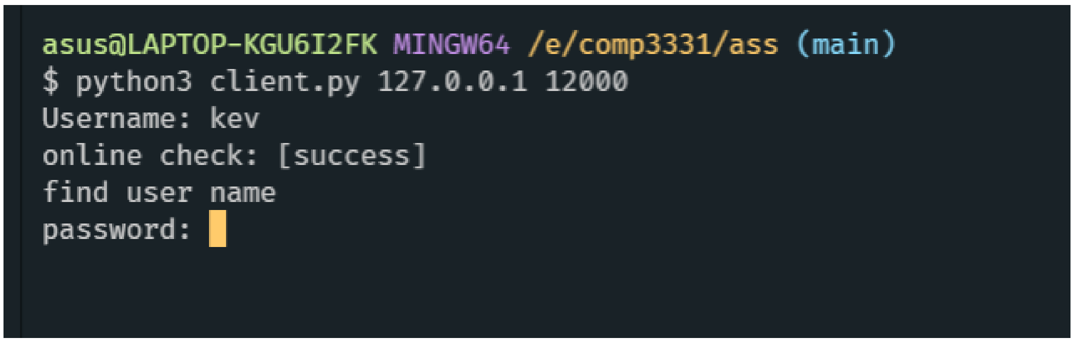
```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 server.py 12000 120 60

===== Server is running =====
===== Waiting for connection request from clients...=====
```

- set up client
  - input user name



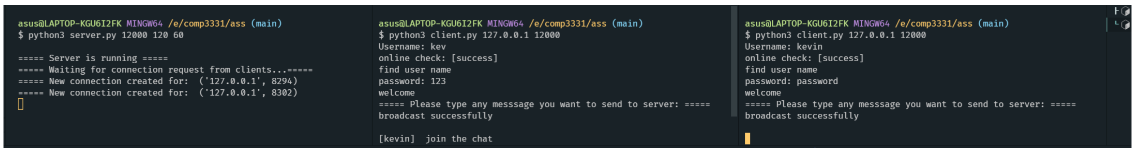
```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 client.py 127.0.0.1 12000
Username: 
```

  - input the password



```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 client.py 127.0.0.1 12000
Username: kev
online check: [success]
find user name
password: 
```

you can have multiple user

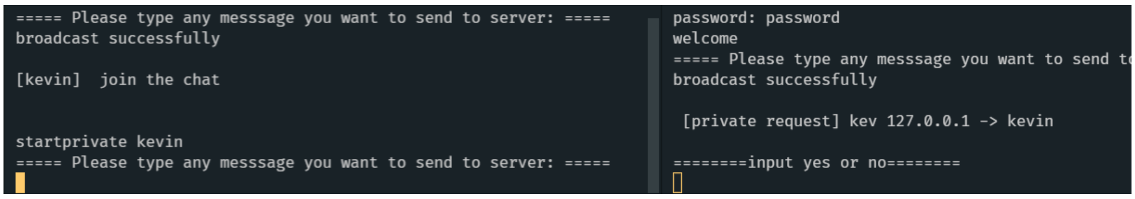


```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 server.py 12000 120 60

===== Server is running =====
===== Waiting for connection request from clients...=====
===== New connection created for:  ('127.0.0.1', 8294)
===== New connection created for:  ('127.0.0.1', 8302)
```

```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 client.py 127.0.0.1 12000
Username: kev
online check: [success]
find user name
password: 123
welcome
===== Please type any messsage you want to send to server: =====
broadcast successfully

[kevin]  join the chat
```

```
asus@LAPTOP-KGU6I2FK MINGW64 /e/comp3331/ass (main)
$ python3 client.py 127.0.0.1 12000
Username: kevin
online check: [success]
find user name
password: password
welcome
===== Please type any messsage you want to send to server: =====
broadcast successfully
```

- when you want to create a p2p connection, remember to input the `yes` or `no`



```
===== Please type any messsage you want to send to server: =====
broadcast successfully

[kevin]  join the chat


startprivate kevin
===== Please type any messsage you want to send to server: =====
```

```
password: password
welcome
===== Please type any messsage you want to send t
broadcast successfully

 [private request] kev 127.0.0.1 -> kevin

========input yes or no========
```

the other system design is similar to the assignment specification

errors

- in the `credentials.txt` there may have some test user name and password, delete it before test
- logout broadcast in private not finish
- sometime there will have some json error, it happens rarely(I don't know it still exists or not)

