server.py

client.py

**Server-Side:**

1. **Libraries:** You'll need libraries like socket for network communication and potentially threading for handling multiple clients.
2. **IP Address:** Instead of a local IP (like 192.168.1.X), the server needs your public IP address. This can be found by searching "what is my ip" online.
3. **Port Forwarding:** Your router needs to be configured to forward incoming traffic on a specific port (e.g., 5000) to the computer running the server. Consult your router's manual for port forwarding instructions.
4. **Security:** A WAN chat room exposes your server to the internet. Consider user authentication and message encryption for a production environment. For learning purposes, a simple implementation might suffice, but be aware of the security risks.

**Client-Side:**

1. **Public IP:** The client needs to connect to the server's public IP address and the chosen port number.

Here's a basic example structure (remember, this is for educational purposes and lacks security features):

**Server (server.py):**

import socket

import threading

HOST = 'YOUR\_PUBLIC\_IP' # Replace with your actual public IP

PORT = 5000

# Function to handle each client connection

def handle\_client(conn, addr):

print(f"[CONNECTED] {addr}")

while True:

message = conn.recv(1024).decode('utf-8')

if message:

broadcast(message, conn)

else:

break

conn.close()

print(f"[DISCONNECTED] {addr}")

def broadcast(message, sender):

for client in clients:

if client != sender:

try:

client.send(message.encode('utf-8'))

except:

clients.remove(client)

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.bind((HOST, PORT))

s.listen()

clients = []

print(f"Server listening on {HOST}:{PORT}")

while True:

conn, addr = s.accept()

clients.append(conn)

thread = threading.Thread(target=handle\_client, args=(conn, addr))

thread.start()

**Client (client.py):**

import socket

HOST = 'SERVER\_PUBLIC\_IP' # Replace with server's public IP

PORT = 5000

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

s.connect((HOST, PORT))

username = input("Enter Username: ")

while True:

message = input(f"{username}> ")

s.send(f"{username}: {message}".encode('utf-8'))

data = s.recv(1024).decode('utf-8')

if data:

print(data)

else:

break