COMPUTER NETWORKS
(UE22CS252B)
MINI PROJECT

FILE TRANSFER SYSTEM

AAKANKSH SEELIN(PES2UG22CS003)

AAKRITI MISHRA(PES2UG22CS004)

Client.py:

client.py | Win64OpenSSL_Light-3_2_1.exe | text.txt

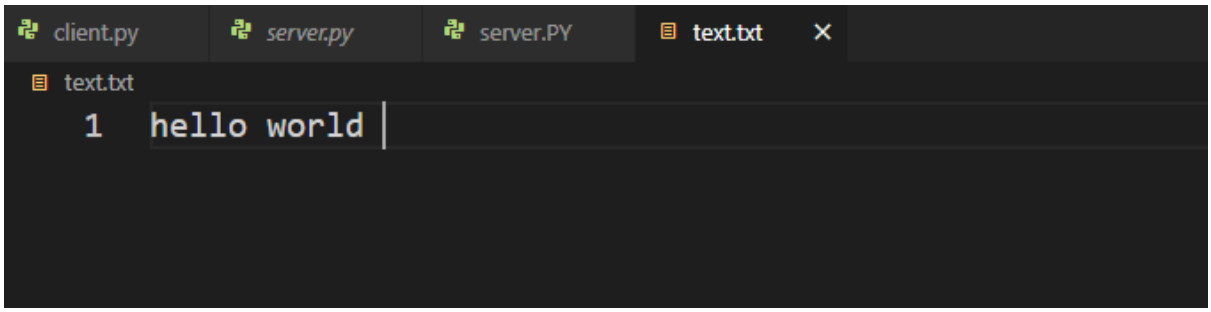
client.py > ...

```python
import socket
import ssl
import os

def send_file_to_server(filename, server_ip, server_port):
    context = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
    context.check_hostname = False
    context.verify_mode = ssl.CERT_NONE

    with socket.create_connection((server_ip, server_port)) as sock:
        with context.wrap_socket(sock, server_hostname=server_ip) as ssl_sock:
            with open(filename, 'rb') as f:
                file_data = f.read()
                ssl_sock.sendall(file_data)

if __name__ == '__main__':
    server_ip = '10.1.19.138'  # Server IP address (same as server)
    server_port = 8080       # Port number (same as server)

    while True:
        filename = input('Enter the filename to send (or type "exit" to quit): ')
        if filename.lower() == 'exit':
            break
        try:
            send_file_to_server(filename, server_ip, server_port)
            print(f'Successfully sent {filename} to the server.')
        except FileNotFoundError:
            print(f'File {filename} not found. Please try again.')

    print('Client connection terminated.')

