Title (Simple Chat Application using Python)

Dr.siba alia li
Hiba sleem mohmmad

□ ABSTRACT □

Chat refers to the process of communicating, interacting and/or exchanging messages over the Internet. It involves two or more individuals that communicate through a chat-enabled service or software. Chat may be delivered through text, audioor video communication via the Internet.A chat application has basic two components, viz server and client. A server is a computer program or a device that provides functionality for other programs or devices. Clients who want to chat with each other connect to the server.The chat application we are going to make will be more like a chat room, rather than a peer to peer chat. So this means that multiple users can connect to the chat server and send their messages. Every message is broadcasted to every connected chat user

Keywords: chat , python ,server/client ,sender, receiver ,application ,GUI.

العنوان (تطبيق دردشة بسيط بأستخدام البايثون)

المؤلف* صبا علي علي
هبة سليم محمد

□ ملخّص □

تشير الدردشة إلى عملية الاتصال و / أو التفاعل و / أو تبادل الرسائل عبر الإنترنت. وهي تتضمن شخصين أو أكثر يتواصلون من خلال خدمة أو برنامج يدعم الدردشة. يمكن تسليم الدردشة من خلال الرسائل النصية أو الصوتية أو المرئية عبر الإنترنت. يحتوي تطبيق الدردشة على مكونين أساسيين ، الخادم والعميل. الخادم هو برنامج كمبيوتر أو جهاز يوفر وظائف لبرامج أو أجهزة أخرى. العملاء الذين يريدون الدردشة مع بعضهم البعض يتصلون بالخادم. سيكون تطبيق الدردشة الذي سننشئه أشبه بغرفة محادثة ، وليس دردشة نظير إلى نظير. هذا يعني أنه يمكن لعدة مستخدمين الاتصال بخادم الدردشة وإرسال رسائلهم. يتم بث كل رسالة إلى كل مستخدم دردشة متصل

الكلمات المفتاحية: GUI, application, receiver, sender, server/client, python , chat.

مقدمة:

المحادثة الفورية هي مجموعة تقنيات الاتصالات التي تستخدم من أجل المحادثات النصية بين مشترك أو أكثر عبر الانترنت أو أي نوع آخر من الشبكات.

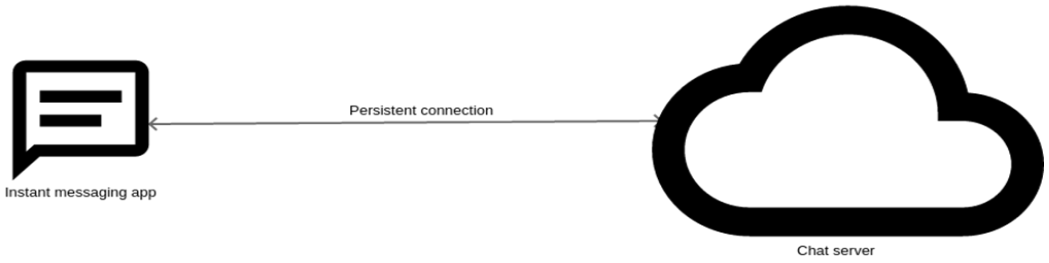
تختلف هذه التقنية عن البريد الإلكتروني بكونها تتم في الوقت الحقيقي مع ميزات كثيرة كإمكانية رؤية المشتركين لبعضهم عبر الكاميرا أو التواصل الصوتي ونقل الملفات وغيرها.

وغيرها وانتشرت بشكل هائل وذلك بعد facebook messenger و whats app ظهرت تطبيقات المحادثة مثل انتشار الهواتف الذكية وأصبحت هذه التطبيقات جزءا أساسيا من حياتنا اليومية.

مكونات تطبيق المحادثة:

تتألف تطبيقات المحادثة من ثلاثة أجزاء رئيسية وهي:

1. **برنامج المراسلة(client)**: يمثل الجزء الخاص بالزبون ويتألف من التعليمات البرمجية المتعلقة بالاتصال بالمخدم ومعالجة الرسائل و تصميم واجهة المستخدم.
2. **المخدم(server)**: والذي يقوم بالاستماع لاتصالات الزبائن وتوجيه الرسائل إلى المستخدمين وإدارتها.
3. **الاتصال المستمر**: وهو ضروري لعمل التطبيق حيث أن الزبون يجب أن يظل متصلا بالمخدم بشكل دائم.



