

# 22CS501

## COMPUTER NETWORKS

### MINI PROJECT

**Name:** Evangline R

**Register Number:** 111722102033

**Branch:** 3<sup>rd</sup> BE CSE

**Section:** 'A'

### Simple LAN Chat

A Local Area Network (LAN) chat is a real-time communication tool that allows users on the same network to exchange messages instantly. It typically works by connecting users through either

- ✓ client-server or
- ✓ peer-to-peer (P2P) architecture.

### Key Components of LAN Chat:

#### 1. Client-Server Model:

- ✚ **Server:** Manages connections, relays messages between users, and ensures the flow of communication.
- ✚ **Client:** The application used by each user to send and receive messages.

#### 2. Peer-to-Peer (P2P) Model:

In contrast to the client-server setup, this model enables clients to communicate directly without the need for a central server.







### Communication Protocols:

- ✓ **Sockets:** LAN chats rely on sockets for network communication.
- ✓ **TCP (Transmission Control Protocol):** Provides reliable message delivery, ensuring data packets are transmitted and acknowledged.
- ✓ **UDP (User Datagram Protocol):** Faster but less reliable, often used when speed is prioritized over guaranteed delivery (e.g., in gaming).

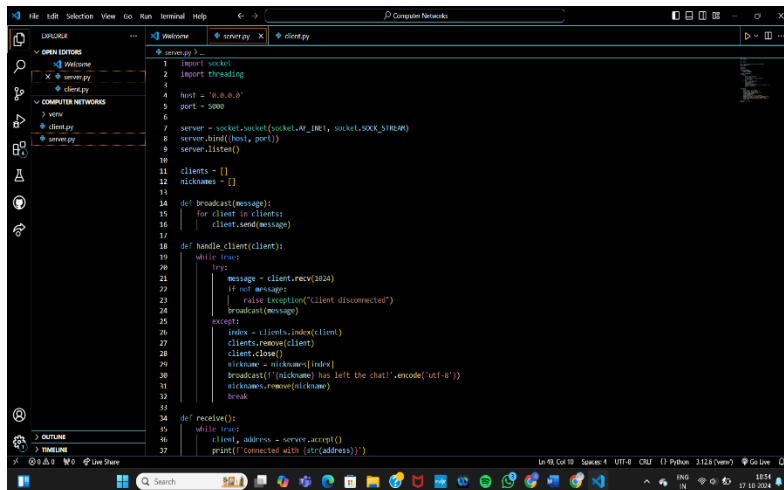
## Steps in LAN Chat Communication:

- 1. Server Initialization:** The server opens a socket on a specific IP address and port, waiting for clients to connect.
- 2. Client Connection:** Clients connect to the server by specifying the correct IP and port.
- 3. Message Exchange:** After a successful connection, clients can send messages, which the server broadcasts to all connected clients.
- 4. Disconnection Handling:** When a client leaves, the server removes them from the active chat session and informs other users.

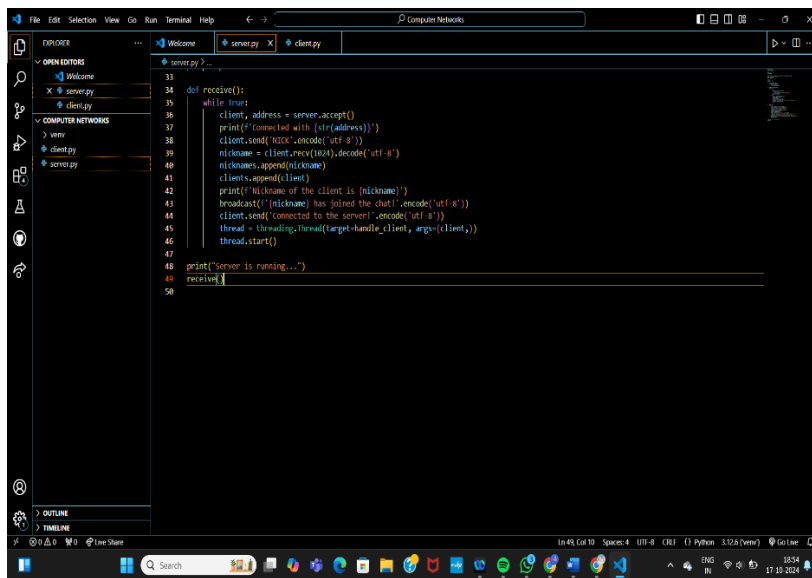
## Important Features of LAN Chat:

