CODE AND OUTPUT

Creating a server and client setup using OpenSSL for certificate-based authentication involves generating certificates, setting up the server, and configuring the client.

Below are the steps to achieve this on a Windows operating system using Command Prompt (CMD) commands.

Please note that OpenSSL needs to be installed on your system for these commands to work.

Step 1:

1. Generate Certificates
2. Open Command Prompt.
3. Navigate to a directory where you want to store your certificates using the cd command.
4. Generate a private key for the Certification Authority (CA):

```
openssl genpkey -algorithm RSA -out ca.key
```

5. Generate a self-signed certificate for the CA:

```
openssl req -new -x509 -key ca.key -out ca.crt
```

6. Generate a private key for the server:

```
openssl genpkey -algorithm RSA -out server.key
```

7. Create a certificate signing request (CSR) for the server:

```
openssl req -new -key server.key -out server.csr
```

8. Sign the server's CSR with the CA's key to generate the server certificate:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -out server.crt -CA
-CAcreateserial
```

9. Generate a private key for the client:

```
openssl genpkey -algorithm RSA -out client.key
```

10. Create a CSR for the client:

```
openssl req -new -key client.key -out client.csr
```

11. Sign the client's CSR with the CA's key to generate the client certificate:

```
openssl x509 -req -in client.csr -CA ca.crt -CAkey ca.key -out client.crt -
```

```
client.crt -CAcreateserial
```

Step 2:

1. Set Up Server
2. Create a new file named server.py and add your server code.
3. Use the server.crt and server.key files in your server code to enable SSL/TLS.

Step 3:

1. Configure Client
2. Create a new file named client.py and add your client code.
3. Use the client.crt and client.key files in your client code to enable SSL/TLS.

Step 4:

1. Add Server Code to the server.py file and change the IP address accordingly.
2. Add Client Code to the client.py file and change the IP Address to the server Address.
3. Run server.py and then client.py to make connections from client to server.

- **Code for Server Client Connections to share Messages from Local Client’s Device.**

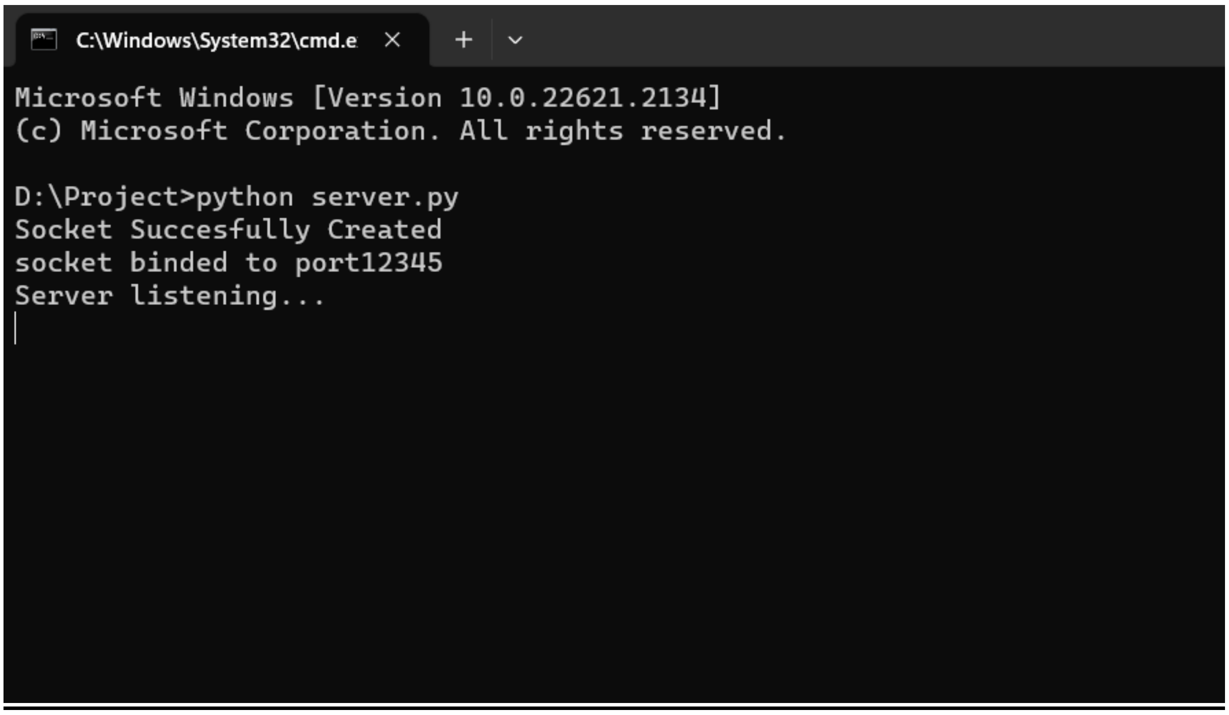
Server.py

