Weekly Assessment – 10

Thanvi Katakam

(2023006366)

AIM: To implement a file transfer system using socket programming in Python. The system includes:

1. **Sequential File Transfer Server** – Handles one client at a time.
2. **Concurrent File Transfer Server** – Handles multiple clients simultaneously using threads.

DESCRIPTION: Socket programming allows processes to communicate over a network. It enables file transfer between a client and a server using **TCP sockets**.

- **Sequential Server:** The server serves one client at a time. A client requests a file, the server sends it, and then waits for the next client.
- **Concurrent Server:** The server uses **threads** to handle multiple clients simultaneously, allowing concurrent file transfers.

PROCEDURE:

Exercise 1: Sequential File Transfer Server

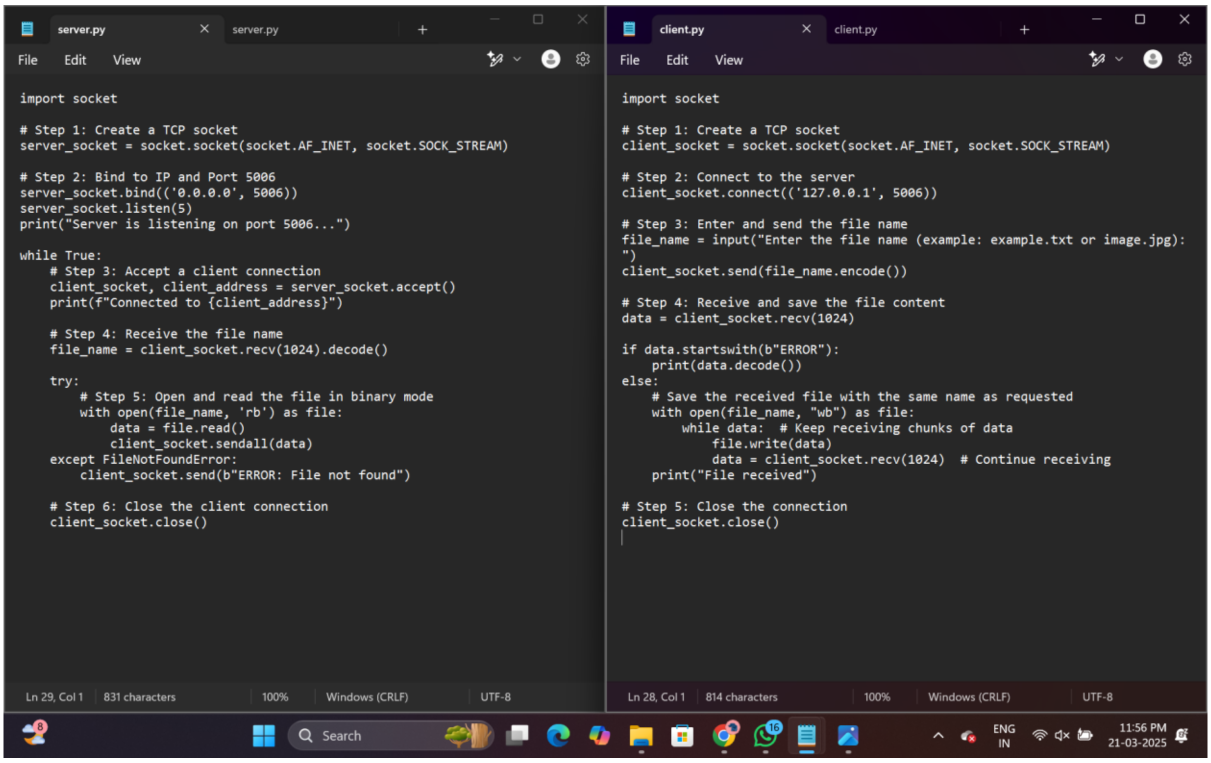
1. Folder Setup

- Create a folder structure:
- socket_file_transfer/
- — sequential_server/
    - — server.py
    - — client.py

      — example.txt

      — image.jpg (Optional)
- Add a sample text file (example.txt).