الشكل(1):مكونات تطبيق المحادثة

آلية عمل تطبيقات المحادثة :

تمر تطبيقات المحادثة بالعديد من المراحل لإنشاء المحادثة وهي:

1. **connection الاتصال:**

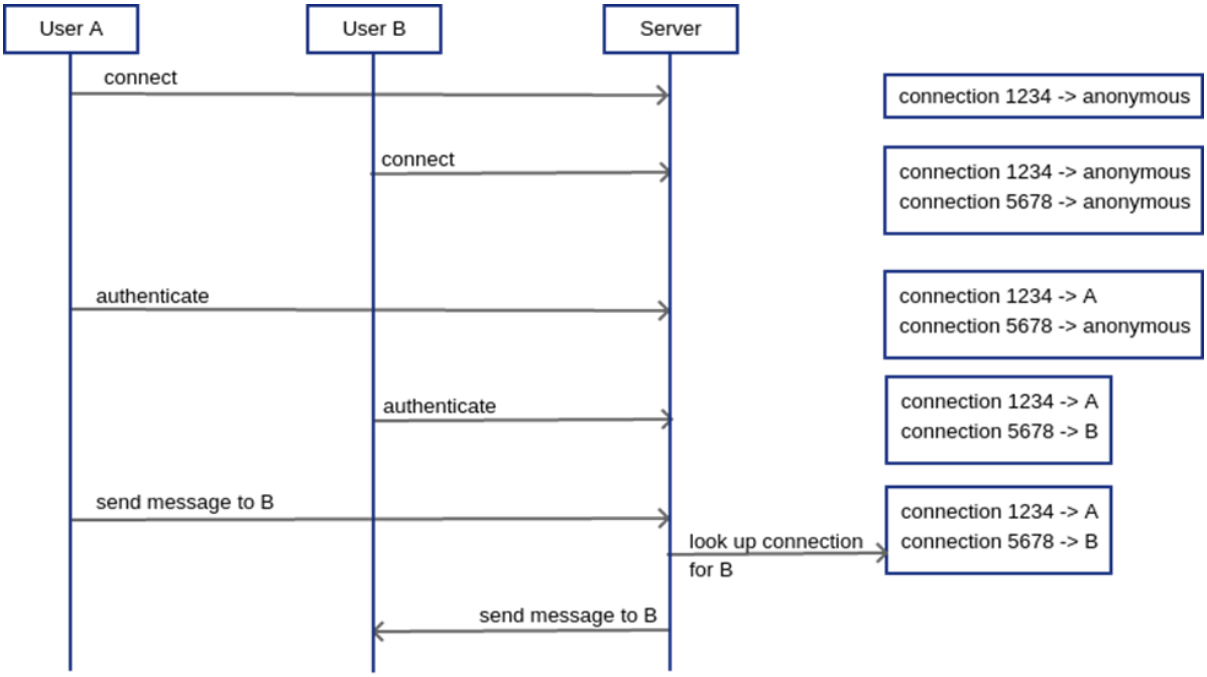
- وهي الخطوة الأولى التي يقوم بها التطبيق حيث يتم انشاء اتصال بين الزبون ومخدم معين تابع للشركة المالكة للتطبيق.

2. **التعريف Identification:**

- يقوم الزبون بالتعريف عن نفسه للمخدم حيث يقوم بعملية sign in وذلك بتقديم الوثائق الخاصة به كاسم المستخدم وكلمة السر، وهناك العديد من التطبيقات التي تقوم باستخدام رقم الهاتف وإرسال كلمة سر واحدة عند التسجيل في التطبيق لتقوم بعدها بتسجيل الدخول تلقائيا.
- حالما يتصل المستخدم ويعرف عن نفسه للمخدم يقوم المخدم بوضع المستخدم في خارطة الاتصالات ومع دخول عدة مستخدمين يقوم المخدم بتوجيه الرسائل بين المستخدمين وذلك اعتمادا على خارطة الاتصالات.

