22CS501
COMPUTER NETWORKS
MINI PROJECT

Name: Evangline R

Register Number: 111722102033

Branch: 3rd 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.

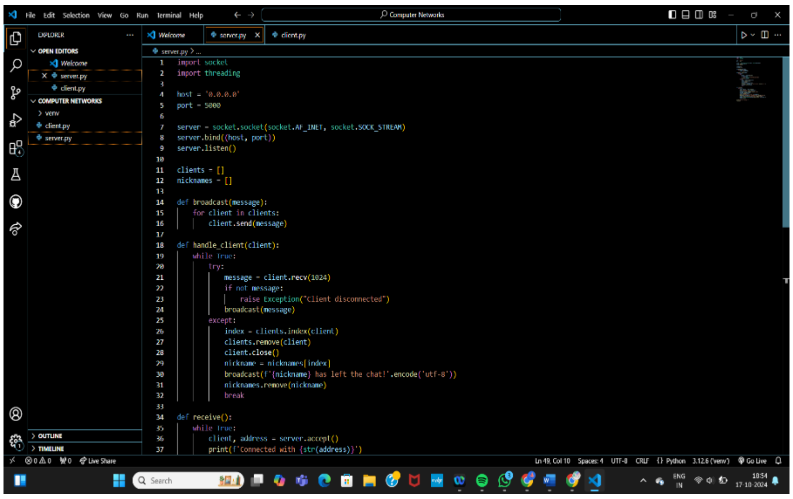
3. Message Exchange: After a successful connection, clients can send messages, which the server broadcasts to all connected clients.

4. Disconnection Handling: When a client leaves, the server removes them from the active chat session and informs other users.

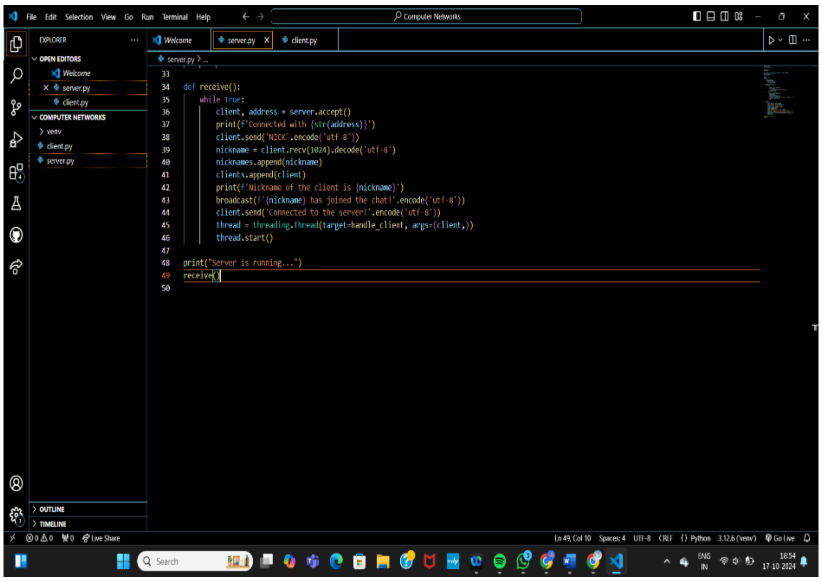
Important Features of LAN Chat:

- **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.

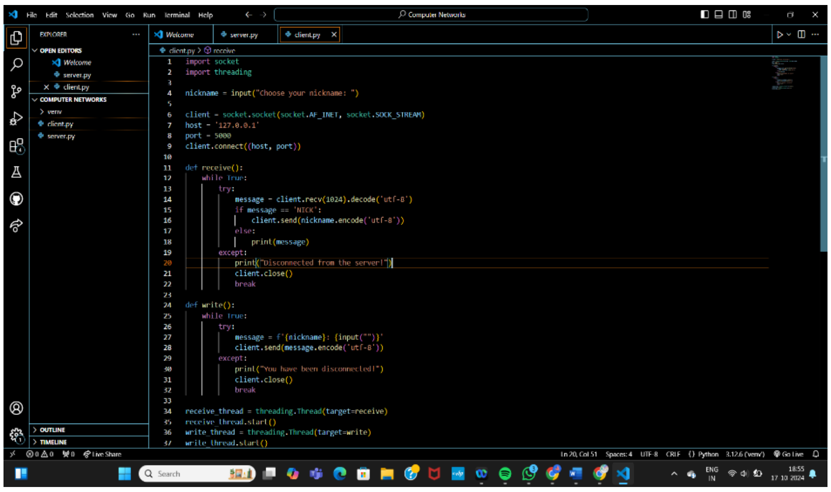
Example Implementation:



```python
import socket
import threading

host = '0.0.0.0'
port = 5000

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind((host, port))
server.listen()

clients = []
nicknames = []

def broadcast(message):
    for client in clients:
        client.send(message)

def handle_client(client):
    while True:
        try:
            message = client.recv(1024)
            if not message:
                raise Exception("Client disconnected")
            broadcast(message)
        except:
            index = clients.index(client)
            clients.remove(client)
            client.close()
            nickname = nicknames[index]
            broadcast(f'{nickname} has left the chat!'.encode('utf-8'))
            nicknames.remove(nickname)
            break

def receive():
    while True:
        client, address = server.accept()
        print(f'Connected with {str(address)}')
```

