File Sharing App - 3

What is our GOAL for this MODULE?

In this class, we have applied our knowledge of FTP and we did the second part of the File sharing app. During the second part of the module, we discussed how to connect and disconnect with clients and how to make their button work. The goal of this module is to learn how to make our GUI buttons work.

What did we ACHIEVE in the class TODAY?

- Connect Button function
- Disconnect Button function

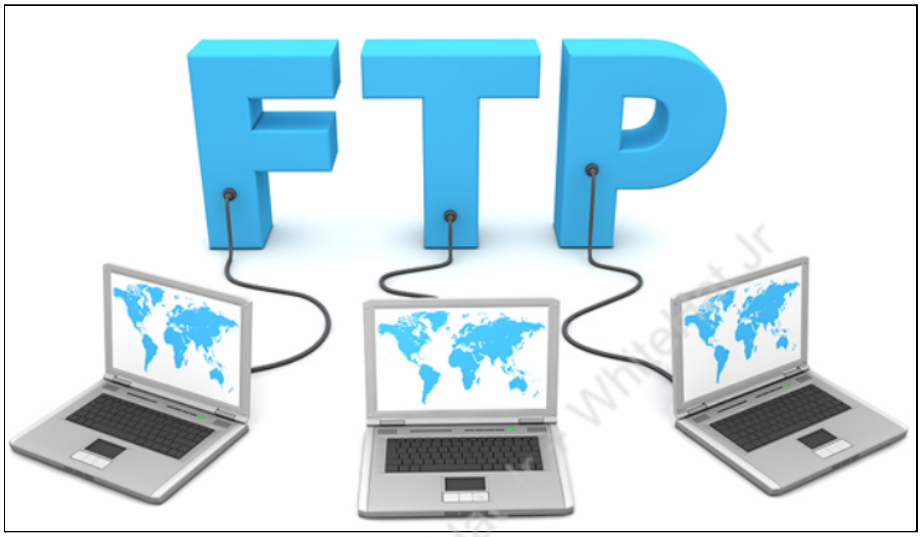
Which CONCEPTS/CODING BLOCKS did we cover today?

- We learned how to make connect button working
- We learned how to make disconnect button working

The KEY CONCEPT

1. What is FTP?

   The File Transfer Protocol is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network



File transfer protocol (FTP) is a set of rules that computers follow for the transferring of files from one system to another over the internet. It may be used by a business to transfer files from one computer system to another, or websites may use FTP to upload or download files from a website's server.

How did we DO the activities?

1. Devices needed to create a FTP
   - Server
   - Client
   - GUI
   - FTP
   - Other functions

2. Create a function **connectWithClient()** for the connect button

```
def connectWithClient():
    global SERVER
    global listbox

    text=listbox.get(ANCHOR)
    list_item = text.split(":")
    msg="connect "+list_item[1]
    SERVER.send(msg.encode('ascii'))
```

3. Call the function at the user-interface in order for Connect to work.

```
connectButton=Button(window,text="Connect",bd=1, font = ("Calibri",10), command = connectWithClient)
connectButton.place(x=282,y=160)
```

4. In order to connect a client at server side **connectClient()** function is used.

```
def connectClient(message, client, client_name):
    global clients

    entered_client_name = message[8:].strip()
    if(entered_client_name in clients):
        if(not clients[client_name]["connected_with"]):
            clients[entered_client_name]["connected_with"] = client_name
            clients[client_name]["connected_with"]  = entered_client_name

            other_client_socket = clients[entered_client_name]["client"]

            greet_message = f"Hello, {entered_client_name} {client_name} connected with you !!!"
            other_client_socket.send(greet_message.encode())

            msg = f"You are successfully connected with {entered_client_name}"
            client.send(msg.encode())
        else:
            other_client_name = clients[client_name]["connected_with"]
            msg = f"You are already connected with {other_client_name}"
            client.send(msg.encode())
```

5. Create a function **handleMessage()** to send messages based on user input

```
def handleMessges(client, message, client_name):
    if(message == 'show list'):
        handleShowList(client)
    elif(message[:7] == 'connect'):
        connectClient(message, client, client_name)
    elif(message[:10] == 'disconnect'):
        pass
```

6. Create a function dis**connectWithClient()** for the disconnect button

```
def disconnectWithClient():
    global SERVER

    text=listbox.get(ANCHOR)
    list_item = text.split(":")
    msg="disconnect "+list_item[1]
    SERVER.send(msg.encode('ascii'))
```

7. Call the function at the user-interface in order for Disconnect to work.

```
disconnectButton=Button(window,text="Disconnect",bd=1, font = ("Calibri",10),command = disconnectWithClient)
disconnectButton.place(x=350,y=160)
```

8. Create a function dis**connectWithClient()** function at the server side

```
def disconnectWithClient(message, client, client_name):
    global clients

    entered_client_name = message[11:].strip()
    if(entered_client_name in clients):
        clients[entered_client_name]["connected_with"] = ""
        clients[client_name]["connected_with"]  = ""

        other_client_socket = clients[entered_client_name]["client"]

        greet_message = f"Hello, {entered_client_name} you are successfully disconnected with {client_name} !!!"
        other_client_socket.send(greet_message.encode())

        msg = f"You are successfully disconnected with {entered_client_name}"
        client.send(msg.encode())
```

9. The disconnect function has been created for both the server and client, so lets call it on the handleMessage() function side as well.