3. **تبادل الرسائل Exchange of Messages:**

• إذا أرسل المستخدم A إلى المستخدم B رسالة فإن هذه الرسالة تنتقل عبر الاتصال بين المخدم والمستخدم A والذي بدوره يستخدم خارطة المستخدمين لديه ويوجه هذه الرسالة عبر اتصال بين المخدم والمستخدم B إلى المستخدم B كما هو موضح في الشكل (2).



الشكل (2): يمثل تبادل الرسائل بين المستخدم A و B

تشفير البيانات :

- إن من أبرز الميزات التي يسعى مطورو تطبيقات المحادثة إلى تحسينها هي تشفير الرسائل في الاتصال بين المستخدمين حيث تم استخدام تشفير نهاية إلى نهاية end-to-end encryption والذي يقوم بتشفير الرسائل بمفتاح خاص بين المرسل والمستقبل وذلك لحماية البيانات من الاطلاع عليها من قبل طرف ثالث.

- تكمن أهمية التشفير في تطبيقات المحادثة أيضا في الحماية من هجمات التي تحدث على الاتصال كهجمات رجل في المنتصف man in the middle وغيرها.

تحديات تطبيقات المحادثة:

- هناك العديد من التحديات التي تواجهها تطبيقات المحادثة لعل أبرزها:

A. التوسعية Scalability: إن التزايد الهائل في عدد المستخدمين يتطلب إمكانية التوسع في البنية التحتية للتطبيق سواء من الناحية البرمجية أو العتاد.

B. الأمن Security: إن أبرز القضايا الأمنية التي تواجه تطبيقات المحادثة هي سرقة البيانات وانتحال الشخصية بالإضافة إلى العديد من القضايا الأخرى، لذلك تقوم الشركات المالكة للتطبيقات بتطوير آليات لحماية تطبيقاتها كالتشفير نهاية إلى نهاية والعديد من التقنيات الأخرى لحماية الخصوصية.

C. التكامل Integration: تسعى تطبيقات المحادثة إلى تطبيق آلية Cross-Platform عبور البيئات للسماح للمستخدمين بالتواصل بغض النظر عن بيئة عمل المستخدم.

فرضية الكود:

- يتألف البرنامج من ملفين برمجيين بلغة بايثون أحدهما يمثل المخدم والآخر يمثل الزبون.

- في كود المخدم قمنا باستخدام مكتبتي socket و threading حيث تم تعريف تابع لاستقبال اتصالات الزبائن وتابع للتعامل مع كل مستخدم على حدى ومن ثم تابع لإرسال الرسائل لجميع المستخدمين، إضافة إلى استخدام كائن socket وربطه بعنوان الجهاز المضيف ويستمع إلى اتصالات الزبائن عبر البورت غير المخصص ذي الرقم 33000 ، وإتاحة إمكانية اتصال أكثر من زبون باستخدام كائن threading.

- في كود الزبون أيضا استخدمنا المكتبات socket و threading إضافة إلى المكتبة tkinter حيث قمنا بتعريف تابع لاستقبال الرسائل وآخر لإرسالها وتابع في حالة الإغلاق من قبل المخدم عبر مدير النوافذ، وقمنا باستخدام المكتبة tkinter بتصميم واجهة بسيطة وقمنا بربطها بالكود وعرفنا كائن socket وسمحنا

للمستخدم بإدخال عنوان المخدم ورقم البورت، أيضا قمنا بإضافة كائن threading لإتاحة إمكانية تزامن الارسال والاستقبال.

- يمكن ترتيب العمليات التي تتم عند تنفيذ التطبيق إلى ما يلي:

1. الاتصال بالمخدم بإدخال عنوان المخدم ورقم البورت.
2. إدخال اسم المستخدم عبر النافذة tkinter .
3. بدء المحادثة بين المستخدمين.
4. إغلاق المحادثة واتصال المخدم عبر إدخال الأمر shutdown.