```
import socket
import ssl


server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)


print("Socket Succesfully Created")
server.bind(('127.0.0.1', 12345))
server.listen(5)


context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain(certfile='server.crt', keyfile='server.key')
print(f'socket binded to port{12345}')
print("Server listening...")


while True:


    client, addr = server.accept()
    print("Got Connection from", addr)


    ssl_client = context.wrap_socket(client, server_side=True)
    data = ssl_client.recv(1024)
    print(f"Received: {data.decode()}")


    ssl_client.send("Hello from server!".encode())
    ssl_client.close()
```

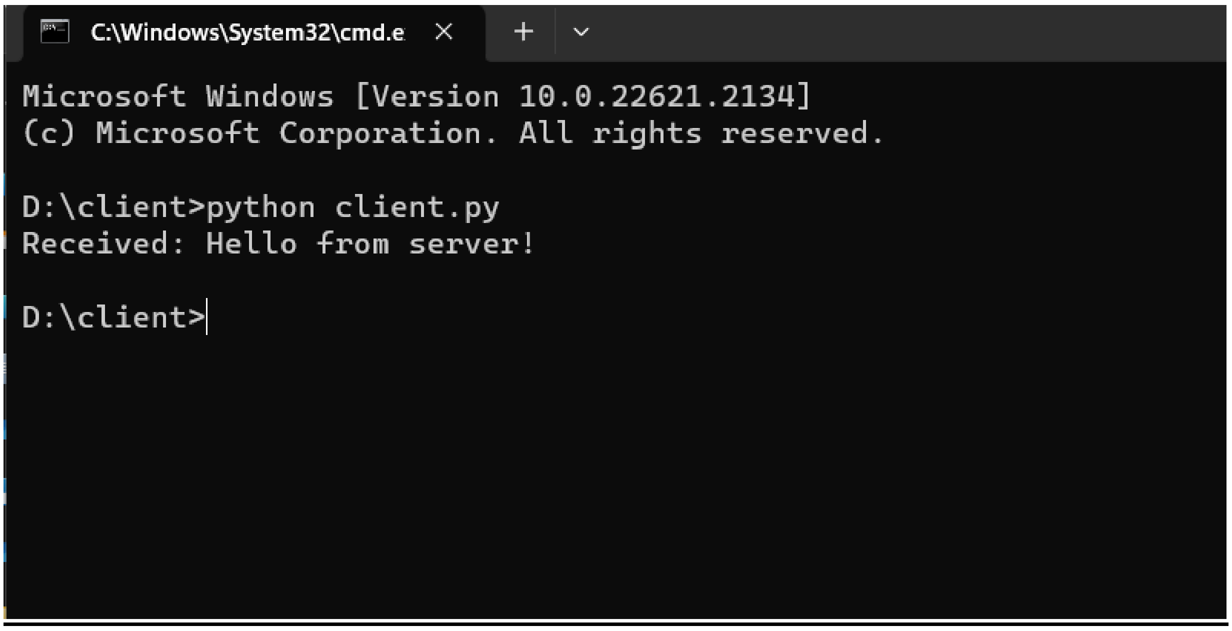
Client.py

```
import socket
import ssl



context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH, cafile='ca.crt')
context.check_hostname = False


context.verify_mode = ssl.CERT_NONE
context.set_ciphers('DEFAULT@SECLEVEL=1')
context.set_alpn_protocols(['http/1.1'])

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ssl_client = context.wrap_socket(client, server_hostname='127.0.0.1')
ssl_client.connect(('127.0.0.1', 12345))

ssl_client.send("Hello from client!".encode())
data = ssl_client.recv(1024)

print(f"Received: {data.decode()}")

