#Server Side:

import socket

import threading

import pickle

import os

import sys

chat\_groups = {}

file\_transfer\_condition = threading.Condition()

class ChatGroup:

def \_\_init\_\_(self, admin, client\_socket):

self.admin = admin

self.clients = {}

self.offline\_messages = {}

self.all\_members = set()

self.online\_members = set()

self.join\_requests = set()

self.wait\_clients = {}

self.clients[admin] = client\_socket

self.all\_members.add(admin)

self.online\_members.add(admin)

def disconnect\_member(self, username):

self.online\_members.remove(username)

del self.clients[username]

def connect\_member(self, username, client\_socket):

self.online\_members.add(username)

self.clients[username] = client\_socket

def send\_message(self, message, sender):

for member in self.online\_members:

if member != sender:

self.clients[member].send(bytes(sender + ": " + message, "utf-8"))

def handle\_chat(client\_socket, username, group\_name):

while True:

msg = client\_socket.recv(1024).decode("utf-8")

if msg == "/view\_requests":

client\_socket.send(b"/view\_requests")

client\_socket.recv(1024).decode("utf-8")

if username == chat\_groups[group\_name].admin:

client\_socket.send(b"/sending\_data")

client\_socket.recv(1024)

client\_socket.send(pickle.dumps(chat\_groups[group\_name].join\_requests))

else:

client\_socket.send(b"You're not an admin.")

elif msg == "/approve\_request":

client\_socket.send(b"/approve\_request")

client\_socket.recv(1024).decode("utf-8")

if username == chat\_groups[group\_name].admin:

client\_socket.send(b"/proceed")

username\_to\_approve = client\_socket.recv(1024).decode("utf-8")

if username\_to\_approve in chat\_groups[group\_name].join\_requests:

chat\_groups[group\_name].join\_requests.remove(username\_to\_approve)

chat\_groups[group\_name].all\_members.add(username\_to\_approve)

if username\_to\_approve in chat\_groups[group\_name].wait\_clients:

chat\_groups[group\_name].wait\_clients[username\_to\_approve].send(b"/accepted")

chat\_groups[group\_name].connect\_member(username\_to\_approve, chat\_groups[group\_name].wait\_clients[username\_to\_approve])

del chat\_groups[group\_name].wait\_clients[username\_to\_approve]

print("Member Approved:", username\_to\_approve, "| Group:", group\_name)

client\_socket.send(b"User has been added to the group.")

else:

client\_socket.send(b"The user has not requested to join.")

else:

client\_socket.send(b"You're not an admin.")

elif msg == "/disconnect":

client\_socket.send(b"/disconnect")

client\_socket.recv(1024).decode("utf-8")

chat\_groups[group\_name].disconnect\_member(username)

print("User Disconnected:", username, "| Group:", group\_name)

break

elif msg == "/message\_send":

client\_socket.send(b"/message\_send")

message = client\_socket.recv(1024).decode("utf-8")

chat\_groups[group\_name].send\_message(message, username)

elif msg == "/wait\_disconnect":

client\_socket.send(b"/wait\_disconnect")

del chat\_groups[group\_name].wait\_clients[username]

print("Waiting Client:", username, "Disconnected")

break

elif msg == "/all\_members":

client\_socket.send(b"/all\_members")

client\_socket.recv(1024).decode("utf-8")

client\_socket.send(pickle.dumps(chat\_groups[group\_name].all\_members))

elif msg == "/online\_members":

client\_socket.send(b"/online\_members")

client\_socket.recv(1024).decode("utf-8")

client\_socket.send(pickle.dumps(chat\_groups[group\_name].online\_members))

elif msg == "/change\_admin":

client\_socket.send(b"/change\_admin")

client\_socket.recv(1024).decode("utf-8")

if username == chat\_groups[group\_name].admin:

client\_socket.send(b"/proceed")

new\_admin\_username = client\_socket.recv(1024).decode("utf-8")

if new\_admin\_username in chat\_groups[group\_name].all\_members:

chat\_groups[group\_name].admin = new\_admin\_username

print("New Admin:", new\_admin\_username, "| Group:", group\_name)

client\_socket.send(b"Your adminship is now transferred to the specified user.")

else:

client\_socket.send(b"The user is not a member of this group.")

else:

client\_socket.send(b"You're not an admin.")