-  **Broadcasting:** Messages sent by one user are distributed to all clients via the server.
-  **Multithreading:** The server often uses threads to manage multiple client connections simultaneously.
-  **User Identification:** A unique identifier typically recognises each client, such as a username or IP address.
-  **Private Messaging:** Clients can send messages to a specific user, rather than broadcasting to everyone.
-  **File Sharing:** Some LAN chat applications allow users to send files across the network.
-  **Chat History:** Messages can be stored to provide users with access to previous conversations.

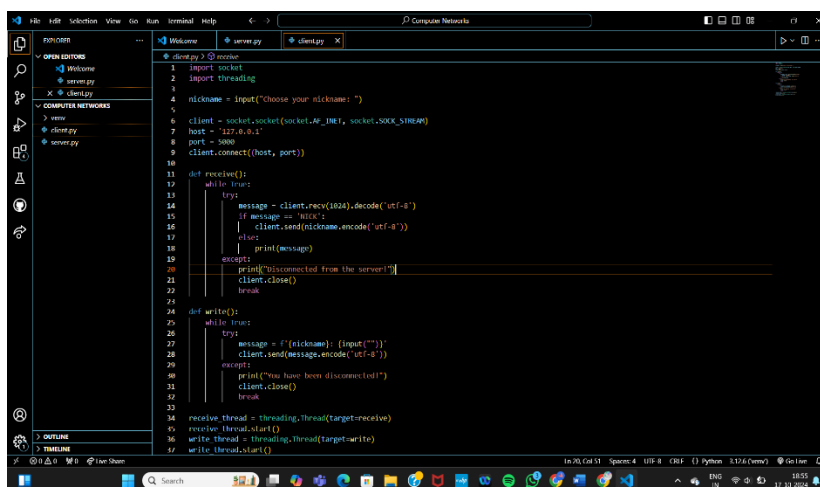
## Example Implementation:



```
1 import socket
2 import threading
3
4 host = '0.0.0.0'
5 port = 5000
6
7 server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 server.bind((host, port))
9 server.listen()
10
11 clients = []
12 nicknames = []
13
14 def broadcast(message):
15     for client in clients:
16         client.send(message)
17
18 def handle_client(client):
19     while True:
20         try:
21             message = client.recv(1024)
22             if not message:
23                 raise Exception("Client disconnected")
24             broadcast(message)
25         except:
26             index = clients.index(client)
27             clients.remove(client)
28             client.close()
29             nickname = nicknames[index]
30             broadcast(f'{nickname} has left the chat!'.encode('utf-8'))
31             nicknames.remove(nickname)
32             break
33
34 def receive():
35     while True:
36         client, address = server.accept()
37         print(f'Connected with {str(address)}')
38         client.send('NICK'.encode('utf-8'))
39         nickname = client.recv(1024).decode('utf-8')
40         nicknames.append(nickname)
41         clients.append(client)
42         print(f'Nickname of the client is {nickname}')
43         broadcast(f'{nickname} has joined the chat!'.encode('utf-8'))
44         client.send(f'Connected to the server!'.encode('utf-8'))
45         thread = threading.Thread(target=handle_client, args=(client,))
46         thread.start()
47
48 print('Server is running...')
49 receive()
50
```