Server code:(server.py)   Client code:(Client.py)

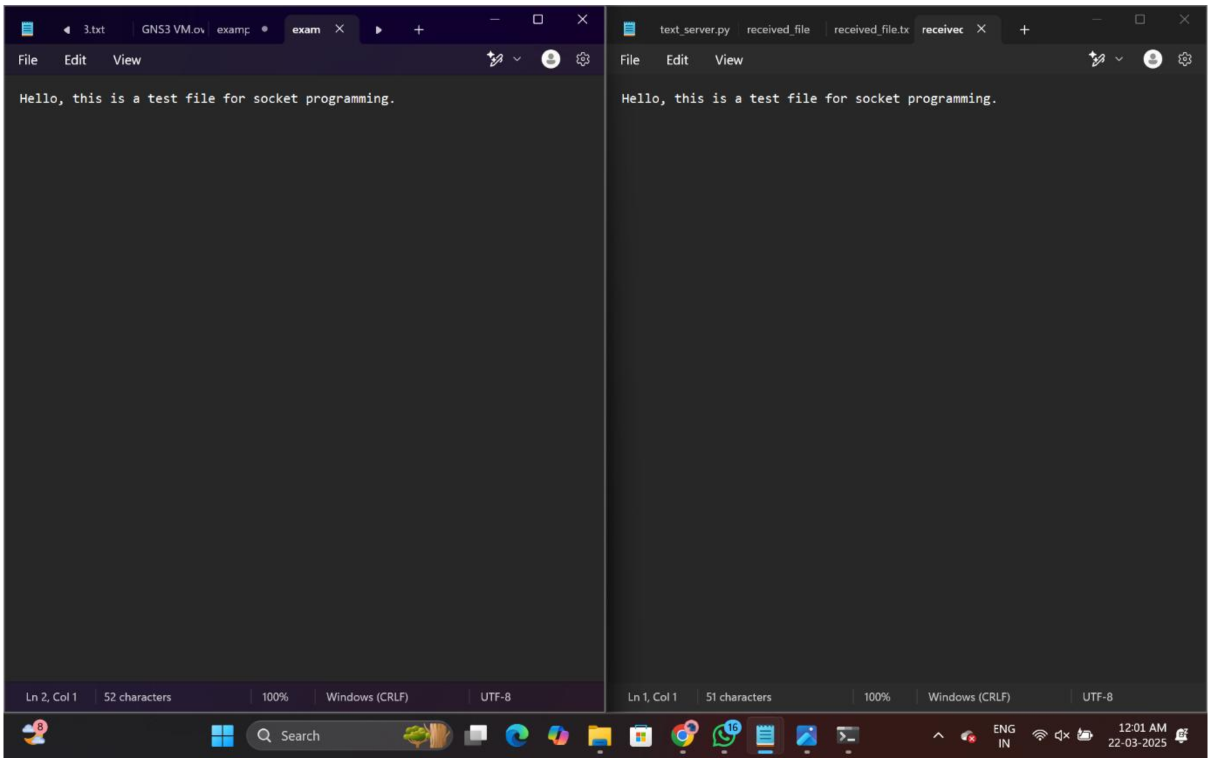


```python
import socket

# Step 1: Create a TCP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Step 2: Bind to IP and Port 5006
server_socket.bind(('0.0.0.0', 5006))
server_socket.listen(5)
print("Server is listening on port 5006...")

while True:
    # Step 3: Accept a client connection
    client_socket, client_address = server_socket.accept()
    print(f"Connected to {client_address}")

    # Step 4: Receive the file name
    file_name = client_socket.recv(1024).decode()

    try:
        # Step 5: Open and read the file in binary mode
        with open(file_name, 'rb') as file:
            data = file.read()
            client_socket.sendall(data)
    except FileNotFoundError:
        client_socket.send(b"ERROR: File not found")

    # Step 6: Close the client connection
    client_socket.close()
```

```python
import socket

# Step 1: Create a TCP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Step 2: Connect to the server
client_socket.connect(('127.0.0.1', 5006))

# Step 3: Enter and send the file name
file_name = input("Enter the file name (example: example.txt or image.jpg):
")
client_socket.send(file_name.encode())

# Step 4: Receive and save the file content
data = client_socket.recv(1024)

if data.startswith(b"ERROR"):
    print(data.decode())
else:
    # Save the received file with the same name as requested
    with open(file_name, "wb") as file:
        while data:  # Keep receiving chunks of data
            file.write(data)
            data = client_socket.recv(1024)  # Continue receiving
    print("File received")

# Step 5: Close the connection
client_socket.close()
```