elif msg == "/who\_admin":

client\_socket.send(b"/who\_admin")

group\_name = client\_socket.recv(1024).decode("utf-8")

client\_socket.send(bytes("Admin: " + chat\_groups[group\_name].admin, "utf-8"))

elif msg == "/kick\_member":

client\_socket.send(b"/kick\_member")

client\_socket.recv(1024).decode("utf-8")

if username == chat\_groups[group\_name].admin:

client\_socket.send(b"/proceed")

username\_to\_kick = client\_socket.recv(1024).decode("utf-8")

if username\_to\_kick in chat\_groups[group\_name].all\_members:

chat\_groups[group\_name].all\_members.remove(username\_to\_kick)

if username\_to\_kick in chat\_groups[group\_name].online\_members:

chat\_groups[group\_name].clients[username\_to\_kick].send(b"/kicked")

chat\_groups[group\_name].online\_members.remove(username\_to\_kick)

del chat\_groups[group\_name].clients[username\_to\_kick]

print("User Removed:", username\_to\_kick, "| Group:", group\_name)

client\_socket.send(b"The specified user is removed from the group.")

else:

client\_socket.send(b"The user is not a member of this group.")

else:

client\_socket.send(b"You're not an admin.")

elif msg == "/file\_transfer":

client\_socket.send(b"/file\_transfer")

filename = client\_socket.recv(1024).decode("utf-8")

if filename == "~error~":

continue

client\_socket.send(b"/send\_file")

remaining = int.from\_bytes(client\_socket.recv(4), 'big')

f = open(filename, "wb")

while remaining:

data = client\_socket.recv(min(remaining, 4096))

remaining -= len(data)

f.write(data)

f.close()

print("File received:", filename, "| User:", username, "| Group:", group\_name)

for member in chat\_groups[group\_name].online\_members:

if member != username:

member\_client = chat\_groups[group\_name].clients[member]

member\_client.send(b"/receive\_file")

with file\_transfer\_condition:

file\_transfer\_condition.wait()

member\_client.send(bytes(filename, "utf-8"))

with file\_transfer\_condition:

file\_transfer\_condition.wait()

with open(filename, 'rb') as f:

data = f.read()

data\_len = len(data)

member\_client.send(data\_len.to\_bytes(4, 'big'))

member\_client.send(data)

client\_socket.send(bytes(filename + " successfully sent to all online group members.", "utf-8"))

print("File sent", filename, "| Group: ", group\_name)

os.remove(filename)

elif msg == "/send\_filename" or msg == "/send\_file":

with file\_transfer\_condition:

file\_transfer\_condition.notify()

else:

print("UNIDENTIFIED COMMAND:", msg)

def handshake(client\_socket):

username = client\_socket.recv(1024).decode("utf-8")

client\_socket.send(b"/send\_groupname")

group\_name = client\_socket.recv(1024).decode("utf-8")

if group\_name in chat\_groups:

if username in chat\_groups[group\_name].all\_members:

chat\_groups[group\_name].connect\_member(username, client\_socket)

client\_socket.send(b"/ready")

print("User Connected:", username, "| Group:", group\_name)

else:

chat\_groups[group\_name].join\_requests.add(username)

chat\_groups[group\_name].wait\_clients[username] = client\_socket

chat\_groups[group\_name].send\_message(username + " has requested to join the group.", "BAATKARO")

client\_socket.send(b"/wait")

print("Join Request:", username, "| Group:", group\_name)

threading.Thread(target=handle\_chat, args=(client\_socket, username, group\_name,)).start()

else:

chat\_groups[group\_name] = ChatGroup(username, client\_socket)

threading.Thread(target=handle\_chat, args=(client\_socket, username, group\_name,)).start()

client\_socket.send(b"/admin\_ready")

print("New Group:", group\_name, "| Admin:", username)

def main():

if len(sys.argv) < 3:

print("USAGE: python server.py <IP> <Port>")

print("EXAMPLE: python server.py localhost 8888")

return

listen\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

listen\_socket.bind((sys.argv[1], int(sys.argv[2])))

listen\_socket.listen(10)

print("BAATKARO Server running")

while True:

client\_socket, \_ = listen\_socket.accept()

threading.Thread(target=handshake, args=(client\_socket,)).start()

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

main()

#Client Side

import socket

import threading

import pickle

import sys

client\_state = {}

def listen\_to\_server(server\_socket):

while True:

msg = server\_socket.recv(1024).decode("utf-8")

if msg == "/view\_requests":

server\_socket.send(bytes(".", "utf-8"))

response = server\_socket.recv(1024).decode("utf-8")

if response == "/sending\_data":

server\_socket.send(b"/ready\_for\_data")

data = pickle.loads(server\_socket.recv(1024))

if data == set():

print("No pending requests.")