ssl_client.close()
```

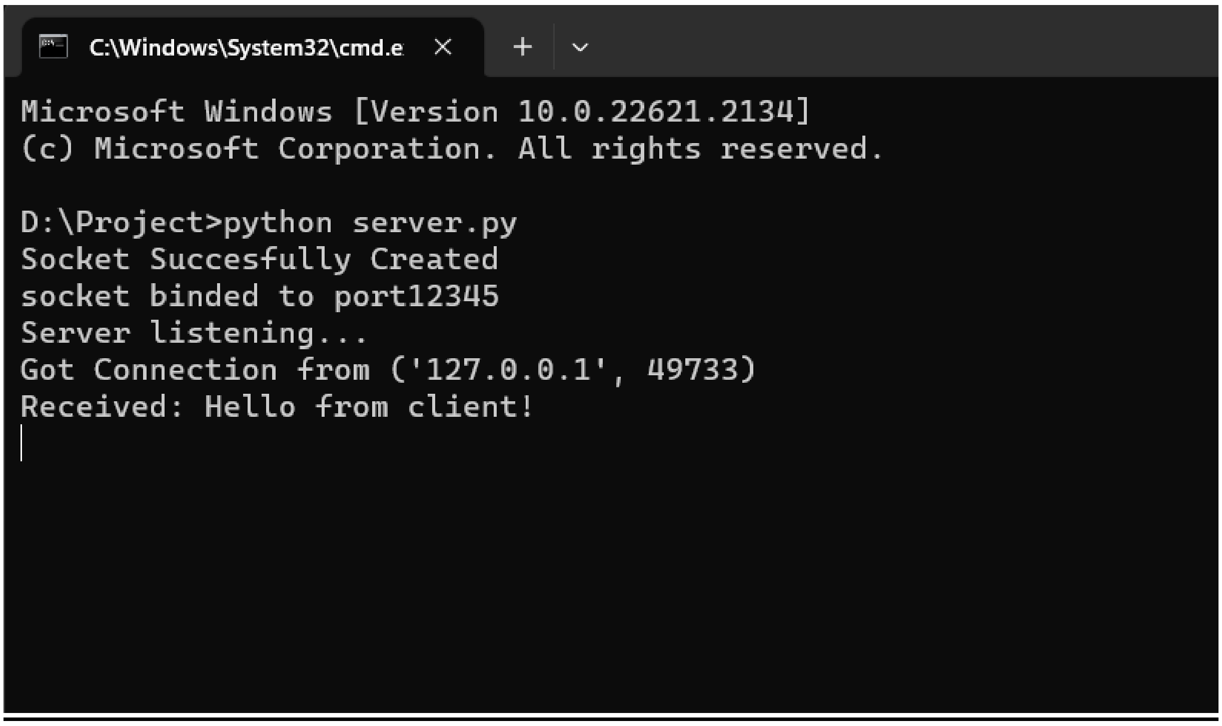
OUTPUT



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server.py
Socket Succesfully Created
socket binded to port12345
Server listening...
```



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\client>python client.py
Received: Hello from server!

D:\client>
```



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server.py
Socket Succesfully Created
socket binded to port12345
Server listening...
Got Connection from ('127.0.0.1', 49733)
Received: Hello from client!
```

- **<u>Code for Server Client Connections to share Text File from Other/External Client’s Device.</u>**

<u>Server.py</u>

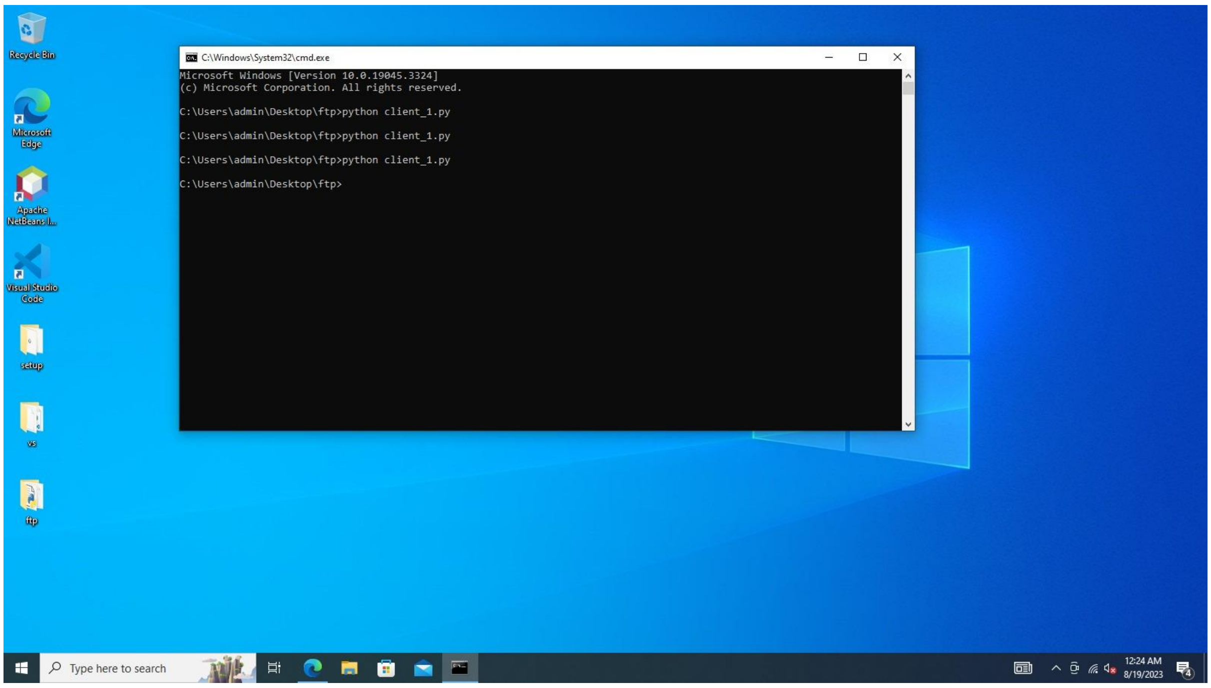
```
import socket
import ssl


def receive_file(ssl_socket, filename):
    with open(filename, 'wb') as file:


        while True:


            data = ssl_socket.recv(4096)
            if not data:
                break
            file.write(data)
```