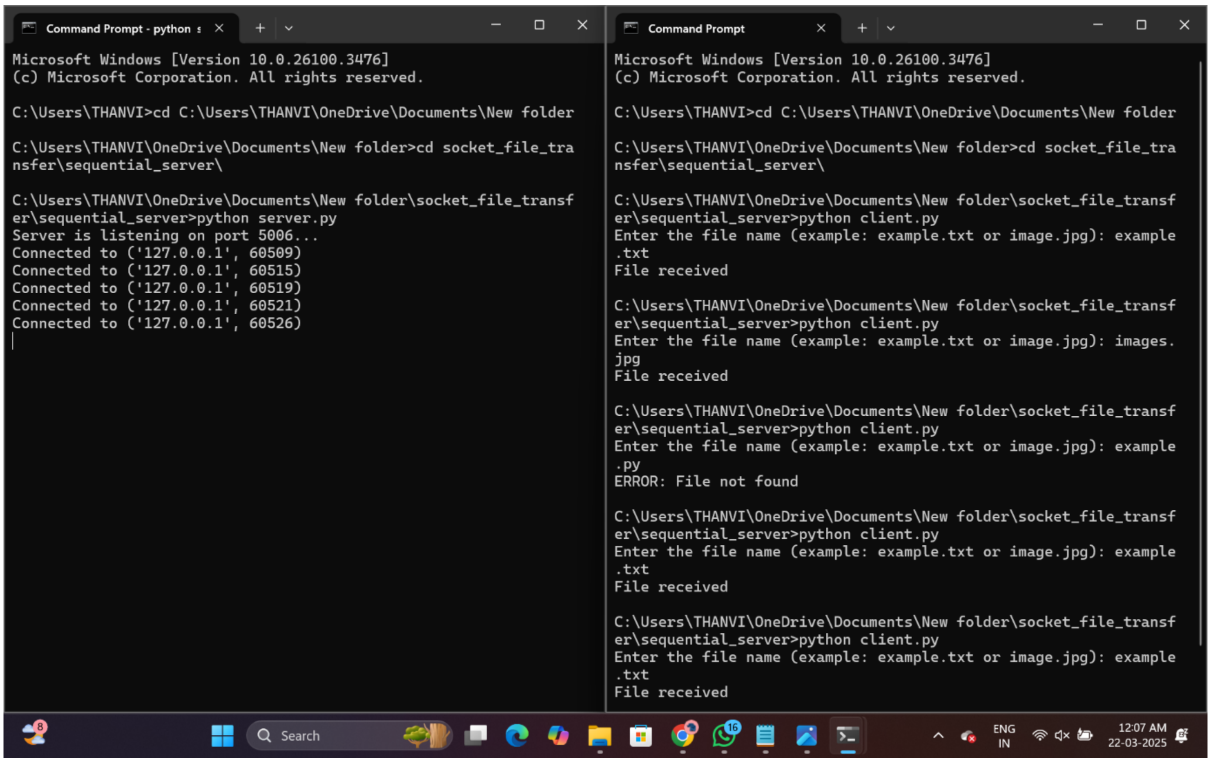
- Run multiple clients that request the server for binary files.
- The server serves each client one after the other before terminating the connection.
- Use a try-except clause to catch an exception for a file not found on the server.