else:

print("Pending Requests:")

for element in data:

print(element)

else:

print(response)

elif msg == "/approve\_request":

server\_socket.send(bytes(".", "utf-8"))

response = server\_socket.recv(1024).decode("utf-8")

if response == "/proceed":

client\_state["input\_message"] = False

print("Please enter the username to approve: ")

with client\_state["input\_condition"]:

client\_state["input\_condition"].wait()

client\_state["input\_message"] = True

server\_socket.send(bytes(client\_state["user\_input"], "utf-8"))

print(server\_socket.recv(1024).decode("utf-8"))

else:

print(response)

elif msg == "/disconnect":

server\_socket.send(bytes(".", "utf-8"))

client\_state["is\_alive"] = False

break

elif msg == "/message\_send":

server\_socket.send(bytes(client\_state["user\_input"], "utf-8"))

client\_state["send\_message\_lock"].release()

elif msg == "/all\_members":

server\_socket.send(bytes(".", "utf-8"))

data = pickle.loads(server\_socket.recv(1024))

print("All Group Members:")

for element in data:

print(element)

elif msg == "/online\_members":

server\_socket.send(bytes(".", "utf-8"))

data = pickle.loads(server\_socket.recv(1024))

print("Online Group Members:")

for element in data:

print(element)

elif msg == "/change\_admin":

server\_socket.send(bytes(".", "utf-8"))

response = server\_socket.recv(1024).decode("utf-8")

if response == "/proceed":

client\_state["input\_message"] = False

print("Please enter the username of the new admin: ")

with client\_state["input\_condition"]:

client\_state["input\_condition"].wait()

client\_state["input\_message"] = True

server\_socket.send(bytes(client\_state["user\_input"], "utf-8"))

print(server\_socket.recv(1024).decode("utf-8"))

else:

print(response)

elif msg == "/who\_admin":

server\_socket.send(bytes(client\_state["group\_name"], "utf-8"))

print(server\_socket.recv(1024).decode("utf-8"))

elif msg == "/kick\_member":

server\_socket.send(bytes(".", "utf-8"))

response = server\_socket.recv(1024).decode("utf-8")

if response == "/proceed":

client\_state["input\_message"] = False

print("Please enter the username to kick: ")

with client\_state["input\_condition"]:

client\_state["input\_condition"].wait()

client\_state["input\_message"] = True

server\_socket.send(bytes(client\_state["user\_input"], "utf-8"))

print(server\_socket.recv(1024).decode("utf-8"))

else:

print(response)

elif msg == "/kicked":

client\_state["is\_alive"] = False

client\_state["input\_message"] = False

print("You have been kicked. Press any key to quit.")

break

elif msg == "/file\_transfer":

client\_state["input\_message"] = False

print("Please enter the filename: ")

with client\_state["input\_condition"]:

client\_state["input\_condition"].wait()

client\_state["input\_message"] = True

filename = client\_state["user\_input"]

try:

f = open(filename, 'rb')

f.close()

except FileNotFoundError:

print("The requested file does not exist.")

server\_socket.send(bytes("~error~", "utf-8"))

continue

server\_socket.send(bytes(filename, "utf-8"))

server\_socket.recv(1024)

print("Uploading file to server...")

with open(filename, 'rb') as f:

data = f.read()

data\_len = len(data)

server\_socket.send(data\_len.to\_bytes(4, 'big'))

server\_socket.send(data)

print(server\_socket.recv(1024).decode("utf-8"))

elif msg == "/receive\_file":

print("Receiving shared group file...")

server\_socket.send(b"/send\_filename")

filename = server\_socket.recv(1024).decode("utf-8")

server\_socket.send(b"/send\_file")

remaining = int.from\_bytes(server\_socket.recv(4), 'big')

f = open(filename, "wb")

while remaining:

data = server\_socket.recv(min(remaining, 4096))

remaining -= len(data)

f.write(data)

f.close()

print("Received file saved as", filename)

else:

print(msg)

def get\_user\_input(server\_socket):

while client\_state["is\_alive"]:

client\_state["send\_message\_lock"].acquire()

client\_state["user\_input"] = input()

client\_state["send\_message\_lock"].release()

with client\_state["input\_condition"]:

client\_state["input\_condition"].notify()

if client\_state["user\_input"] == "/1":