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('192.168.80.160', 12345))

server.listen(5)

context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain(certfile='server.crt', keyfile='server.key')

print("Server listening...")


while True:

    client, addr = server.accept()

    print("Got Connection from", addr)

    ssl_client = context.wrap_socket(client, server_side=True)

    filename = ssl_client.recv(1024).decode()

    print(f"Receiving file: {filename}")

    receive_file(ssl_client, filename)

    print("File Received")

    ssl_client.close()
```

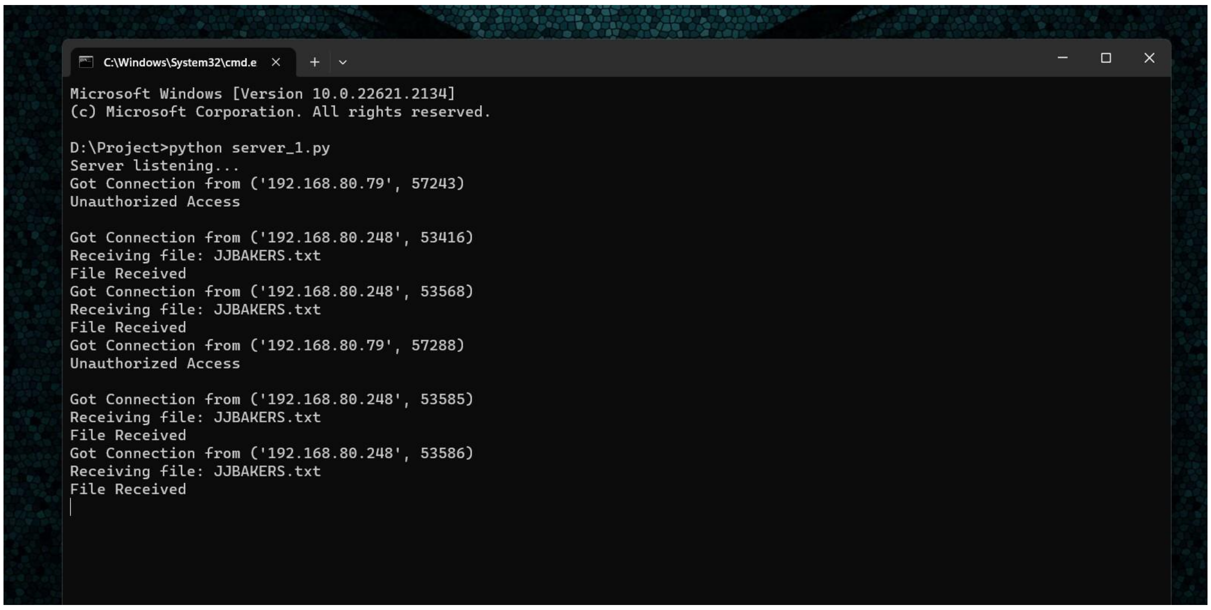
Client.py

```
import socket
import ssl

def send_file(ssl_socket, filename):

    with open(filename, 'rb') as file:
        for data in file:
            ssl_socket.send(data)

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH, cafile='ca.crt')
context.check_hostname = False

context.verify_mode = ssl.CERT_NONE

ssl_client = context.wrap_socket(client, server_hostname='192.168.80.160')
ssl_client.connect(('192.168.80.160', 12345))

filename = 'data.txt'

ssl_client.send(filename.encode())

send_file(ssl_client, filename)

ssl_client.close()
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\admin\Desktop\ftp>python client_1.py