الكود:

- **كود المخدم:**

```
# server code

#import needed library
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread


#handling incoming connections
def accept_incoming_connections():
    while True:
        client, client_address = SERVER.accept()
        print("%s:%s has connected." % client_address)
        client.send(bytes("Greetings from the server! Now type your name and press enter!", "utf8"))
        addresses[client] = client_address
        Thread(target=handle_client, args=(client,)).start()

#handle every connectin alone
def handle_client(client):  # Takes client socket as argument.
    name = client.recv(BUFSIZ).decode("utf8")
    welcome = 'Welcome %s! If you ever want to quit, type shutdown to exit.' % name
    client.send(bytes(welcome, "utf8"))
    msg = "%s has joined the chat!" % name
    broadcast(bytes(msg, "utf8"))
    clients[client] = name
```

```
    while True:
        msg = client.recv(BUFSIZ)
        if msg != bytes("shutdown", "utf8"):
            broadcast(msg, name+": ")
        else:
            #client.send(bytes("shutdown", "utf8"))
            client.close()
            del clients[client]
            broadcast(bytes("%s has left the chat." % name, "utf8"))
            break

# Broadcasts a message to all the clients
def broadcast(msg, prefix=""):  # prefix is for name identification.
    for sock in clients:
        sock.send(bytes(prefix, "utf8")+msg)


clients = {}      # clients list
addresses = {}    # addresses of clients list

HOST = ''         # host ip
PORT = 33000      # port number
BUFSIZ = 1024     # buffer size
ADDR = (HOST, PORT)

SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(ADDR)     # link ip and port to socket object

if __name__ == "__main__":
    SERVER.listen(5)
    print("Waiting for connection...")
    ACCEPT_THREAD = Thread(target=accept_incoming_connections)
    ACCEPT_THREAD.start()
    ACCEPT_THREAD.join()
    SERVER.close()
```

- تعريف التوابع

Def accept incoming connection

هو تابع يقبل اتصالات المستخدمين كل مستخدم يتصل يقبله الخادم ويضيفه الى قائمة المستخدمين ويضيف عنوانه

Send byte

تحويل الرسائل من stringالى byteلان تابع ال sendلا ينقل الا byte

Thread (target:handle client)

توجيه التابع بعد تنفيذه الى targetللتعامل مع كل اتصال لوحده

Def handle client(client)

لاخذ السوكيت كبارمتر

Name=client
recv(1024)

تليها رسالة ترحيب للمستخدم باسمه

Msg=
Broadcast byte
msg

عرفت msgو من خلال تابع broadcastيرسلها الخادم الى كل chat

Client(client)=name

نضيف المستخدم الى قائمة المستخدمين

If msg!=shutdown/n

broadcast(msg,name)

كل رسالة استقبلتا يجب التأكد من انها ليست shutdownاذا تحقق الشرط يعمل broadcastويرسل الرساله مع اسم المستخدم اذا لم يتحقق ننتقل الى elseوتغلق الواجهة ويحذف ipالمستخدم من قائمة المستخدمين

Def
broadcast(msg,prefix)

تابع ال broadcastيأخذ بارمتر الرساله وبارمتر اسم المستخدم

نبحث عن اسم المستخدم في قائمة المستخدمين من خلال حلقة forاذا موجود برسل الاسم مع الرساله

Host=' '

Ipالخادم نفس ipالجهاز ثم عرفت قيمة البورت والقيمة العم اقراها من buffer و ip و port

Server=socket
(af,stream)

Server
bind(addr)

عرفت سوكيت من نوع tcpبعدا ربطتا بالعنوان (ipو(port

If name=="main"

اذا في اكتر من كود بنفس المجلد فهو الكود الرئيسي

Server.listen(5)

الخادم يستمع الى 5 اتصالات يليها threadلكي يستطيع اكثر من مستخدم ان يتواصل مع السيرفر

Accept

thread=thread(target)