example.txt:


```
Hello, this is a test file for socket programming.
```

```
Hello, this is a test file for socket programming.
```

Server output:(server.py)   Client output:(client.py)


```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\THANVI>cd C:\Users\THANVI\OneDrive\Documents\New folder

C:\Users\THANVI\OneDrive\Documents\New folder>cd socket_file_tra
nsfer\sequential_server\

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python server.py
Server is listening on port 5006...
Connected to ('127.0.0.1', 60509)
Connected to ('127.0.0.1', 60515)
Connected to ('127.0.0.1', 60519)
Connected to ('127.0.0.1', 60521)
Connected to ('127.0.0.1', 60526)
```

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\THANVI>cd C:\Users\THANVI\OneDrive\Documents\New folder

C:\Users\THANVI\OneDrive\Documents\New folder>cd socket_file_tra
nsfer\sequential_server\

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.txt
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python client.py
Enter the file name (example: example.txt or image.jpg): images.
jpg
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.py
ERROR: File not found

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.txt
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\sequential_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.txt
File received
```

Exercise 2: Concurrent File Transfer Server

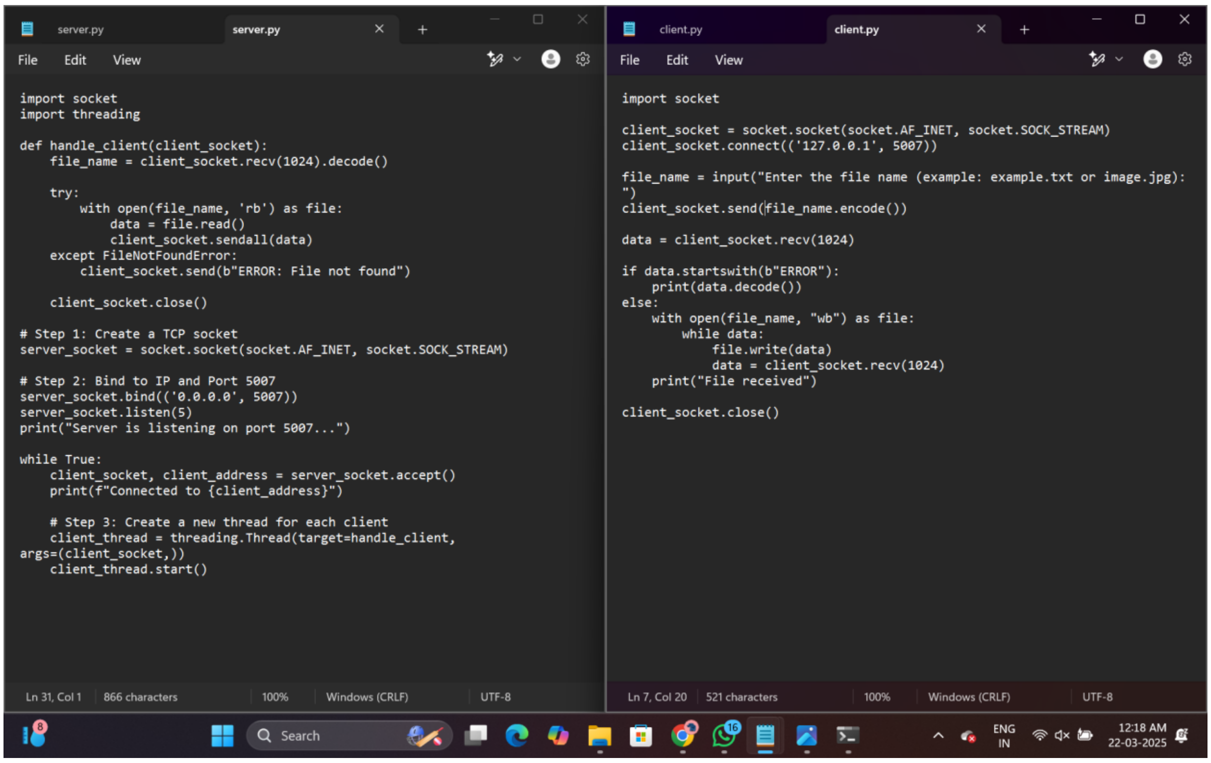
1. Folder Setup

- Create a folder structure:
- socket_file_transfer/
- –→ concurrent_server/
    - — server.py

      — client.py

      — example.txt

Server code(server.py)   Client code(client.py)



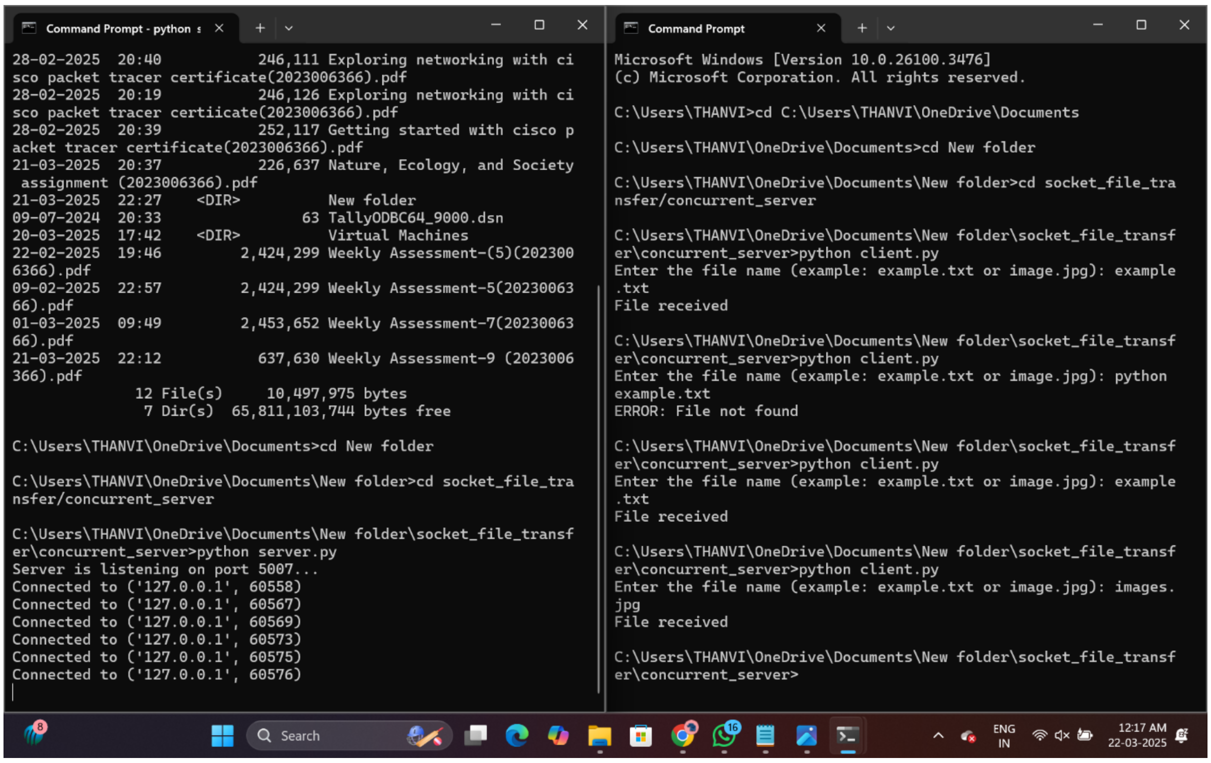
```python
import socket
import threading

def handle_client(client_socket):
    file_name = client_socket.recv(1024).decode()

    try:
        with open(file_name, 'rb') as file:
            data = file.read()
            client_socket.sendall(data)
    except FileNotFoundError:
        client_socket.send(b"ERROR: File not found")

    client_socket.close()

# Step 1: Create a TCP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Step 2: Bind to IP and Port 5007
server_socket.bind(('0.0.0.0', 5007))
server_socket.listen(5)
print("Server is listening on port 5007...")