C:\Users\admin\Desktop\ftp>python client_1.py

C:\Users\admin\Desktop\ftp>python client_1.py

C:\Users\admin\Desktop\ftp>
```



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server_1.py
Server listening...
Got Connection from ('192.168.80.79', 57243)
Unauthorized Access

Got Connection from ('192.168.80.248', 53416)
Receiving file: JJBAKERS.txt
File Received
Got Connection from ('192.168.80.248', 53568)
Receiving file: JJBAKERS.txt
File Received
Got Connection from ('192.168.80.79', 57288)
Unauthorized Access

Got Connection from ('192.168.80.248', 53585)
Receiving file: JJBAKERS.txt
File Received
Got Connection from ('192.168.80.248', 53586)
Receiving file: JJBAKERS.txt
File Received
```

- **Code for Server Client Connections to share Mp3 File from Local Client’s Device.**

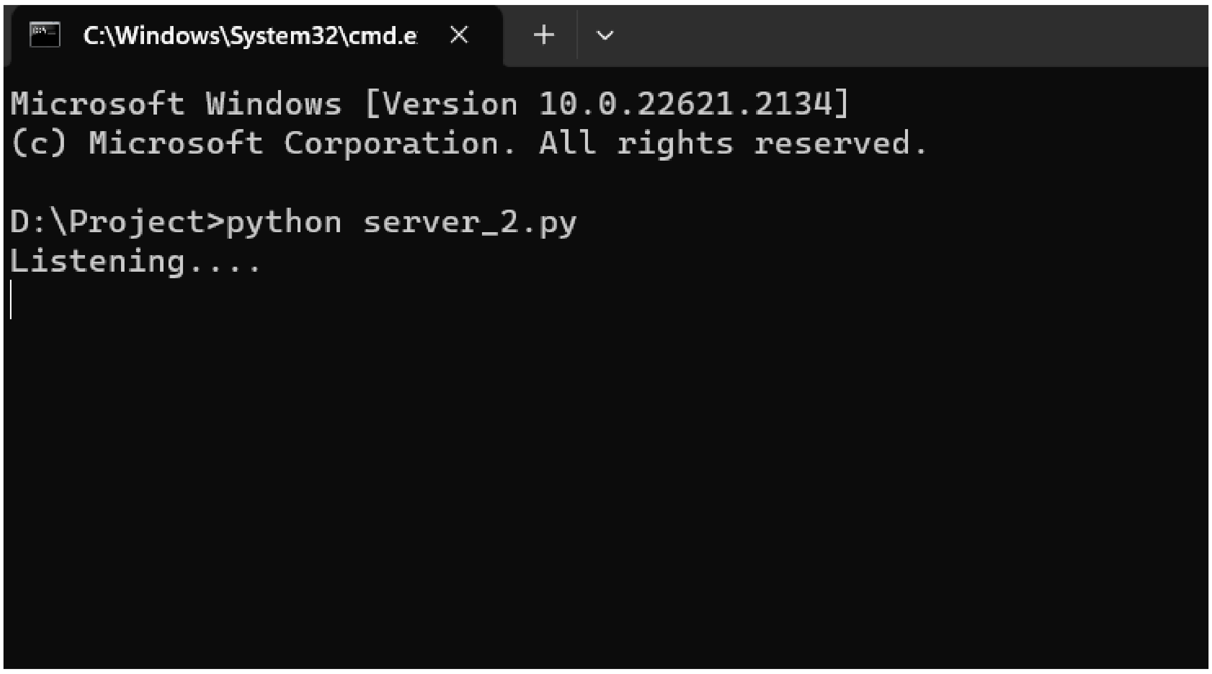
Server.py

```
import socket
import ssl

context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain(certfile='server.crt', keyfile='server.key')
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('127.0.0.1', 12345))
server.listen(5)

while True:
    print("Listening....")
    client, addr = server.accept()
    ssl_client = context.wrap_socket(client, server_side=True)
    mp3_data = b''
    while True:
        chunk = ssl_client.recv(1024)
        if not chunk:
            break
        mp3_data += chunk

    with open('1230.mp3', 'wb') as received_file:
        received_file.write(mp3_data)
    print("File received")

    ssl_client.close()
```