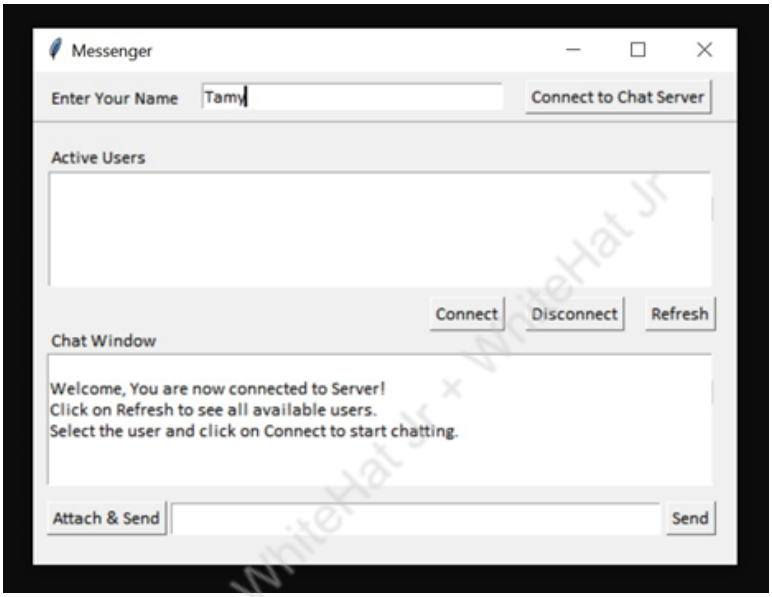
```
def handleMessges(client, message, client_name):
    if(message == 'show list'):
        handleShowList(client)
    elif(message[:7] == 'connect'):
        connectClient(message, client, client_name)
    elif(message[:10] == 'disconnect'):
        disconnectWithClient(message, client, client_name)
```

10. server.py in terminal/cmd looks like -

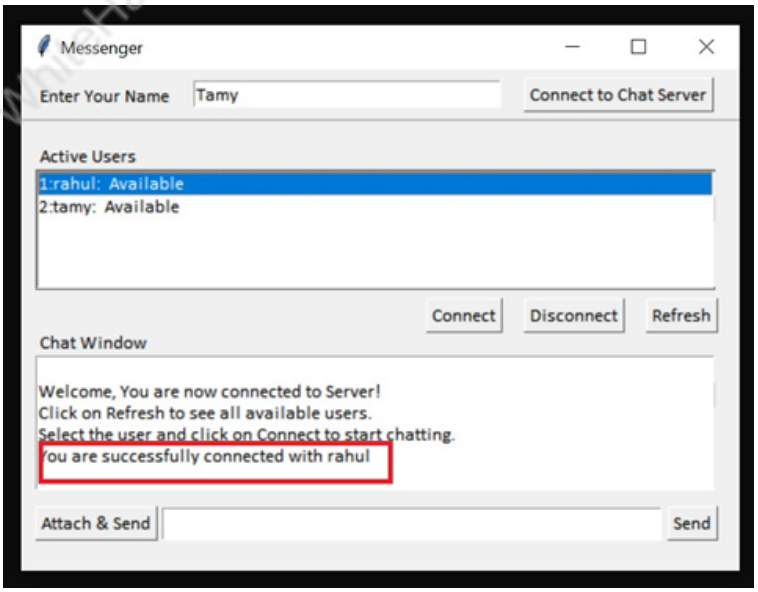
```
                    IP MESSENGER

SERVER IS WAITING FOR INCOMMING CONNECTIONS...
```

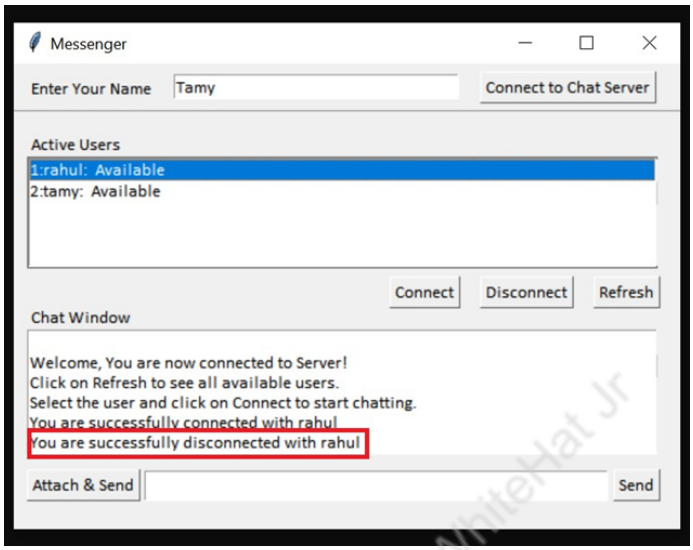
11. client.py in the terminal/cmd looks like -



12. On clicking “Connect” button it will display user list, below window will appear like this



13. On clicking the Disconnect button it will display the user list, below the window will appear like this.



We have completed the second part of the app!

What's NEXT?

In the next class we will _____

EXTEND YOUR KNOWLEDGE

You can learn more about messaging from https://en.wikipedia.org/wiki/Windows_Messenger_service.