while True:
    client_socket, client_address = server_socket.accept()
    print(f"Connected to {client_address}")

    # Step 3: Create a new thread for each client
    client_thread = threading.Thread(target=handle_client,
args=(client_socket,))
    client_thread.start()
```

```python
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('127.0.0.1', 5007))

file_name = input("Enter the file name (example: example.txt or image.jpg):
")
client_socket.send(file_name.encode())

data = client_socket.recv(1024)

if data.startswith(b"ERROR"):
    print(data.decode())
else:
    with open(file_name, "wb") as file:
        while data:
            file.write(data)
            data = client_socket.recv(1024)
    print("File received")

client_socket.close()
```

server.py
File Edit View
Ln 31, Col 1 | 866 characters | 100% | Windows (CRLF) | UTF-8
client.py
File Edit View
Ln 7, Col 20 | 521 characters | 100% | Windows (CRLF) | UTF-8
Search
ENG IN
12:18 AM
22-03-2025

- Run the server: python server.py
- The server should service up to **5 concurrent clients**.
- Modify the program to support **any number of clients concurrently**.
- Discuss **what limits the number of clients** (e.g., system resources, network bandwidth, and thread management).
- Run multiple clients at the same time.
- Each client receives files concurrently.

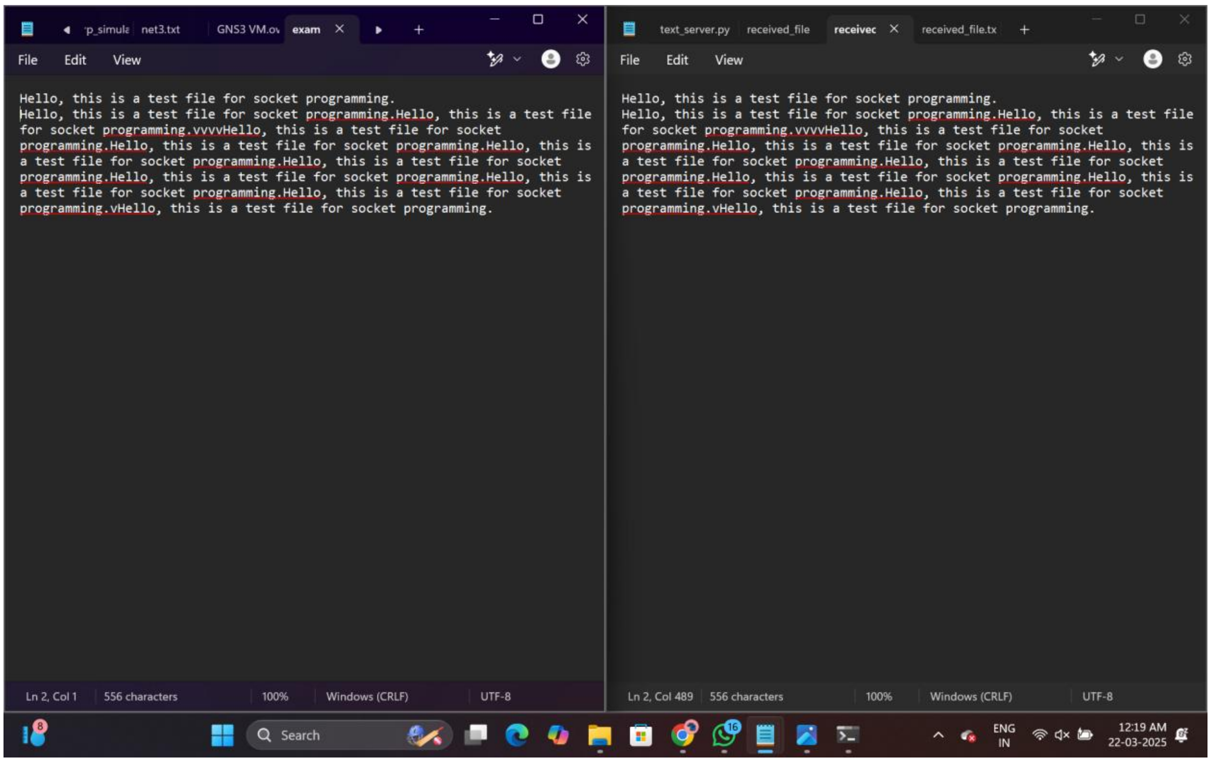
Server Output(server.py)   Client Output(client.py)