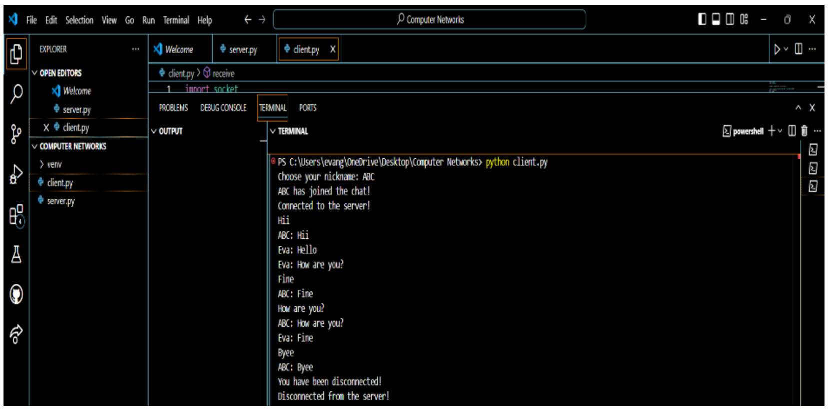


```python

def receive():
    while True:
        client, address = server.accept()
        print(f'Connected with {str(address)}')
        client.send('NICK'.encode('utf-8'))
        nickname = client.recv(1024).decode('utf-8')
        nicknames.append(nickname)
        clients.append(client)
        print(f'Nickname of the client is {nickname}')
        broadcast(f'{nickname} has joined the chat!'.encode('utf-8'))
        client.send('Connected to the server!'.encode('utf-8'))
        thread = threading.Thread(target=handle_client, args=(client,))
        thread.start()

print("Server is running...")
receive()

```

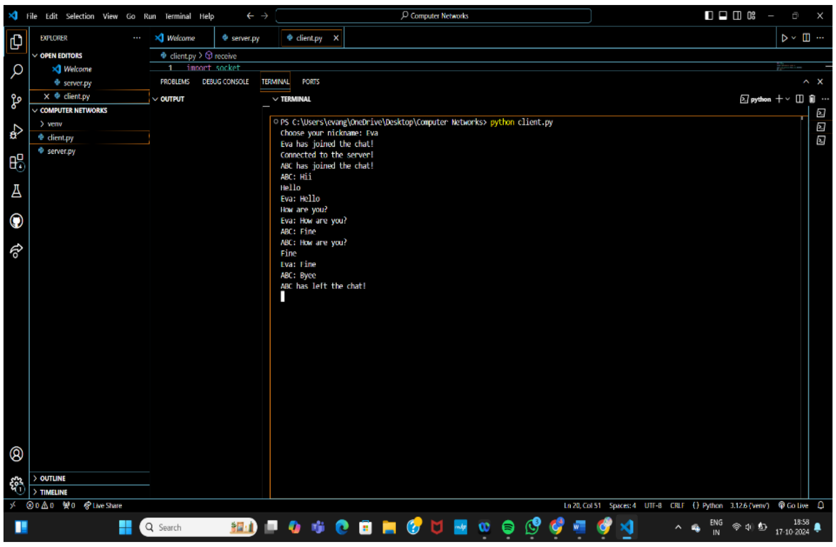


```python
import socket
import threading

nickname = input("Choose your nickname: ")

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
host = '127.0.0.1'
port = 5000
client.connect((host, port))

def receive():
    while True:
        try:
            message = client.recv(1024).decode('utf-8')
            if message == 'NICK':
                client.send(nickname.encode('utf-8'))
            else:
                print(message)
        except:
            print("Disconnected from the server!")
            client.close()
            break

def write():
    while True:
        try:
            message = f'{nickname}: {input("")}'
            client.send(message.encode('utf-8'))
        except:
            print("You have been disconnected!")
            client.close()
            break

receive_thread = threading.Thread(target=receive)
receive_thread.start()
write_thread = threading.Thread(target=write)
write_thread.start()
```

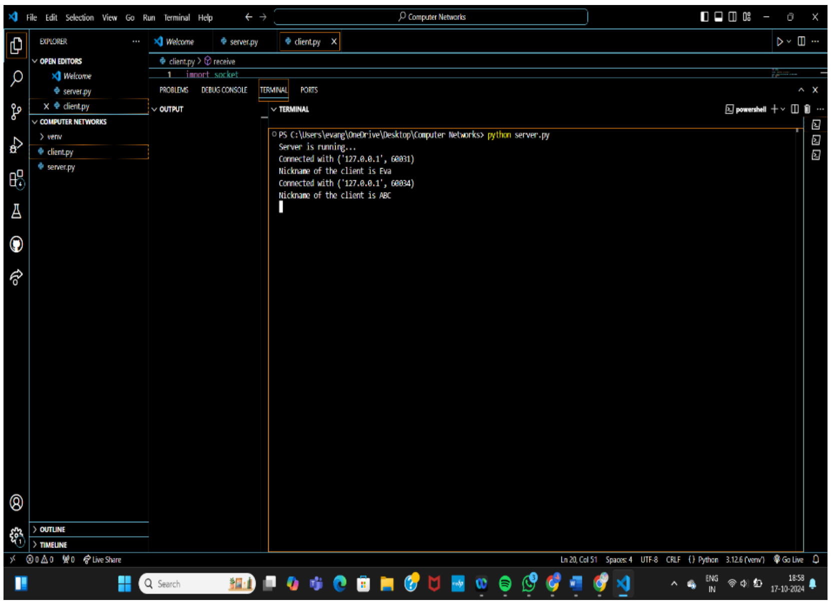
Output:



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



```
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: Hii
Hello
Eva: Hello
How are you?
Eva: How are you?
ABC: Fine
ABC: How are you?
Fine
Eva: Fine
ABC: Byee
ABC has left the chat!
```



```
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
```