```

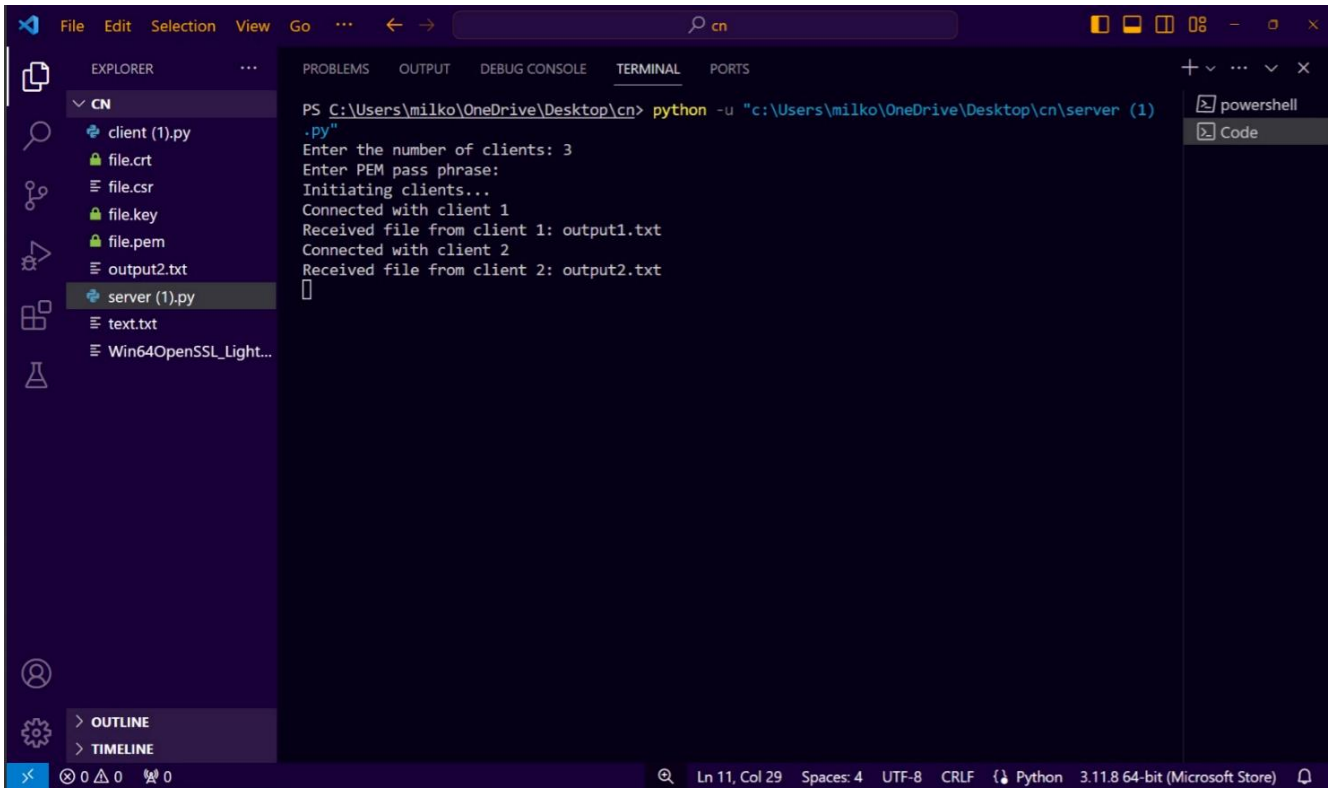
Server.py:

```python
import socket
import ssl
import threading

def handle_client(conn, fileno):
    try:
        filename = f'output{fileno}.txt'
        with open(filename, "wb") as fo:
            while True:
                data = conn.recv(1024)
                if not data:
                    break
                fo.write(data)
        print(f'Received file from client {fileno}: {filename}')
    finally:
        conn.close()

if __name__ == '__main__':
    host = '10.1.19.138'  # Server IP address (localhost for testing)
    port = 8080           # Port number

    total_clients = int(input('Enter the number of clients: '))
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
    context.load_cert_chain('file.crt', 'file.key')

    ssl_sock = context.wrap_socket(sock, server_side=True)
    ssl_sock.bind((host, port))
    ssl_sock.listen(total_clients)

    connections = []
    print('Initiating clients...')
    try:
        for i in range(total_clients):
            conn, _ = ssl_sock.accept()
            connections.append(conn)
            print(f'Connected with client {i + 1}')
            threading.Thread(target=handle_client, args=(conn, i + 1)).start()
    finally:
        ssl_sock.close()

    print('All files received successfully!')
```

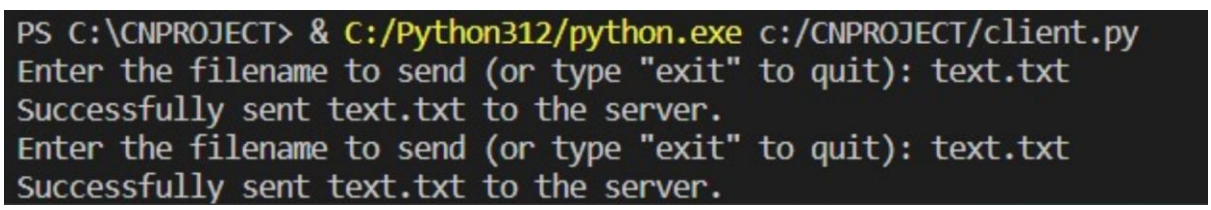
Text.txt on client side:



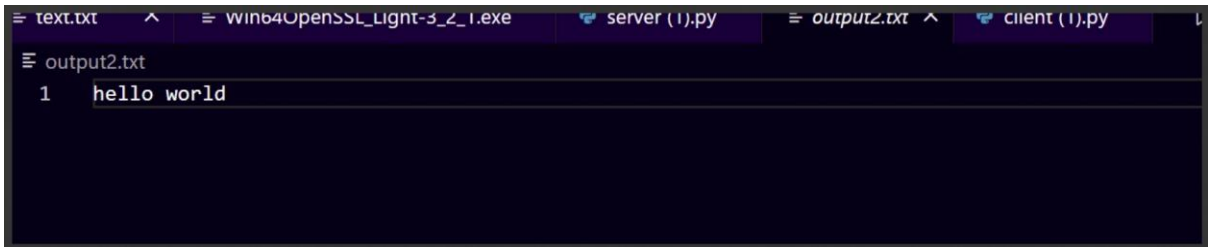
Server running:



Client side:



Text.txt received on server side:



SSL certificate: