| Topic | File Sharing App - 4 | |
|---|---|---|
| Class Description | **Students will able to learn File sharing desktop application The students will set up FTP servers and learn how to browse files using GUI buttons** | |
| Class | **C-211** | |
| Class time | **45 mins** | |
| Goal | ● Understand about FTP Server<br>● Functionality of the Attach and Send buttons<br>● How to browse file | |
| Resources Required | ● Teacher Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Visual Studio Code<br><br>● Student Resources:<br>○ Laptop with internet connectivity<br>○ Earphones with mic<br>○ Notebook and pen<br>○ Visual Studio Code | |
| Class structure | **Warm-Up**<br>**Teacher - led Activity 1**<br>**Student - led Activity 1**<br>**Wrap-Up** | **10 mins**<br>**10 mins**<br>**20 mins**<br>**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** |
| Why do we choose 127.0.0.01 for the local server?<br><br>A. **127.0.0.1** is reserved for **IP** traffic local to your host.<br>B. Public IP<br>C. Private IP<br>D. None of the above | **A** |
| Why do we need global variables?<br><br>A. The data must be accessed by single function<br>B. The data must be accessed by multiple functions<br>C. Every time, assign a new value<br>D. None of the above | **B** |
| **TEACHER-LED ACTIVITY - 10mins** | |
| **Teacher Initiates Screen Share** | |
| **ACTIVITY**<br><br>• **Import FTP libraries**<br>• **Access the browsing path of a file** | |
| **Teacher Action** | **Student Action** |
| Okay, so you remember what we did in the last session<br><br>Great!<br><br>Any doubts from last session? | **ESR**<br>Functions for buttons |

| | |
|---|---|
| *The teacher clarifies doubts (if any)*<br><br>How about moving on to the next part?<br><br>Let's move on to the fourth part of the FTP application?<br><br>In previous session we added functionalities to **"*Connect to server*", "*Refresh*", "*Connect*" and "*Disconnect*"** buttons<br><br>Right!<br><br>The focus of today's lesson is the ***"send"*** button and ***"attach & send"*** button.<br><br>We must write code for both the client side and the server side to handle all of these buttons.<br><br>As the name implies attach & send, this button is not as simple as the others. A document will be located on the server side on a shared folder that we will create, and it can be shared with another client from there, which will directly download it in the computer's Download folder.<br><br>The second send button is used for sending chat messages<br><br>Do you have any suggestions on how to go about this?<br><br>Like we created the server on our localhost, so likewise, we must create an FTP server<br><br>FTP servers facilitate file transfers across the Internet. When files are sent via FTP, they are either uploaded or downloaded to the FTP server. When you upload files, they are transferred from your computer to the server. When you download files, the files are transferred from the server to your computer. | **ESR:**<br>Yes!<br><br>**ESR:**<br>Yes!<br><br>**ESR:**<br>Varied! |

| | |
|---|---|
| When you want to play music, you need a music player, a music library, and other kinds of software<br>Right! | **ESR:**<br>Yes |
| *Teacher Opens the Teacher Activity 1* | *Student opens the Student Activity 1* |
| FTP servers require new libraries, which we also have to install on our system.<br><br>Are you ready for this?<br>We will learn how to do FTP (file transfer protocol) transfers using ftplib. We will cover both uploading and downloading files from a server.<br><br>For this to work, we must import ftplib and install it in our system<br><br>For that you need to your command prompt and type<br><br>*pip install pyftpdlib* | **ESR**<br>Yes! |

```
Microsoft Windows [Version 10.0.19041.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Tamanna>pip install pyftpdlib
```

| | |
|---|---|
| The ftp library has been installed in our system, and now we must write code to use it | |
| *Teacher download the boilerplate code from Teacher Activity 2* | *Student download the code from Student Activity 2* |

| | |
|---|---|
| Python's ftplib module will be used to upload and download the file. Python comes with an inbuilt module for it.<br>Let's ***import ftplib*** | |
| `import ftplib`<br>`from ftplib import FTP` | |
| As we need to access files from our operating system,the OS module provides access to various functions related to operating systems. These modules provide many functions to interact with filesystems. | |
| `import os` | |
| As we need to access files from our operating system,the OS module provides access to various functions related to operating systems. These modules provide many functions to interact with filesystems.<br>To get path file we need to import ntpath library<br><br>With the Time module, you can perform all time-related tasks. We need both the client and server to import this module. | |
| `import ntpath`<br>`import time` | |
| We need to download a dummy authorization class on the server. An "authorizer" is a class that handles authentications and permissions for the FTP server.<br><br>This class is used by the FTP Handler to verify a user's password, retrieve the user's home directory, check user permissions when a filesystem read/write event occurs, and change the user before accessing a filesystem.<br><br>DummyAuthorizer is the base authorizer, which provides an interface to manage FTP. | |

```
from pyftpdlib.authorizers import DummyAuthorizer
from pyftpdlib.handlers import FTPHandler
from pyftpdlib.servers import FTPServer
```

