

Topic	File Sharing App - 5	
Class Description	Students will be able to learn File sharing desktop application Students will create a folder on the server and then learn how to FTP download the files and save them to their computer's download folder	
Class	C-212	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> • Create folder on the server • Download file through FTP • Save file in Computer's Download Section 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code 	
Class structure	Warm-Up Teacher - led Activity 1 Student - led Activity 1 Wrap-Up	10 mins 10 mins 20 mins 5 mins
WARM UP SESSION - 10mins		
Teacher Action		Student Action

Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?	ESR: Hi, thanks, yes, I am excited about it!
Q&A Session	
Question	Answer
What does chunk mean? A. Data packets B. Data management C. Data format D. None of the above	A
What can we do with list slicing? A. Multiply the list B. Display the list C. Generate new list from existing List D. None of the above	C
TEACHER-LED ACTIVITY - 10mins	
Teacher Initiates Screen Share	
<p style="text-align: center;"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Make path directory • Create functions for grantAccess, declineAccess • Create functions for sending file 	
Teacher Action	Student Action
Okay, so you remember what we did in the last session Great! Any doubts from last session?	ESR Yes!

<p><i>The teacher clarifies doubts (if any)</i></p> <p>How about moving on to the next part?</p> <p>Let's move on to the last part of the FTP application?</p> <p>In previous sessions, we created FTP servers and learned how to browse files from the system</p> <p>Right!</p> <p>In today's session, we will learn how a server creates a shared folder and from a shared folder, it sends the file to your Downloads folder.</p> <p>Are you ready for this?</p> <p>We imported os module in the last session, Remember!</p> <p>What is the purpose of importing that?</p> <p>Right!</p> <p>In our last session, we retrieved the file name from the operating system. After accessing a file, we need to share it, and in order to do this we need to create a folder from where the user can download it.</p> <p>In fact, we do this every time when we upload & download data.</p>	<p>ESR: Yes!</p> <p>ESR Yes!</p> <p>ESR To access file from operating system</p>
<p><i>Teacher download the boilerplate code from Teacher Activity 1</i></p>	<p><i>Student download the code from Student Activity 1</i></p>

<p>Define variables <code>sending_file</code>, <code>downloading_file</code>, and <code>filetodownload</code> and set their values to <code>None</code></p>	
<pre>sending_file = None downloading_file = None filetodownload = None</pre>	
<p>Now, we will create <code>shraed_folder</code> at server side</p> <ul style="list-style-type: none"> • Create a variable <code>is_dir_exist</code> which will store the filesystem path. • Path: An object that represents a filesystem path. A path-like object is either a string or a bytes object that represents a path. • The <code>os.path.isdir ()</code> method in Python checks if the specified path has a <code>shared_files</code> directory. If it doesn't exist, then it makes the directory named <code>shared_files</code>. 	
<pre>is_dir_exists = os.path.isdir('shared_files') if(not is_dir_exists): os.makedirs('shared_files')</pre>	
<p>Look, sending the file and then downloading it requires the user's permission. To do this, we take their permission from the user whether they want to accept or deny the file.</p> <p>To do this we must create functions for <code>grantAccess()</code> and <code>declineAccess()</code>, at the server side, which will be called when downloading the file.</p> <p>Let's name the function <code>grantAccess()</code></p> <ul style="list-style-type: none"> • Pass the arguments <code>client_name</code> 	

- Other_client_name will save the client name and connection information for the person they want to send the file
- In other_client_socket, the socket information of the sender is stored
- Create a variable msg which will store "access granted"
- Other_client_socket will send this information using **send()** in encoded form

```
def grantAccess(client_name):
    global clients

    other_client_name = clients[client_name]["connected_with"]
    other_client_socket = clients[other_client_name]["client"]
    msg = "access granted"
    other_client_socket.send(msg.encode())
```

Now we are going to do the same thing for permission denied

Let's name the function **decline Access()**

- Pass the arguments client_name
- Define global variable clients
- Other_client_name will save the client name and connection information for the person they want to send the file
- In other_client_socket, the socket information of the sender is stored
- Create a variable msg which will store name along with message "decline your request"

- Other_client_socket will send this information using **send()** in encoded form

```
def declineAccess(client_name):
    global clients

    other_client_name = clients[client_name]["connected_with"]
    other_client_socket = clients[other_client_name]["client"]
    msg = f"Oops!!! {client_name} decline your request..."
    other_client_socket.send(msg.encode())
```

The boiler codes end here

Let's send the file now,
Whenever we send a file to another client, it will verify the file_name and size as well as who sent the file and to whom the file is being sent. In addition, it will display the message "Do you want to download?" on text -area.

For that lets create function **handleSendFile()**

- Set up a global variable client
- Use the list client, which will save the client_name and file_name and store the data in the file_name
- Use the list client, which will save the client_name and file_size and store the data in the file_size
- Other_client_name will save the name of the client it is connected to
- Other_client_socket will save the client ,other_client_name
- Using a variable msg, which store a record of client name, file name, along with the message do you want to download?
- Other_client_socket will send this message using

send() in encoded form.

- **time.sleep()** will give delay of one second

```
def handleSendFile(client_name, file_name, file_size):
    global clients

    clients[client_name]["file_name"] = file_name
    clients[client_name]["file_size"] = file_size
    other_client_name = clients[client_name]["connected_with"]
    other_client_socket = clients[other_client_name]["client"]
    msg = f"\n(client_name) want to send (file_name) file with size (file_size) bytes. Do you want to download ? Y/N "
    other_client_socket.send(msg.encode())
    time.sleep(1)
    msgdown=f"Download:{file_name}"
    other_client_socket.send(msgdown.encode())
```

Now, when the user sends the file, it will ask for permission from the client and the permissions will be displayed on the text-area, such as do you want to access the file?

ESR
handleMessage()

What function did we use to display a message on the screen?

Right!

Let's add code in our **handleMessage()** function

In this step, we'll display a message that we'll write in the handleMessage() function to ask permission.

User can either type **"y" or "Y"** to access grant or **"n" or "N"** to declineAccess

If the user types **"y" or "Y"** then the access grant function is called; otherwise, the decline access function is called.

```
def handleMessage(client, message, client_name):  
    if(message == 'show list'):  
        handleShowList(client)  
    elif(message[:7] == 'connect'):  
        handleClientConnection(message, client, client_name)  
    elif(message[:10] == 'disconnect'):  
        disconnectWithClient(message, client, client_name)  
    elif(message[:4] == "send"):  
        file_name = message.split(" ")[1]  
        file_size = int(message.split(" ")[2])  
        handleSendFile(client_name, file_name, file_size)  
        print(client_name+" "+file_name+" "+file_size)  
    elif(message == "y" or message == "yes"):  
        grantAccess(client_name)  
    elif(message == "n" or message == "no"):  
        declineAccess(client_name)  
    else:  
        connected = clients[client_name]["connected_with"]  
        if(connected):  
            sendTextMessage(client_name, message)  
        else:  
            handleErrorMessage(client)
```

We have to modify our **receiveMessage()** function after grant and decline.

- The second condition where the message text from server to client contains "access granted" is to acknowledge that the other client has accepted and downloaded the file transferred from this client. Sender's chat window (text area) will contain this message
- In the third condition, the recipient client declines the file transfer request and the sender is notified accordingly, just like when it was accepted under the second condition. In this case, just the text will be "Oops!" Your request has been declined.

- When the other client has sent a file and the client has the option to accept it by sending a "Y" or "Y" as a message to the server or to decline it by sending a "N" or "N". The message will be identified by the text "Download?". Until now, the file from the sender client has been uploaded to the shared_files folder on the server and is awaiting acceptance or rejection from the recipient client.
- If none of the above condition matches, then the message will be simply appended to the the chat window(textarea) of the recipient client

```
def receiveMessage():
    global SERVER
    global name
    global textarea
    global BUFFER_SIZE
    global downloading_file
    global filetodownload

    while True:
        chunk = SERVER.recv(BUFFER_SIZE)
        try:
            if("tiul" in chunk.decode() and "1.0," not in chunk.decode()):
                letter_list = chunk.decode().split(",")
                listbox.insert(letter_list[0],letter_list[0]+":"+letter_list[1]+": "+letter_list[3]+" "+letter_list[5])
                print(letter_list[0],letter_list[0]+":"+letter_list[1]+": "+letter_list[3]+" "+letter_list[5])

            elif(chunk.decode() == "access granted"):
                labelchat.configure(text="")
                textarea.insert(END,"\n"+chunk.decode('ascii'))
                textarea.see("end")
            elif(chunk.decode() == "Oops!!! client decline your request..."):
                labelchat.configure(text="")
                textarea.insert(END,"\n"+chunk.decode('ascii'))
                textarea.see("end")
            elif("download ?" in chunk.decode()):
                downloading_file = chunk.decode('ascii').split(" ")[4].strip()
                BUFFER_SIZE = int(chunk.decode('ascii').split(" ")[5])
                textarea.insert(END,"\n"+chunk.decode('ascii'))
                textarea.see("end")
                print(chunk.decode('ascii'))

            elif("Download:" in chunk.decode()):
                getfilename = chunk.decode().split(":")
                filetodownload = getfilename[1]

        except:
            textarea.insert(END,"\n"+chunk.decode('ascii'))
            textarea.see("end")
```

Teacher Stops Screen Share

STUDENT-LED ACTIVITY - 20 mins

- Ask the student to press the ESC key to come back to the panel.
- Guide the student to start Screen Share.
- The teacher gets into Full Screen.

ACTIVITY

- Get the path to the shared folder
- Save the file in the Downloads folder

Teacher Action	Student Action
<i>Guide the student to get the boilerplate code from Student Activity 1</i>	<i>Student download the code from Student Activity1</i>
<p>Now that we have browsed the file, we need to store it in a shared folder and allow the user to accept or reject the request.</p> <ul style="list-style-type: none"> • To inform the other client and the server that a file transfer has been initiated we form a message as send file_name,file_size. • By including the send keyword in the message, the server and the other client will be notified that a file transfer has been initiated, and the server will notify the other client either to accept or to deny the request. • By this time, the file would already be on the server. 	

```
def browseFiles():
    global sending_file
    global textarea
    global filePathLabel

    try:
        filename = filedialog.askopenfilename()
        filePathLabel.configure(text=filename)
        HOSTNAME = "127.0.0.1"
        USERNAME = "lftpd"
        PASSWORD = "lftpd"

        ftp_server = ftplib.FTP(HOSTNAME, USERNAME, PASSWORD)
        ftp_server.encoding = "utf-8"
        ftp_server.cwd('shared_files')
        fname=ntpath.basename(filename)
        with open(filename, 'rb') as file:
            ftp_server.storbinary(f"STOR {fname}", file)

        ftp_server.dir()
        ftp_server.quit()

        message=("send "+fname)
        if(message[:4] == "send"):
            print("Please wait ..... \n")
            textarea.insert(END, "\n" + "\nPlease wait ..... \n")
            textarea.see("end")
            sending_file = message[5:]
            file_size = getFileSize("shared_files/"+sending_file)
            final_message = message + " " + str(file_size)
            SERVER.send(final_message.encode())
            textarea.insert(END, "file successfully sent..")

    except FileNotFoundError:
        print("Cancle Button Pressed")
```

Once the server has placed the file in the shared folder, you will need to download the file and save it in the computer's Downloads folder.

sendMessage() function will determine this

- The **sendMessage()** function at client end **SERVER.send()** is the socket connection made with the server.
- While the text sent is inserted in the sender textarea with function **textarea.insert()** and **textarea.see("end")** is to scroll down the textarea automatically.

- There may be a situation when the recipient client must decide whether to accept or deny the download from the server. This will happen when some other client has transferred a file to this client. The denial of the request will be identified by a “N” or “n” in the message which will be taken care of by the server when it receives these keywords in the message by a client.
- But if the user accepts the file download by sending a message as “Y” or “y”, the client also needs to download the file from the server. So, along with sending this positive response to server, an ftp connection will be made with the server and the file waiting at server will be downloaded to the client's Downloads folder

