Awesome project idea 🎯 Let’s build a **“Python Chat App (LAN)”** that works over a local network with **socket**, **threading**, and a **Tkinter UI**. We’ll create a **server** and a **client** script.

**Project Structure**

python-chat-app/

server.py # Socket server (multi-client)

client.py # Tkinter chat client

chat\_logs.txt # Saved chat logs

requirements.txt # Needed libraries

README.md # Usage guide

**requirements.txt**

tkinter # comes built-in with Python

**server.py**

import socket

import threading

import datetime

# -----------------------

# Config

# -----------------------

HOST = "0.0.0.0" # Listen on all interfaces

PORT = 5000

clients = []

LOG\_FILE = "chat\_logs.txt"

# -----------------------

# Helpers

# -----------------------

def log\_message(message):

with open(LOG\_FILE, "a") as f:

f.write(f"{datetime.datetime.now()} - {message}\n")

def broadcast(message, client\_socket=None):

log\_message(message)

for client in clients:

if client != client\_socket:

try:

client.send(message.encode())

except:

clients.remove(client)

def handle\_client(client\_socket, addr):

name = client\_socket.recv(1024).decode()

welcome\_msg = f"{name} joined the chat."

print(welcome\_msg)

broadcast(welcome\_msg)

while True:

try:

msg = client\_socket.recv(1024).decode()

if msg.startswith("/exit"):

goodbye = f"{name} has left the chat."

broadcast(goodbye, client\_socket)

print(goodbye)

clients.remove(client\_socket)

client\_socket.close()

break

else:

message = f"{name}: {msg}"

print(message)

broadcast(message, client\_socket)

except:

clients.remove(client\_socket)

client\_socket.close()

break

# -----------------------

# Main Server

# -----------------------

def start\_server():

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind((HOST, PORT))

server.listen()

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

while True:

client\_socket, addr = server.accept()

clients.append(client\_socket)

threading.Thread(target=handle\_client, args=(client\_socket, addr)).start()

if \_\_name\_\_ == "\_\_main\_\_":

start\_server()

**client.py**

import socket

import threading

import tkinter as tk

from tkinter import simpledialog, scrolledtext, messagebox

# -----------------------

# Config

# -----------------------

HOST = "127.0.0.1" # Server IP (change for LAN)

PORT = 5000

# -----------------------

# Client Class

# -----------------------

class ChatClient:

def \_\_init\_\_(self, master):

self.master = master

self.master.title("LAN Chat App")

self.text\_area = scrolledtext.ScrolledText(master, state="disabled", width=50, height=20)

self.text\_area.pack(padx=10, pady=10)

self.msg\_entry = tk.Entry(master, width=40)

self.msg\_entry.pack(side=tk.LEFT, padx=10, pady=10)

self.msg\_entry.bind("<Return>", self.send\_message)

self.send\_btn = tk.Button(master, text="Send", command=self.send\_message)

self.send\_btn.pack(side=tk.LEFT)

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.sock.connect((HOST, PORT))

self.name = simpledialog.askstring("Name", "Enter your chat name:", parent=master)

self.sock.send(self.name.encode())

self.running = True

threading.Thread(target=self.receive\_messages).start()

def send\_message(self, event=None):

msg = self.msg\_entry.get()

if msg.strip():

if msg == "/exit":

self.sock.send(msg.encode())

self.sock.close()

self.master.quit()

else:

self.sock.send(msg.encode())

self.msg\_entry.delete(0, tk.END)

def receive\_messages(self):

while self.running:

try:

msg = self.sock.recv(1024).decode()

self.text\_area.config(state="normal")

self.text\_area.insert(tk.END, msg + "\n")

self.text\_area.config(state="disabled")

self.text\_area.yview(tk.END)

except:

break

# -----------------------

# Run Client

# -----------------------

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

client = ChatClient(root)

root.protocol("WM\_DELETE\_WINDOW", lambda: (client.sock.send("/exit".encode()), root.destroy()))

root.mainloop()

**README.md**

# 💬 Python Chat App (LAN)

## Objective

Create a local real-time chat app.

## Tools

- `socket` – networking

- `threading` – handle multiple clients

- `tkinter` – chat UI

## Features

1. Socket server supports multiple clients.

2. Tkinter client with chat input/output.

3. Join/leave notifications.

4. Commands:

- `/exit` → leave the chat

- (You can extend with `/mute`)

5. Logs all chats to `chat\_logs.txt`.

## Run

### Start the server:

```bash

python server.py

**Start clients (on same LAN):**

python client.py

💡 For LAN: Change HOST in client.py to the server machine’s IP (e.g., 192.168.1.100).

**Deliverables**

* Server & client code
* 2-user test (run 2x client.py)
* Logs in chat\_logs.txt

---

✅ \*\*Deliverables included\*\*:

- `server.py` (multi-client server)

- `client.py` (Tkinter UI client)

- `chat\_logs.txt` (chat log file)

- `/exit` command working

---