1. 구현 기능

1)

TCP를 기반으로 구현된 Python library인 'socket'을 사용하여, 서버가 가동되고 있는 한 무제한으로 메시지를 주고받을 수 있습니다. 사용자가 채팅방에 접속할 때에는 '*userName*님이 입장하셨습니다' 라는 메시지가 채팅방에 출력되며, 채팅방에서 직접 나갈 때에는 '*userName*님이 퇴장하셨습니다.' 라는 메시지가 채팅방에 출력됩니다.

2)

1대1 채팅이 아닌, 다중 접속이 가능한 채팅방입니다. 채팅방 입장 시 입력한 이름이 곧 채팅 닉네임이자, password의 역할을 합니다. 동일 이름의 사용자의 접속을 막지는 않지만, 사용자 구분 기준이 닉네임인 관계로, 재접속 여부를 구분하는 데에 혼선이 있을 수 있습니다. 이를 방지하기 위해 닉네임은 서로 다른 이름을 사용하는 것을 권장합니다.

3)

크기가 1024byte 미만인 .txt 파일에 한하여 채팅방 이용자 간에 파일 전송이 가능합니다. 파일은 전송자를 포함한 채팅방 참가자 전원에게 전송되며, 기존의 *fileName*은 "To*recieverName*_From*senderName*_*fileName*"의 형태로 변환되어서 전송됩니다.

4)

채팅방을 이용하던 중, 연결이 두절된 경우 '*userName*님이 접속이 끊겼습니다.'라는 메시지가 채팅방에 출력됩니다. 연결이 두절된 사용자에게는 다른 이용자들이 보내는 채팅 메시지와 파일이 전송되지 않습니다.

5)

연결을 복구하고 재접속할 경우, '*userName*님이 재접속했습니다'라는 메시지가 채팅방에 출력됩니다. 단, *userName*이 password의 역할을 하기에, 재접속 시 기존에 사용하던 동일한 *userName*을 입력하고 접속을 해야 재접속으로 인정이 됩니다. 그렇지 않을 경우 신규 접속자로 판단됩니다. 채팅방에 재접속할 경우, 그때부터 다시 다른 이용자들이 보내는 채팅 메시지와 파일을 전송받을 수 있습니다.

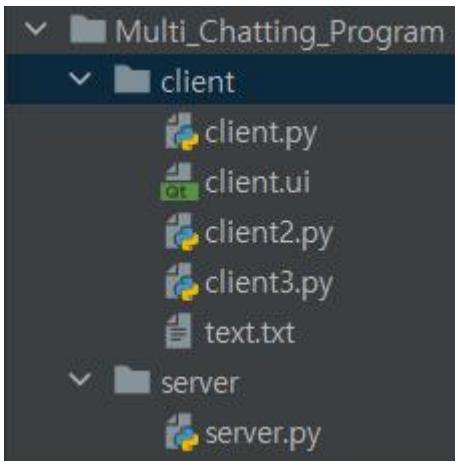
2. 개발 환경

Client side는 Front-end와 Back-end로 구성되어있으며, Server side는 Back-end로만 구성되어 있습니다. 각각의 Back-end는 Python을 이용하여 개발을 진행하였으며, Client side의 Front-end는 PyQt5 library를 이용하여 개발을 진행했습니다. Pycharm IDE를 통해 local 개발 작업을 진행하였고, private git repository를 만들어서 remote에 코드를 저장하면서 코드 관리를 했습니다. 고정 ip를 사용할 수 없는 개발 환경의 한계로 인하여 단일 장치에서 개발을 진행하며 통신 테스트를 하였습니다. 그 결과, ip는 'localhost' 또는 '127.0.0.1'을 사용했으며, port는 임의로 지정한 '8080'을 사용했습니다. 이 프로그램은 단일 장치 내에서의 실행에 한하여 원활한 작동이 보장됩니다. 프로젝트의 팀원은 2명이며, 유영서는 Server side Back-end와 Client side Back-end를 담당했으며, 김연웅은 Client side Front-end와 Client side Back-end를 담당했습니다.

3. 실행 방법

1)

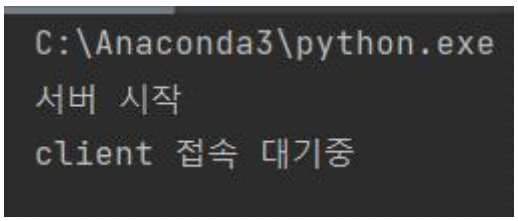
제출된 압축 폴더에서 'Multi_Chatting_Program.zip' 파일을 꺼낸 후, 압축을 풀어줍니다. 압축을 해제하면 다음 그림과 같은 구성을 보이게 됩니다.



client2.py와 clinet3.py는 client.py와 동일한 내용의 파일이며, 원활한 테스트를 위해 같이 첨부했습니다. text.txt는 파일 전송 테스트를 위한 파일입니다.

2)

우선, server.py를 실행시켜 줍니다. 서버가 정상적으로 실행이 되었다면, 다음 그림과 같은 메시지를 출력하게 됩니다.



3)

다음으로 client.py를 실행시켜 줍니다. 클라이언트가 정상적으로 실행이 되었다면, 다음과 같은 UI가 나타나게 됩니다.

4)

IP에 '127.0.0.1', PORT에 '8080'을 넣어주고, NAME에는 원하는 닉네임을 입력한 다음, '접속하기' 버튼을 눌러주면 됩니다.