يذهب مباشرة ال تابع قبول الاتصالات

join

ينتظر حتى ينتهي threadمن انتهاء ما يعمل

- **كود الزبون:**

```
#client code

#importing needed library
from socket import AF_INET, socket, SOCK_STREAM
from threading import Thread
import tkinter

#Handles receiving of messages
def receive():
        while True:
                try:

                        msg = client_socket.recv(BUFSIZ).decode("utf8")
                        msg_list.insert(tkinter.END, msg)   #add msg to the end of msg_list
                except OSError:  # Possibly client has left the chat
                        break

#Handles sending of messages
def send(event=None):  # event is passed by binders.
        msg = my_msg.get()
        my_msg.set("")  # Clears input field.
        client_socket.send(bytes(msg, "utf8"))
        if msg == "shutdown":
                client_socket.close()
                top.destroy()

#this function is to be called when the window is closed
def on_closing(event=None):
        top.destroy()

top = tkinter.Tk() #define new tkinter object
top.title("Chatter") #give title to the window
top.geometry('600x400') # define shape of window

messages_frame = tkinter.Frame(top) # define message frame object
my_msg = tkinter.StringVar()  # For the messages to be sent
my_msg.set("")
scrollbar = tkinter.Scrollbar(messages_frame)  # To navigate through past messages.
```

```
# Following will contain the messages.
#define messages listbox object
msg_list = tkinter.Listbox(messages_frame, height=15, width=60, yscrollcommand=scrollbar.set)
scrollbar.pack(side=tkinter.RIGHT, fill=tkinter.Y)  # add scrollbar to object
msg_list.pack(side=tkinter.LEFT, fill=tkinter.BOTH) # add place of messages to object
msg_list.pack()  # add msg_list to the window
messages_frame.pack()  # add message frame to window

#define entry field object link to window and set variable
entry_field = tkinter.Entry(top, textvariable=my_msg)
entry_field.bind("<Return>", send)  # define that the entry field change call send() function
entry_field.pack()    # add entry field to window
#define button with name send and command send()
send_button = tkinter.Button(top, text="Send", command=send)
send_button.pack()   #add button to window
#define what happen when user close window it is call on_closing() function
top.protocol("WM_DELETE_WINDOW", on_closing)

#sockets part
HOST = input('Enter host: ')
PORT = input('Enter port: ')
if not PORT:
        PORT = 33000
else:
        PORT = int(PORT)

BUFSIZ = 1024
ADDR = (HOST, PORT)

client_socket = socket(AF_INET, SOCK_STREAM)
client_socket.connect(ADDR)

receive_thread = Thread(target=receive)  #define Thread object and set target
receive_thread.start() # start thread excution
tkinter.mainloop()  # Starts GUI execution as infinite loop
```

- استخدمنا مكاتب ,socket ,threadingومكتبة thinter هي مكتبة للواجهات
#handles_receiving of massage •
- (استقبال الرسائل)
- يتم استلام الرسالة ثم فك ترميز msg
- ثم نضاف الرسالة الى framالرسائل
Tkinter.end •
- لأضافتها الى اخر صندوق الرسائل
-
except of Error •
- اجل في حال ترك مستخدم البرنامج (غادر) لا ينهار البرنامج
#Handles sending of message •
- (ارسال الرسائل)
msg=mg_msg.get() •
-
- تابع ال (get موجود في Tkinter يأخذ الرسالة التي في مربع الادخال ويعرضها
mg_msg.set(“ “) •
- مكان ادخال الرسالة فاضي
-
- باستخدام send برسل الرسالة وبختبر اذا كانت “shutdown”

- بنشأ نافذة اسمها Top

- تعليمة (event=none)
- أي لا يتم تفعيل أي شيء بالنافذة بمجرد تمرير المؤشر عليها فقط تنفذ اذا نقرنا عليها بالماوس
- (((نشغل الواجهة)))
-

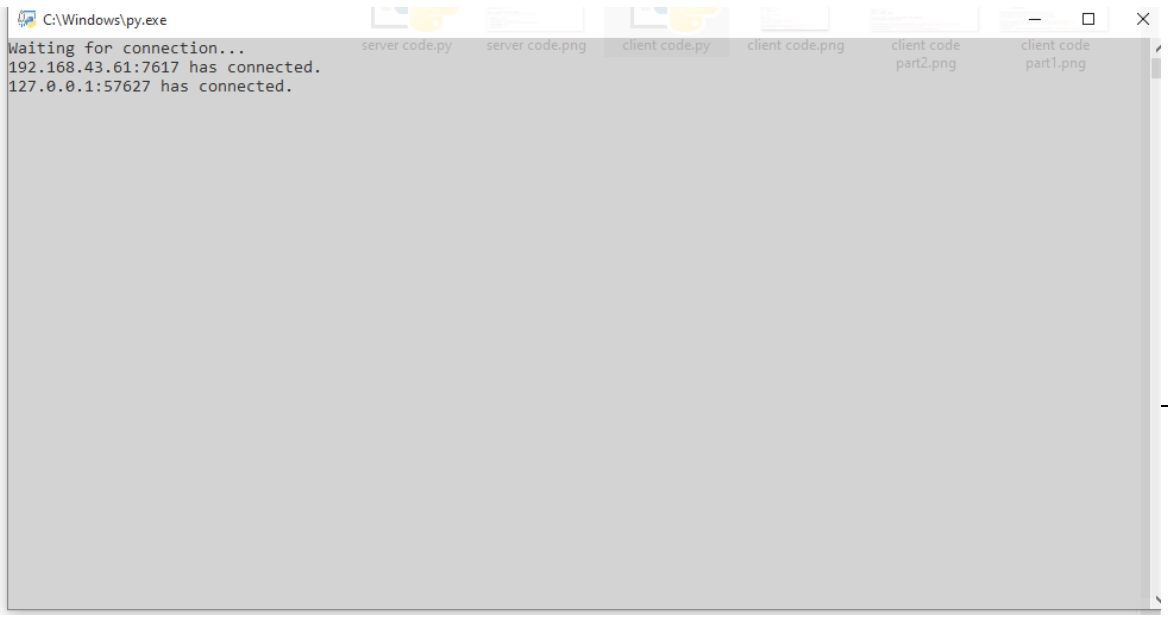
نعرف ثابت messages_frameفيه كل الرسائل وبعرف نوع البيانات المدخلة على انها string ونعرف الرسالة بالبداية تكون فارغة