server\_socket.send(b"/view\_requests")

elif client\_state["user\_input"] == "/2":

server\_socket.send(b"/approve\_request")

elif client\_state["user\_input"] == "/3":

server\_socket.send(b"/disconnect")

break

elif client\_state["user\_input"] == "/4":

server\_socket.send(b"/all\_members")

elif client\_state["user\_input"] == "/5":

server\_socket.send(b"/online\_members")

elif client\_state["user\_input"] == "/6":

server\_socket.send(b"/change\_admin")

elif client\_state["user\_input"] == "/7":

server\_socket.send(b"/who\_admin")

elif client\_state["user\_input"] == "/8":

server\_socket.send(b"/kick\_member")

elif client\_state["user\_input"] == "/9":

server\_socket.send(b"/file\_transfer")

elif client\_state["input\_message"]:

client\_state["send\_message\_lock"].acquire()

server\_socket.send(b"/message\_send")

def wait\_server\_listen(server\_socket):

while not client\_state["is\_alive"]:

msg = server\_socket.recv(1024).decode("utf-8")

if msg == "/accepted":

client\_state["is\_alive"] = True

print("Your join request has been approved. Press any key to begin chatting.")

break

elif msg == "/wait\_disconnect":

client\_state["join\_disconnect"] = True

break

def wait\_user\_input(server\_socket):

while not client\_state["is\_alive"]:

client\_state["user\_input"] = input()

if client\_state["user\_input"] == "/1" and not client\_state["is\_alive"]:

server\_socket.send(b"/wait\_disconnect")

break

def main():

if len(sys.argv) < 3:

print("USAGE: python client.py <IP> <Port>")

print("EXAMPLE: python client.py localhost 8000")

return

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

server\_socket.connect((sys.argv[1], int(sys.argv[2])))

client\_state["input\_condition"] = threading.Condition()

client\_state["send\_message\_lock"] = threading.Lock()

client\_state["username"] = input("Welcome to PyconChat! Please enter your username: ")

client\_state["group\_name"] = input("Please enter the name of the group: ")

client\_state["is\_alive"] = False

client\_state["join\_disconnect"] = False

client\_state["input\_message"] = True

server\_socket.send(bytes(client\_state["username"], "utf-8"))

server\_socket.recv(1024)

server\_socket.send(bytes(client\_state["group\_name"], "utf-8"))

response = server\_socket.recv(1024).decode("utf-8")

if response == "/admin\_ready":

print("You have created the group", client\_state["group\_name"], "and are now an admin.")

client\_state["is\_alive"] = True

elif response == "/ready":

print("You have joined the group", client\_state["group\_name"])

client\_state["is\_alive"] = True

elif response == "/wait":

print("Your request to join the group is pending admin approval.")

print("Available Commands:\n/1 -> Disconnect\n")

wait\_user\_input\_thread = threading.Thread(target=wait\_user\_input, args=(server\_socket,))

wait\_server\_listen\_thread = threading.Thread(target=wait\_server\_listen, args=(server\_socket,))

user\_input\_thread = threading.Thread(target=get\_user\_input, args=(server\_socket,))

listen\_to\_server\_thread = threading.Thread(target=listen\_to\_server, args=(server\_socket,))

wait\_user\_input\_thread.start()

wait\_server\_listen\_thread.start()

while True:

if client\_state["is\_alive"] or client\_state["join\_disconnect"]:

break

if client\_state["is\_alive"]:

print("Available Commands:\n/1 -> View Join Requests (Admins)\n/2 -> Approve Join Requests (Admin)\n/3 -> Disconnect\n/4 -> View All Members\n/5 -> View Online Group Members\n/6 -> Transfer Adminship\n/7 -> Check Group Admin\n/8 -> Kick Member\n/9 -> File Transfer\nType anything else to send a message")

wait\_user\_input\_thread.join()

wait\_server\_listen\_thread.join()

user\_input\_thread.start()

listen\_to\_server\_thread.start()

while True:

if client\_state["join\_disconnect"]:

server\_socket.shutdown(socket.SHUT\_RDWR)

server\_socket.close()

wait\_user\_input\_thread.join()

wait\_server\_listen\_thread.join()

print("Disconnected from PyconChat.")

break

elif not client\_state["is\_alive"]:

server\_socket.shutdown(socket.SHUT\_RDWR)

server\_socket.close()

user\_input\_thread.join()

listen\_to\_server\_thread.join()

print("Disconnected from PyconChat.")

break

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

main()