Client.py

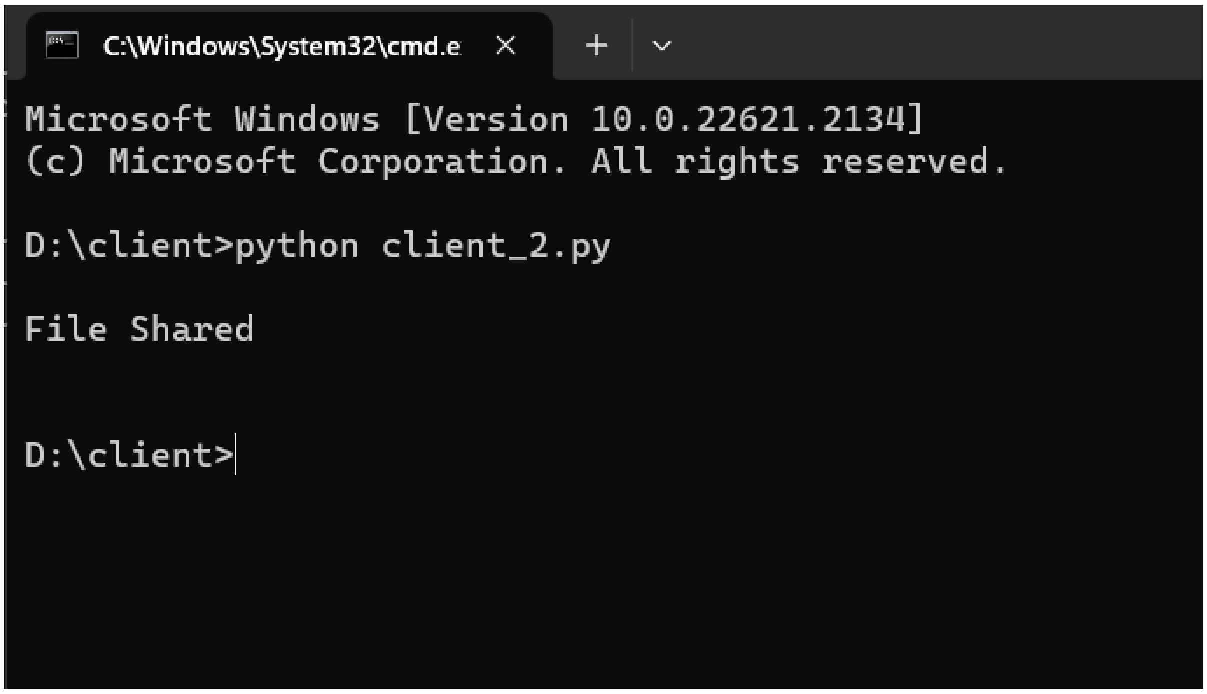
```
import socket
import ssl

context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH)

context.check_hostname = False

context.verify_mode = ssl.CERT_NONE

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

ssl_client = context.wrap_socket(client, server_hostname='127.0.0.1')

ssl_client.connect(('127.0.0.1', 12345))

with open('1230.mp3', 'rb') as mp3_file:

    mp3_data = mp3_file.read()

ssl_client.sendall(mp3_data)

print("\nFile Shared\n")

ssl_client.close()
```

Output.py



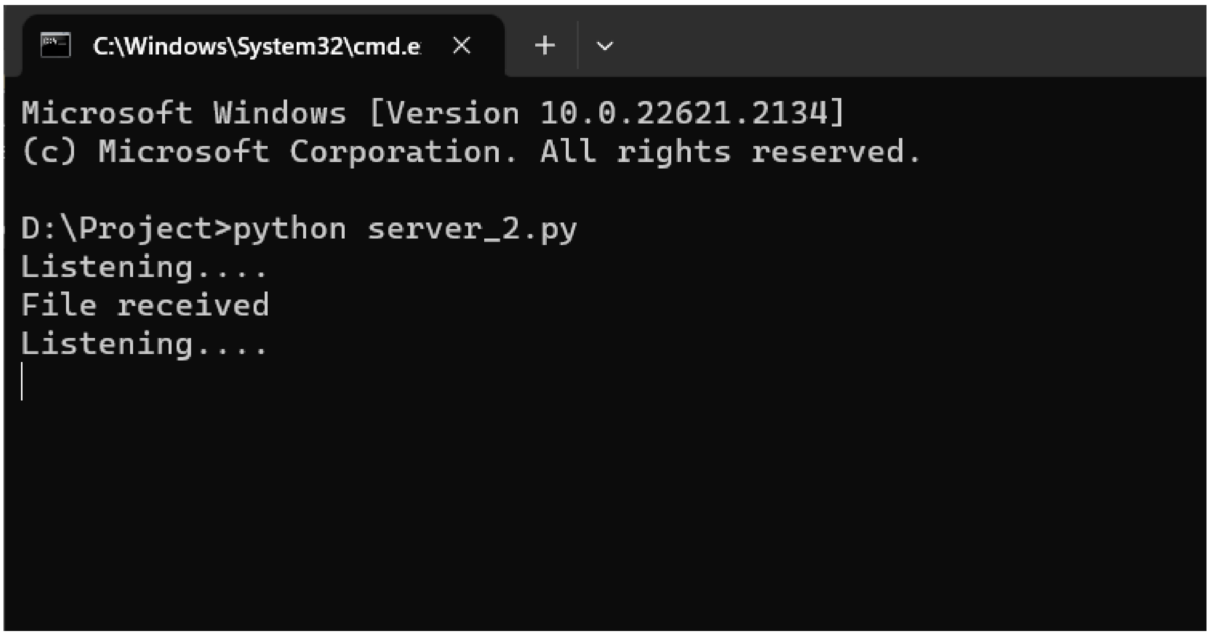
C:\Windows\System32\cmd.e

```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server_2.py
Listening....
```