5)

접속에 성공하였다면, UI와 프로그램에 다음과 같은 메시지가 출력됩니다. 우선, UI는 다음과 같은 결과를 보여줍니다.

다음으로, client.py는 다음과 같은 결과를 보여줍니다.

```
C:\Anaconda3\python.exe
수신 thread 가동
수신 대기중
서버와 연결했습니다
수신 대기중
```

마지막으로, server.py는 다음과 같은 결과를 보여줍니다.

```
('127.0.0.1', 63378) 접속 성공
client 접속 대기중
유영서한테서 받은 메세지: 유영서
유영서에게 전송
"/text"
유영서에게 전송
"유영서님이 입장하셨습니다"
```

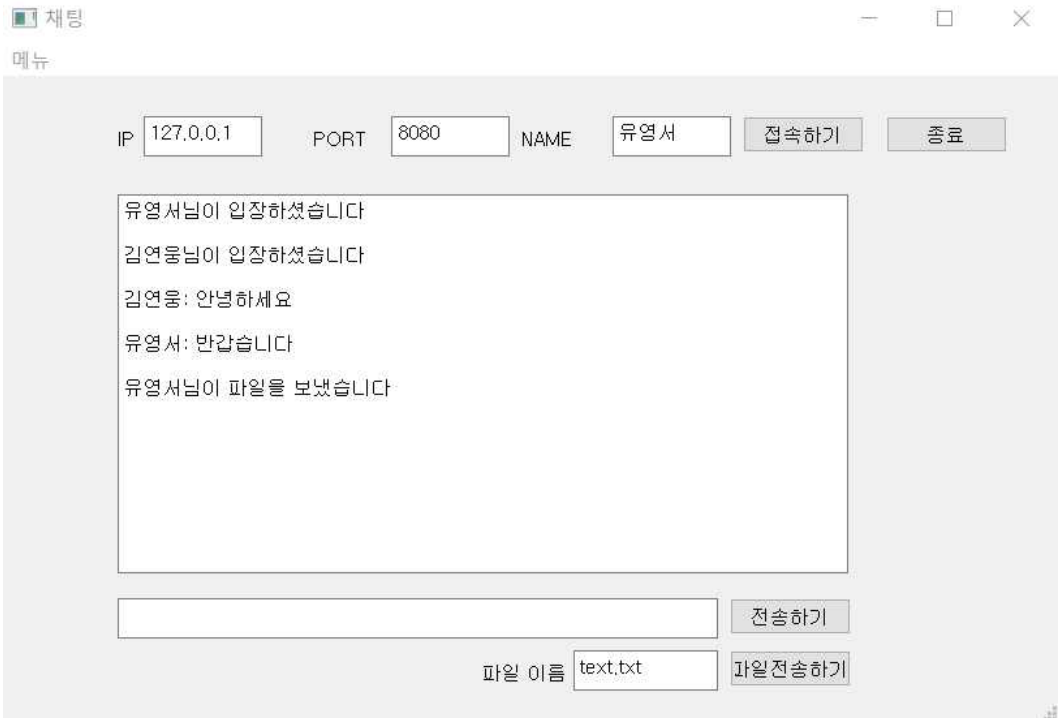
위 과정이 순조롭게 진행되었다면, client가 정상적으로 server에 접속한 것입니다. 같은 방식으로 여러명의 client가 하나의 채팅방 서버에 접속할 수 있습니다.

6)

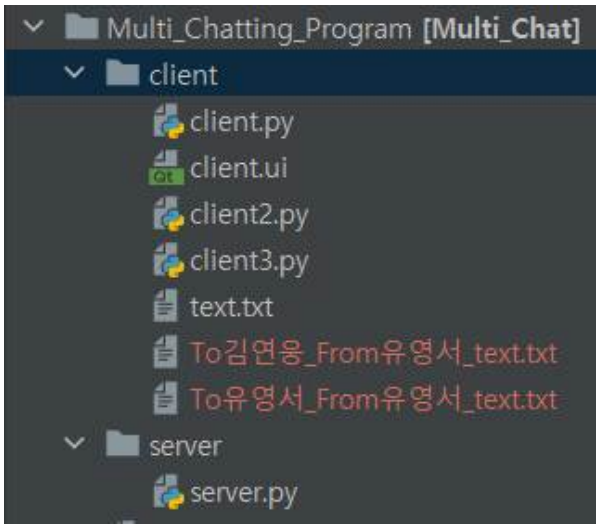
채팅방에 보낼 메시지를 입력한 후, ‘전송하기’ 버튼을 누르면 채팅에 참여중인 모든 이용자들에게 메시지가 전달됩니다.

7)

전송하고자 하는 파일 이름을 입력한 후, '파일전송하기' 버튼을 누르면 채팅에 참여중인 모든 이용자들에게 파일이 전송됩니다. 단, 전송할 파일은 client.py 파일과 같은 경로에 위치해야 하며, 수신되는 파일의 저장 경로 또한 client.py 파일과 같은 경로입니다.



파일명은 "To*recieverName*_From*senderName*_*fileName*"의 형태로 전송이 되며, 정상적으로 파일 전송이 이루어지면 다음과 같은 결과를 보입니다.



파일 전송자한테도 자신이 보낸 파일이 수신되기에, 파일이 제대로 전송되었는지를 직접 확인해 볼 수 있습니다.