Application Layer Protocol

messages the clients and server exchange

- server
  - sending:
    - by `clientsocket.send()`
    - it will auto reply to the client's message
  - receiving:
    - by `clientsocket.recv(1024)`
    - it will split the receiver's data by `<space>`, it will keep tracking the first keyword, and do the corresponding reply
- client
  - sending:
  -
    - by `clientsocket.send()`
    - it will send message corresponding to user input to the server
    - it will read the input from user and auto communication with server
  - receiving:
    - by `clientsocket.recv(1024)`
    - receive user input
    - receive server reply

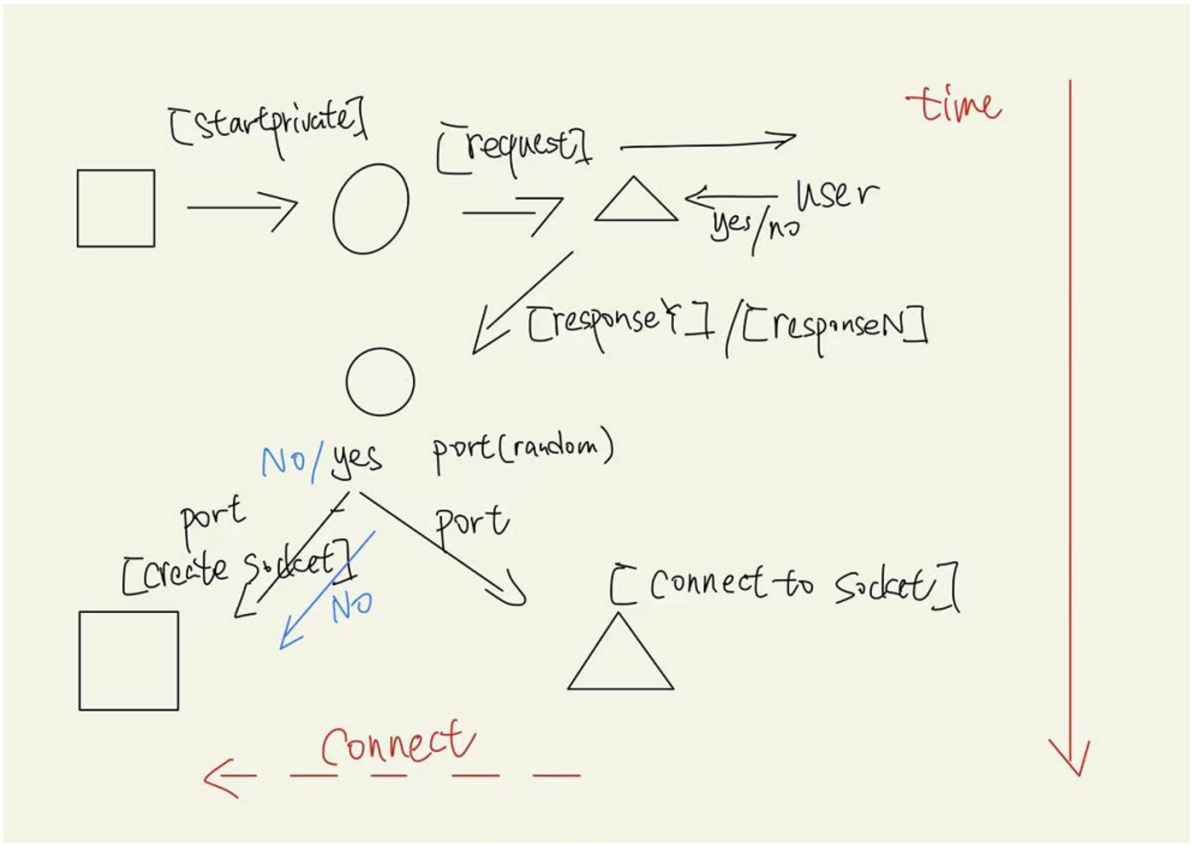
actions

setup

1. the server keeps accepting new client, create a new thread when a new client connected, and the server will auto reply to clients
2. Then the server will do Authentication with client

3. Then client will create two thread for receiving message from server and for receiving input from the user
    - the thread listen to user will detect `Peer to peer Messaging(private()) and stopprivate()`
    - the thread listen to server will detect `startprivate()`
4. the client will ask the server for the offline message
5. the client will tell server to broadcast the presence
6. message exchanging start

private



the square is client1 who want to start a p2p connection with triangle (client 2)

the circle is server

When the triangle receive `[request]`, a global variable `privateMessage` will be changed in `ReceiveServer(Thread)`, it will tell `InputThread(Thread)` to receive a `yes/no`