- تعليمة(scrollbar= tkinter.serollbar(messages_frame
- هو مستطيل لي من خلاله بنتقل بين الرسائل في حال كان عندي رسائل كتير
- #define message list box object
- تعريفات وشروط ادخال
-
- #define entry field object link to window and set variable
- *تعريف مربع ادخال نصي
- #define button with name send and command send()
- *تعريف زر ارسال واخبارة انه للارسال يرسل عند النقر عليه
-

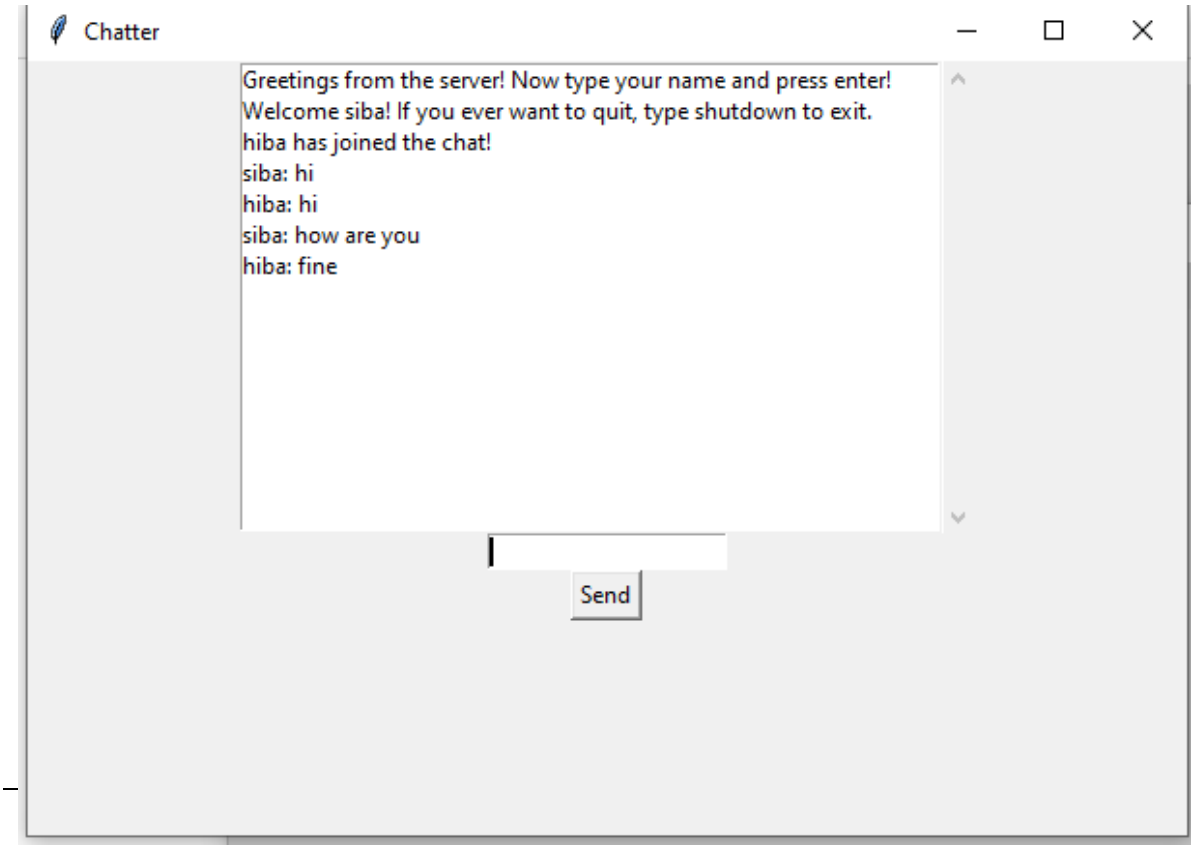
-
- Top .protocol ()
- مربع الاكس بالزاوية اذا نقرنا عليه بيغلق النوافذ (delet window)
- On _closing
- تابع بسكر النافذة
- #socket part
- فيه host ,port يكتبها المستخدم من اجل في حال تغير السيرفر الذي يعمل عليه البرنامج
- اذا لم ندخل port يأخذ تلقائيا 33000
- عرفنا بعدها socketوربطناها بالعنوان
- يستقبل بالبداية وبعدها يبدأ thread وتبدأ الواجهة (tkinter)