```
28-02-2025  20:40          246,111 Exploring networking with ci
sco packet tracer certificate(2023006366).pdf
28-02-2025  20:19          246,126 Exploring networking with ci
sco packet tracer certiicate(2023006366).pdf
28-02-2025  20:39          252,117 Getting started with cisco p
acket tracer certificate(2023006366).pdf
21-03-2025  20:37          226,637 Nature, Ecology, and Society
 assignment (2023006366).pdf
21-03-2025  22:27    <DIR>          New folder
09-07-2024  20:33               63 TallyODBC64_9000.dsn
20-03-2025  17:42    <DIR>          Virtual Machines
22-02-2025  19:46        2,424,299 Weekly Assessment-(5)(202300
6366).pdf
09-02-2025  22:57        2,424,299 Weekly Assessment-5(20230063
66).pdf
01-03-2025  09:49        2,453,652 Weekly Assessment-7(20230063
66).pdf
21-03-2025  22:12          637,630 Weekly Assessment-9 (2023006
366).pdf
              12 File(s)     10,497,975 bytes
               7 Dir(s)  65,811,103,744 bytes free

C:\Users\THANVI\OneDrive\Documents>cd New folder

C:\Users\THANVI\OneDrive\Documents\New folder>cd socket_file_tra
nsfer/concurrent_server

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>python server.py
Server is listening on port 5007...
Connected to ('127.0.0.1', 60558)
Connected to ('127.0.0.1', 60567)
Connected to ('127.0.0.1', 60569)
Connected to ('127.0.0.1', 60573)
Connected to ('127.0.0.1', 60575)
Connected to ('127.0.0.1', 60576)
```

```
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Users\THANVI>cd C:\Users\THANVI\OneDrive\Documents

C:\Users\THANVI\OneDrive\Documents>cd New folder

C:\Users\THANVI\OneDrive\Documents\New folder>cd socket_file_tra
nsfer/concurrent_server

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.txt
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>python client.py
Enter the file name (example: example.txt or image.jpg): python
example.txt
ERROR: File not found

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>python client.py
Enter the file name (example: example.txt or image.jpg): example
.txt
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>python client.py
Enter the file name (example: example.txt or image.jpg): images.
jpg
File received

C:\Users\THANVI\OneDrive\Documents\New folder\socket_file_transf
er\concurrent_server>
```

example.txt: received_text.txt:



```
Hello, this is a test file for socket programming.
Hello, this is a test file for socket programming.Hello, this is a test file
for socket programming.vvvvHello, this is a test file for socket
programming.Hello, this is a test file for socket programming.Hello, this is
a test file for socket programming.Hello, this is a test file for socket
programming.Hello, this is a test file for socket programming.Hello, this is
a test file for socket programming.Hello, this is a test file for socket
programming.vHello, this is a test file for socket programming.
```

```
Hello, this is a test file for socket programming.
Hello, this is a test file for socket programming.Hello, this is a test file
for socket programming.vvvvHello, this is a test file for socket
programming.Hello, this is a test file for socket programming.Hello, this is
a test file for socket programming.Hello, this is a test file for socket
programming.Hello, this is a test file for socket programming.Hello, this is
a test file for socket programming.Hello, this is a test file for socket
programming.vHello, this is a test file for socket programming.
```

CONCLUSION:
Sequential Server: Clients are served one by one.
Concurrent Server: Multiple clients can request files at the same time.
System performance limits the number of concurrent clients.
Socket programming enables efficient file transfers over a network.