

# Report

## How to run?

First make sure you are in parent folder of 'src', typically (3331\_ass1\_messenger).

To run server, type

```
'python3 -m src.server.main SERVER_PORT MAX_CLIENT'.
```

To run client, open another terminal, type

```
'python3 -m src.client.main SERVER_IP SERVER_PORT CLIENT_UDP_PORT'
```

NOTE: client1 folder does not do anything with the functionality, it is for your convenience to test p2pvideo command. It is simply a duplicate of client. (testing mentioned p2pvideo will be tested in different folder for different client).

## How it works?

Client talks to Server via TCP and receive private/group message and file. Server holds different Thread for different client, process command and send response.

## Program Design

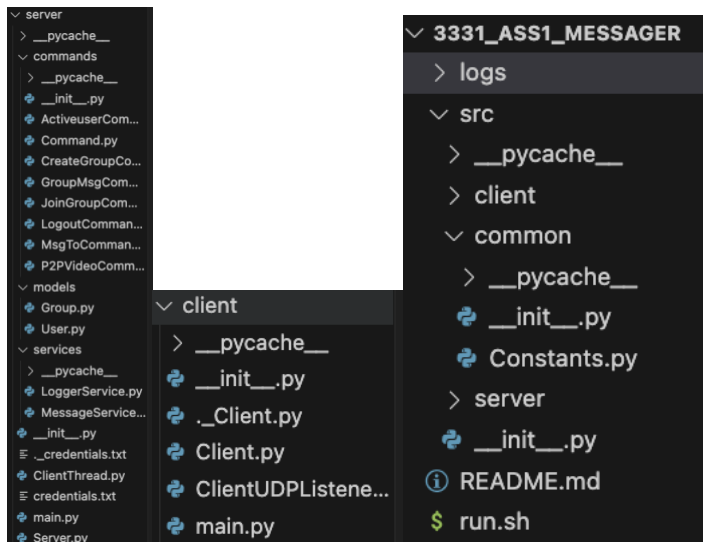
### Structure

My program uses Object Oriented Design.

In src folder, there are server and client, which each has a main. Server and client shares the Constants in common folder.

In client folder, main.py is to boot the client and control the run. clientUDPlistener will handle all incoming private or group message.

In server folder, main.py is to boot the server, Server.py in charge of welcomeSocket and create thread for each client. ClientThread.py is to handle client command. In models folder, we have Group class and User class, these two are for building object. In services folder, logger has all logging methods and messenger has all sending message methods. I used command pattern design here. In commands folder, command.py is a super class, the rest specific commands are subclasses. Each command is corresponded to a class and has method 'execute'. The client thread will create command class based on incoming command and invoke execute. Then, command will process user request and send response.



## Variable Storage

In `server.py`, it contains three Sets. `ApprovedUserSet` is for storing the active Users. `BannedUserSet` is for temporarily store banned user, the reason please see *login* section. `GroupSet` is to store all groups created. Each group will contain a list named `groupMembers`, this list will store user who joined the group.

## Login and Logout

One important notice is resetting after ten seconds is based on user request other than timer. The user is not banned until `MAX_ATTEMPTS` reached. if user closed the terminal, attempts reset. if user typed two times wrong and waited for 10 seconds, attempts won't reset. if user reached the limit, get banned and get unbanned when user send a new request after 10 seconds.

Logout will remove user from all groups joined.

## Application Layer Message Format

### Sample format for user log:

*1; 14 Oct 2023 22:51:09; username; 127.0.0.1; 57025*

### Sample format for /activeuser:

*username, 2023-10-15 11:34:05.625095, 127.0.0.1, 56846*

### Sample format for sending confirmation:

*PRIVATE\_MSG - No: 1, Timestamp: 15 Oct 2023 11:35:31*

*GROUP\_MSG - No: 1, Timestamp: 15 Oct 2023 11:36:28*

### Sample format for private Message

*15/10/2023 11:35, username: hello*

### Sample format for group message

*15/10/2023 11:36, groupname: hello*

## Design Trade-off

Although OOP design is much more complex than putting everything in `client.py` and `server.py`, it is much handier to maintain and add new features. E.g., I just need to add a new class when creating a new command instead of going deep down to the code and change. Message sending, I simply used the specific header without statuses code. That is because this program is relatively small and no need to consider further.


## Potential Improvements


My structure is a bit messy. I should abstract more methods into corresponding class. E.g., user can have `joingroup` and `sendmessage` methods. These can be encapsulated into `User` class. For future design, state pattern might be needed for active and inactive user state.

## Proof Of Development


Below are my commies.


Oct 10, 2023

 **adding UDP, changed User**  
Nuodi Liu authored 4 days ago


 **finished authenticate**  
Nuodi Liu authored 5 days ago


Oct 08, 2023

 **added function toprocess command**  
Nuodi Liu authored 6 days ago


 **finsihed logging**  
Nuodi Liu authored 1 week ago


Oct 03, 2023

 **add script to run, import bug unfixed**  
Nuodi Liu authored 1 week ago

 **finished autenticate, waiting for testing**  
Nuodi Liu authored 1 week ago

Oct 02, 2023

 **created client class**  
Nuodi Liu authored 1 week ago


 **built basic struct of server, implemented autenticate**  
Nuodi Liu authored 1 week ago


Nuodi Liu > 3331\_ass1\_messenger > Commits

main ▾


3331\_ass1\_messenger

Oct 15, 2023


 **added model folder**  
Nuodi Liu authored 1 hour ago


 **finished p2p, waiting for test**  
Nuodi Liu authored 11 hours ago


Oct 14, 2023


 **fixed a bug when delete active user when logging out**  
Nuodi Liu authored 13 hours ago


finished logout  
Nuodi Liu authored 13 hours ago


 **added more to logout, neally finished**  
Nuodi Liu authored 13 hours ago

 **added code for logout**  
Nuodi Liu authored 13 hours ago


 **fixed bugs in groupchat msg**  
Nuodi Liu authored 14 hours ago


 **finished sending group chat**  
Nuodi Liu authored 15 hours ago

 **finished joining group**  
Nuodi Liu authored 20 hours ago


 **finished create group**  
Nuodi Liu authored 21 hours ago


Oct 13, 2023

 **finished activeuser**  
Nuodi Liu authored 2 days ago

 **finished msgto, tested invalid argu & inactive user**  
Nuodi Liu authored 2 days ago

Oct 11, 2023

 **added UDP listening, msg can be sent, both client cannot receive**  
Nuodi Liu authored 3 days ago

 **modified server to close connection when client reach maxattemp, added listener to client**  
Nuodi Liu authored 3 days ago