Note:Please do check if there is a Downloads directory in your client PC home path else you may change this directory accordingly in the above code. It would be there on Windows by default.)If its MAC system then in place of forward slash we need to use backward slash for directory

```
def sendMessage():
    global SERVER
    global textarea
    global text_message

    msgtosend= text_message.get()

    SERVER.send(msgtosend.encode('ascii'))
    textarea.insert(END, "\n"+"You>" +msgtosend)
    textarea.see("end")
    text_message.delete(0, 'end')
    if(msgtosend == "y" or msgtosend == "Y"):
        #print("\nPlease wait file is downloading.....")
        textarea.insert(END, "\n"+"Please wait file is downloading.....")
        textarea.see("end")
        HOSTNAME = "127.0.0.1"
        USERNAME = "lftpd"
        PASSWORD = "lftpd"
        home = str(Path.home())
        download_path=home+"/Downloads"
        ftp_server = ftplib.FTP(HOSTNAME, USERNAME, PASSWORD)
        ftp_server.encoding = "utf-8"
        ftp_server.cwd('shared_files')
        fname=filetodownload
        local_filename = os.path.join(download_path, fname)
        file = open(local_filename, 'wb')
        ftp_server.retrbinary('RETR '+ fname, file.write)
        ftp_server.dir()
        file.close()
        ftp_server.quit()
        print("File successfully downloaded to path:"+download_path)
        textarea.insert(END, "\n"+"File successfully downloaded to path:"+download_path)
        textarea.see("end")
```