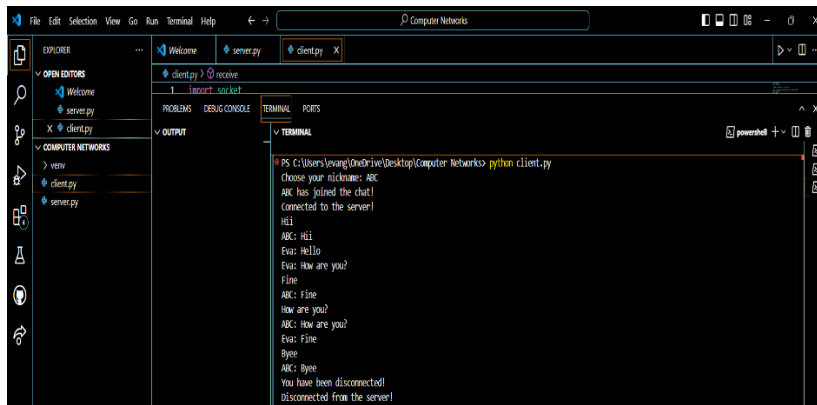


```
33
34
35 def receive():
36     while True:
37         client, address = server.accept()
38         print(f'Connected with {str(address)}')
39         client.send('NICK'.encode('utf-8'))
40         nickname = client.recv(1024).decode('utf-8')
41         nicknames.append(nickname)
42         clients.append(client)
43         print(f'Nickname of the client is {nickname}')
44         broadcast(f'{nickname} has joined the chat!'.encode('utf-8'))
45         client.send(f'Connected to the server!'.encode('utf-8'))
46         thread = threading.Thread(target=handle_client, args=(client,))
47         thread.start()
48
49 print('Server is running...')
50 receive()
51
```



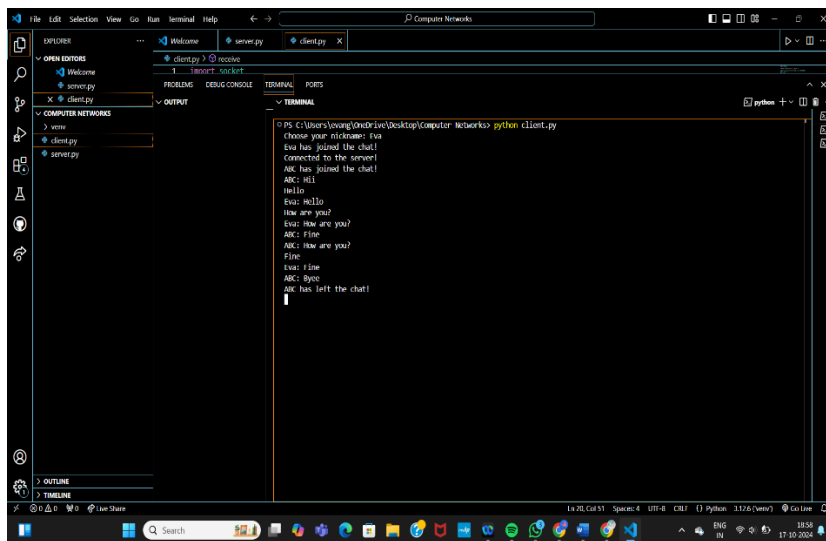
```
1 import socket
2 import threading
3
4 nickname = input("Choose your nickname: ")
5
6 client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7 host = '127.0.0.1'
8 port = 5000
9 client.connect((host, port))
10
11 def receive():
12     while True:
13         try:
14             message = client.recv(1024).decode('utf-8')
15             if message == 'NICK:':
16                 client.send(nickname.encode('utf-8'))
17             else:
18                 print(message)
19         except:
20             print("Disconnected from the server!")
21             client.close()
22             break
23
24 def write():
25     while True:
26         try:
27             message = f'{nickname}: {input("")}'
28             client.send(message.encode('utf-8'))
29         except:
30             print("You have been disconnected!")
31             client.close()
32             break
33
34 receive_thread = threading.Thread(target=receive)
35 receive_thread.start()
36 write_thread = threading.Thread(target=write)
37 write_thread.start()
38
```