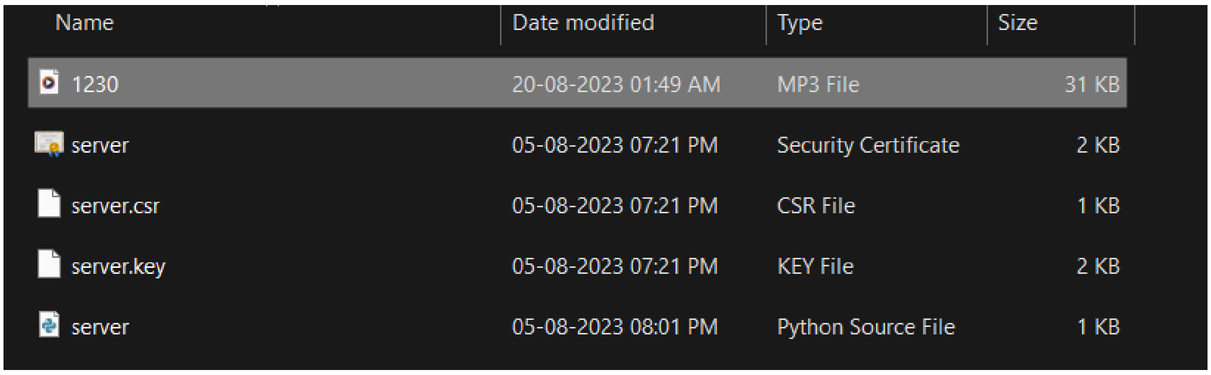


C:\Windows\System32\cmd.e

```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\client>python client_2.py

File Shared


D:\client>
```



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server_2.py
Listening....
File received
Listening....
```



Name	Date modified	Type	Size
1230	20-08-2023 01:49 AM	MP3 File	31 KB
server	05-08-2023 07:21 PM	Security Certificate	2 KB
server.csr	05-08-2023 07:21 PM	CSR File	1 KB
server.key	05-08-2023 07:21 PM	KEY File	2 KB
server	05-08-2023 08:01 PM	Python Source File	1 KB

- **Code for Server Client Connections to Block Unauthorized Client's Device.**

Server.py

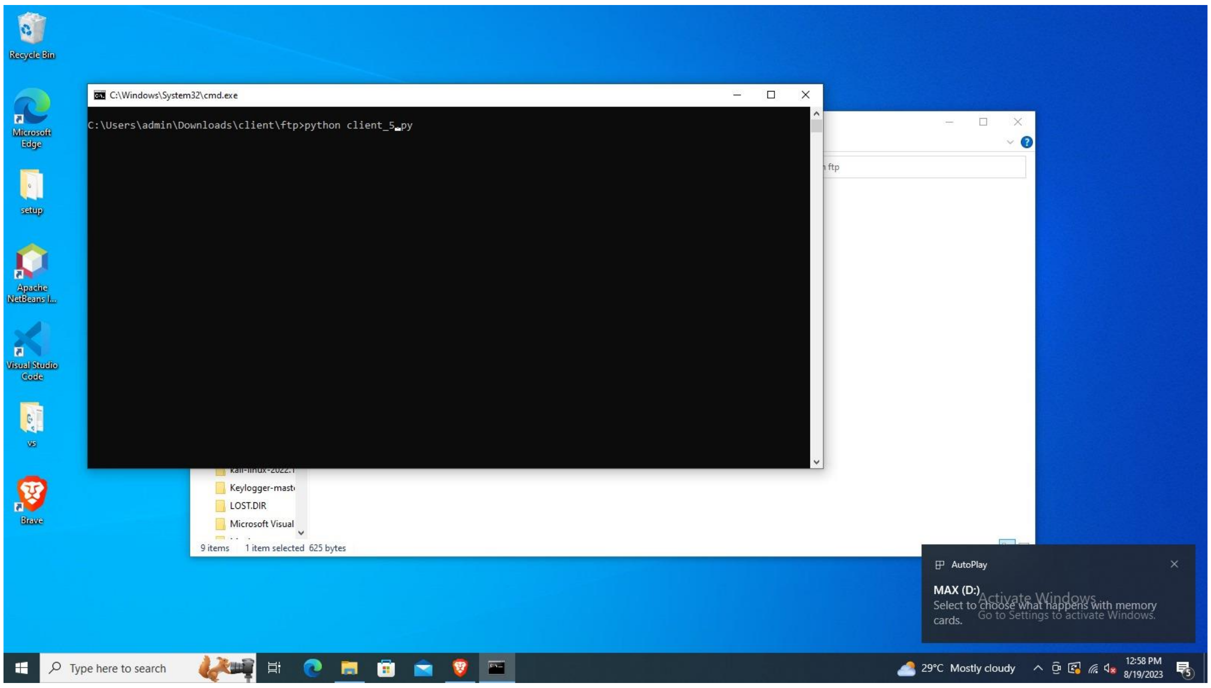
```
import socket
import ssl