النتائج:

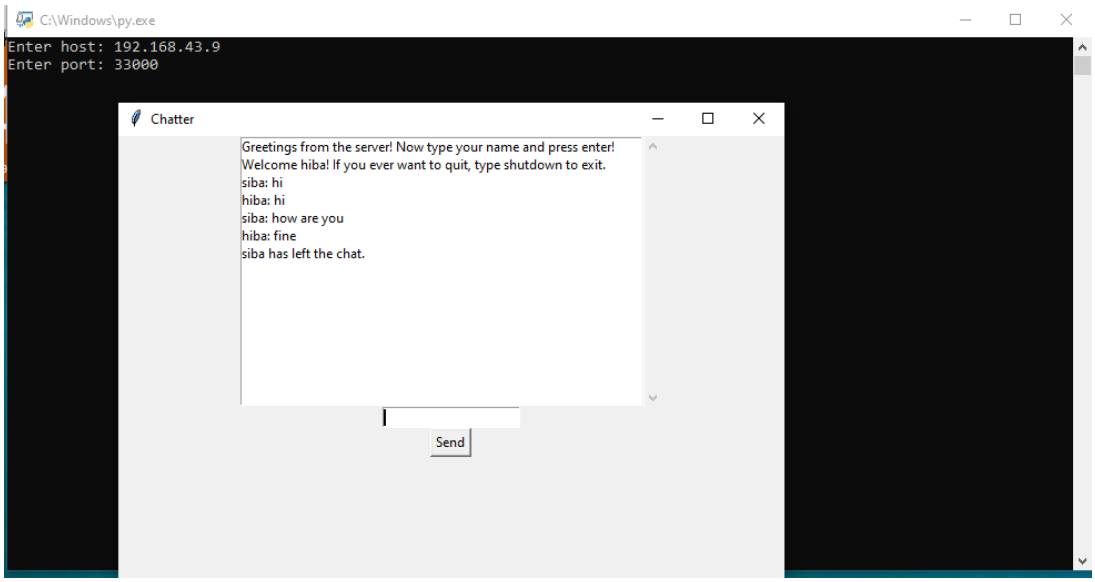
خرج كود السيرفر:



خرج كود ال client1:



خرج كود ال2 client:



References

[1] Rouhiainen, L. (2018). Artificial Intelligence: 101 things you must know today about our future. Lasse Rouhiainen.

[2] Sodhi, P., Awasthi, N., & Sharma, V. (2019). Introduction to machine learning and its basic application in python. In Proceedings of 10th International Conference on Digital Strategies for Organizational Success.

[3] Zhou, W. (2017). An end-to-end encryption plugin for video call software (Doctoral dissertation, Northeastern University).

[4] Sreeharsha, A. S. S. K., Kesapragada, S. M., & Chalamalasetty, S. P. (2022). Building Chatbot Using Amazon Lex and Integrating with A Chat Application. International Journal of Scientific Research in Engineering and Management (IJSREM), 6(04).