## Output:



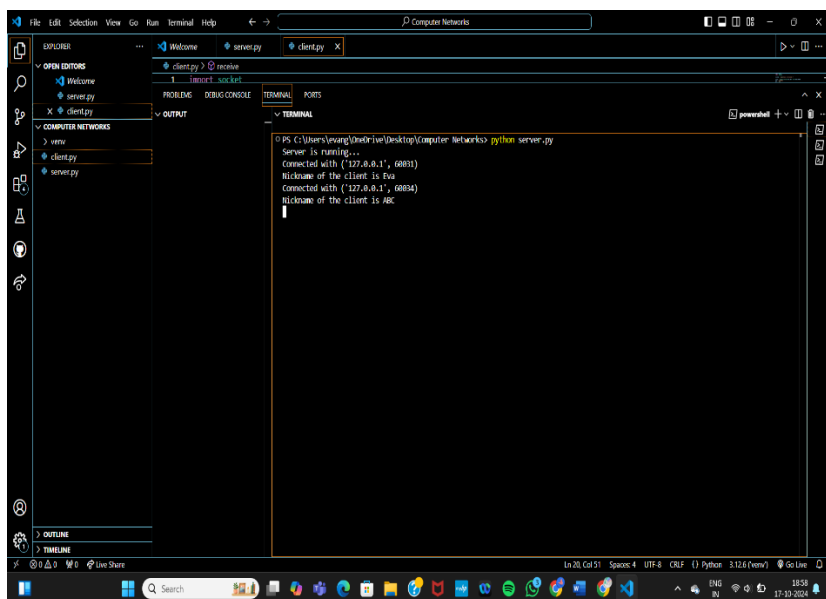
A screenshot of the Visual Studio Code interface. The Explorer pane on the left shows a project named 'Computer Networks' with files 'server.py', 'client.py', and 'venv'. The client.py file is open in the editor, showing a single line of code: `import socket`. The Output pane shows the execution output of client.py, and the Terminal pane shows a PowerShell prompt where the command `python client.py` has been executed. The terminal output shows a chat session where a user named ABC connects to a server, exchanges greetings, and then disconnects.

```
PS C:\Users\evang\OneDrive\Desktop\Computer Networks> python client.py
Choose your nickname: ABC
ABC has joined the chat!
Connected to the server!
Hi!
ABC: Hi!
Eva: Hello
Eva: How are you?
Fine
ABC: Fine
How are you?
ABC: How are you?
Eva: Fine
Bye
ABC: Bye
You have been disconnected!
Disconnected from the server!
```



A screenshot of the Visual Studio Code interface, similar to the first one. The client.py file is open, and the terminal shows the output of `python client.py`. The chat session continues with a user named Eva joining, exchanging greetings with ABC, and then leaving the chat.

```
PS C:\Users\evang\OneDrive\Desktop\Computer Networks> python client.py
Choose your nickname: Eva
Eva has joined the chat!
Connected to the server!
ABC has joined the chat!
ABC: Hi!
Hello
Eva: Hello
How are you?
Eva: How are you?
ABC: Fine
ABC: How are you?
Fine
Eva: Fine
ABC: Bye
ABC has left the chat!
```



A screenshot of the Visual Studio Code interface. The Explorer pane shows the 'Computer Networks' project with files 'server.py', 'client.py', and 'venv'. The server.py file is open in the editor, showing a single line of code: `import socket`. The Output pane shows the execution output of server.py, and the Terminal pane shows a PowerShell prompt where the command `python server.py` has been executed. The terminal output shows the server starting, receiving connections from clients at IP addresses 127.0.0.1, and receiving their nicknames, Eva and ABC.

```
PS C:\Users\evang\OneDrive\Desktop\Computer Networks> python server.py
Server is running...
Connected with ("127.0.0.1", 60031)
Nickname of the client is Eva
Connected with ("127.0.0.1", 60034)
Nickname of the client is ABC
```