def receive_file(ssl_socket, filename):
    with open(filename, 'wb') as file:
        while True:
            data = ssl_socket.recv(4096)
            if not data:
                break
            file.write(data)

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('192.168.80.160', 12345))
server.listen(5)

context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
context.load_cert_chain(certfile='server.crt', keyfile='server.key')

print("Server listening...")
add,port=("192.168.80.79",12345)

while True:
    try:

        client, addr = server.accept()
        print("Got Connection from", addr)
        if(addr[0]==add):
            print("Unauthorized Access\n")
            continue
        else:
            ssl_client = context.wrap_socket(client, server_side=True)

            filename = ssl_client.recv(1024).decode()
            print(f"Receiving file: {filename}")
            receive_file(ssl_client, filename)
            print("File Received")
    except:
        print("Error")
    ssl_client.close()
```

Client.py

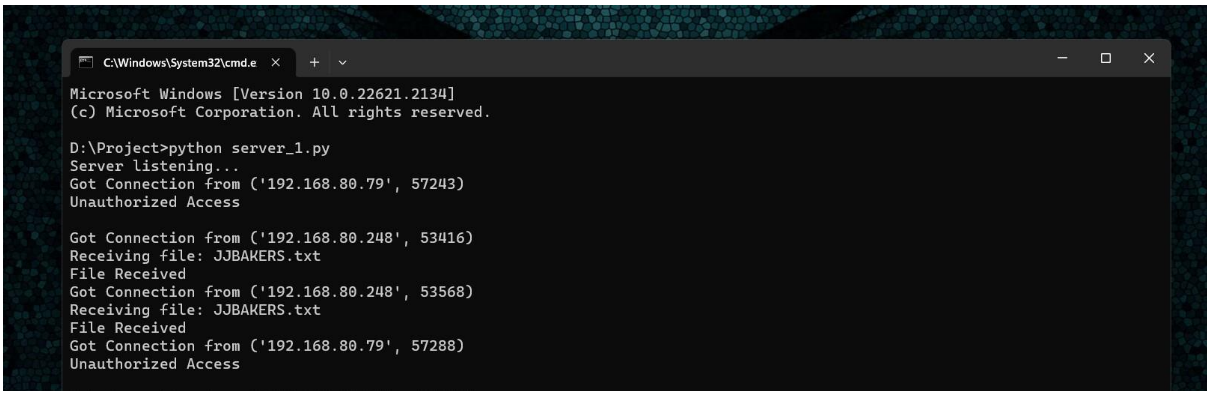
```
import socket
import ssl

def send_file(ssl_socket, filename):
    with open(filename, 'rb') as file:
        for data in file:
            ssl_socket.send(data)
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
context = ssl.create_default_context(ssl.Purpose.SERVER_AUTH, cafile='ca.crt')
context.check_hostname = False
context.verify_mode = ssl.CERT_NONE

ssl_client = context.wrap_socket(client, server_hostname='127.0.0.1')
```

```
ssl_client.connect(('127.0.0.1', 12345))

filename = 'data.txt'
ssl_client.send(filename.encode())
send_file(ssl_client, filename)
ssl_client.close()
```

Output:



```
C:\Users\admin\Downloads\client\ftp>python client_5.py
```



```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

D:\Project>python server_1.py
Server listening...
Got Connection from ('192.168.80.79', 57243)
Unauthorized Access

Got Connection from ('192.168.80.248', 53416)
Receiving file: JJBAKERS.txt
File Received
Got Connection from ('192.168.80.248', 53568)
Receiving file: JJBAKERS.txt
File Received
Got Connection from ('192.168.80.79', 57288)
Unauthorized Access
```