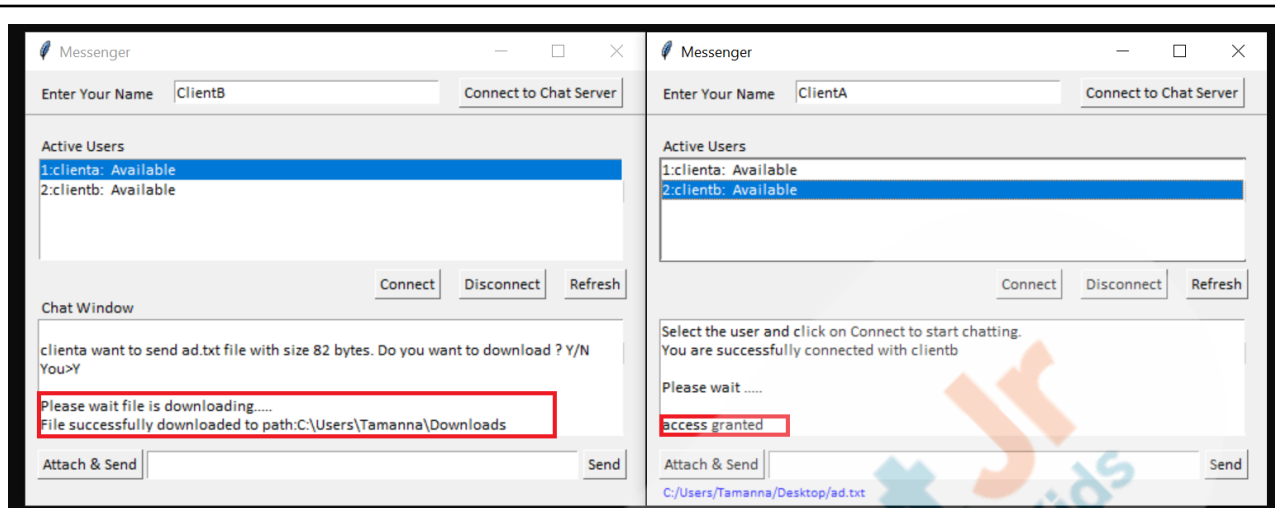
server.py in terminal/cmd looks like -

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

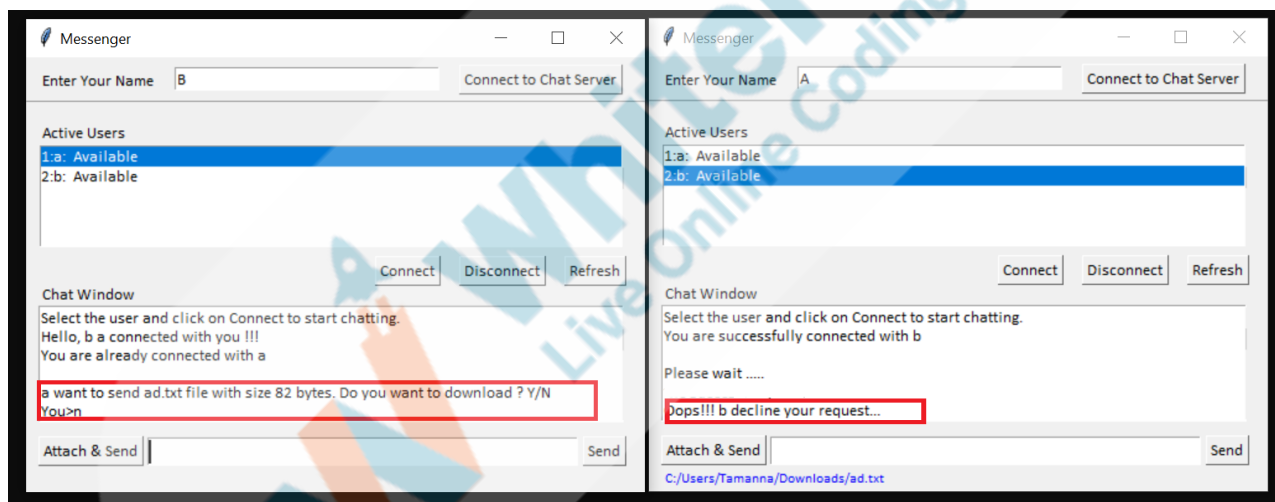
[I 2021-07-14 09:00:03] concurrency model: async
[I 2021-07-14 09:00:03] masquerade (NAT) address: None
[I 2021-07-14 09:00:03] passive ports: None
[I 2021-07-14 09:00:03] >>> starting FTP server on 127.0.0.1:21, pid=38508 <<<
```

client.py in the terminal/cmd looks like -

Click on Attach & send button and access the file from computer system. If Client press "Y" or "y"



After pressing N or n






It's amazing ! We have completed our File sharing Desktop Application.

Teacher Guides Student to Stop Screen Share

WRAP UP SESSION - 5 Mins

Quiz time - Click on in-class quiz

Question	Answer
<p>What does [:3] in Python programming mean if list L = ['a', 'b', 'c', 'd', 'e', 'f']?</p> <p>A. ['a', 'b', 'c', 'd', 'e'] B. ['a', 'b', 'c'] C. ['d', 'e', 'f'] D. ['b', 'c', 'd']</p>	A
<p>What is the purpose of using os.path.isdir() in socket programming?</p> <p>A. To copy path from computer's system B. To remove path from computer's system C. To find if a path exists D. None of the above</p>	A
<p>What options are available for adding color to buttons?</p> <p>A. background B. bgcolor C. bgc D. None of the above</p>	A
End the quiz panel	
<p align="center"><u>FEEDBACK</u></p> <ul style="list-style-type: none"> ● Appreciate the students for their efforts in the class. ● Ask the student to make notes for the reflection journal along with the code they wrote in today's class. 	
Teacher Action	Student Action
<p>You get Hats off for your excellent work!</p> <p>In the next class</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p>

	<div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div>
<p>Project Discussion</p> <p>Goal of the Project:</p> <p>In class, we created a File Sharing application part four. We worked on the user interface buttons. As soon as we browsed the file, we made a shared folder on the server, and then we wrote code to download it in the computer's download folder.</p> <p>Story:</p> <p>Maria enjoys listening to music. With online apps, she gets bored quickly. Her goal is to create her own music desktop application, so whenever she becomes bored, she can listen to a song, download a playlist, or even create a new playlist. Your task is to create a shared folder on the server and then write a function for downloading the file directly in the computer's Downloads folder by accessing path.</p>	
<p>Teacher Clicks</p>	
<p>ADDITIONAL ACTIVITIES</p>	

Additional Activities

Encourage the student to write reflection notes in their reflection journal using markdown.

Use these as guiding questions:

- What happened today?
 - Describe what happened.
 - The code I wrote.
- How did I feel after the class?
- What have I learned about programming and developing games?
- What aspects of the class helped me? What did I find difficult?

The student uses the markdown editor to write her/his reflections in the reflection journal.

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-C212-Teacher-BoilerPlateCode
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/PRO-C212-ReferenceCode
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-C212-StudentBoilerPlateCode