| | |
|---|---|
| Suppose you want to send a file.<br><br>What will you do?<br><br>*Ask the student*<br><br>You will select the client first!<br><br>Right!<br><br>Suppose you don't select a client, but still want to send the file to them.<br><br>Is it possible ?<br><br>The user will receive an error message if tries the same!<br><br>We will create function ***handleErrorMessage()*** and pass the argument client<br><br>Create variable message which will store “error message” Then, client.send will send this message in encoded form | **ESR**<br>Varied!<br><br>**ESR**<br>No! |

```
def handleErrorMessage(client):
    message = '''
    You need to connect with one of the client first before sending any message.
    Click on Refresh to see all available users.'''
    client.send(message.encode())
```

| | |
|---|---|
| Whenever we wish to read or write a file from the system, we first need to open it.<br><br>Python has a built-in ***open()*** function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly. | |

The ***"rb"*** symbol represents binary data read
Chunk variable will store file data using ***read()*** method
In addition, it will return the length of a chunk.

*The boiler code ends here.*

```
def getFileSize(file_name):
    with open(file_name, "rb") as file:
        chunk = file.read()
        return len(chunk)
```

*The teacher will start writing code from here.*

Next task is to write code for the send button.

What will happen when we click the send button?

Right!

In addition to allowing users to chat, send buttons also inform them whether they want to download a particular file

In this case, another function will be developed at client side so that messages will be sent

Let's start with function ***sendMessage()***

- Declare global variable SERVER, text-area, text_message.
- Create a variable msgtosend which will store the message of the user, and it will use the ***get()*** method to receive the message from the text area.
- server.***send()*** will send this message in encoded form.

**ESR**
It will send the messages

```
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')
```

So to get our send button working, we will need to call this function at the interface design.

```
send=Button(window,text="Send",bd=1, font = ("Calibri",10), command = sendMessage)
send.place(x=450,y=305)
```

A function must also be implemented at the server side so that we can handle client **sendMessage()** requests.

- Create function name ***sendTextMessage()*** where will pass two arguments client_name and message.
- other_client_name is used to fetch the recipient client name to which the sender of the message is connected.
- Other_client_socket is the socket connection of the recipient client with the server. final_message which is to be sent to the recipient is formed by including the sender name as prefix and finally the message is sent from server to recipient client in encoded form.

```
def sendTextMessage(client_name, message):
    global clients

    other_client_name = clients[client_name]["connected_with"]
    other_client_socket = clients[other_client_name]["client"]
    final_message = client_name + " > " + message
    other_client_socket.send(final_message.encode())
```

Our ***handleMessage()*** function should also be enhanced since we are going to receive the message from the user

Our handle message was earlier only applicable to the connect, disconnect and refresh buttons, now we must also handle error messages and the send message also:

- For the else condition, we will use if-else, in which the if condition will display the connected clients' information, and if no clients are selected, an error will be displayed.
- Earlier, it looked like this:

```
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)
```

Now, it is like this:

```
def handleMessges(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)
    else:
        connected = clients[client_name]["connected_with"]
        if(connected):
            sendTextMessage(client_name, message)
        else:
            handleErrorMessage(client)
```

**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**

- **FTP Server Setup**
- **Write functionality for Attach & Send Button**

| Teacher Action | Student Action |
|---|---|
| Since we've written the code for the Send button, it's time for the Attach & Send button<br><br>*Guide the student to get the boilerplate code from Student Activity 2* | *Student clones the code from Student Activity2* |
| FTP services should be installed for the configuration of the FTP server. | |

Let's set up FTP Server

Make a function ***ftp()***

- Declared IP_ADDRESS
- Create a dummy authorizer to manage users where it will authenticate username, password.
- Initialize class for FTP handling i.e FTPHandler
- Instantiate FTP server class and listen to IP_Address and Port Number.
- setup_thread to be used in ftp function instead of ***Setup()*** Function.
- Ftp server needs to run in a thread
- Create ftp_thread for handling ftp server client requests.

```
def ftp():
    global IP_ADDRESS

    authorizer = DummyAuthorizer()
    authorizer.add_user("lftpd","lftpd",".",perm="elradfmw")

    handler = FTPHandler
    handler.authorizer = authorizer

    ftp_server = FTPServer((IP_ADDRESS,21),handler)
    ftp_server.serve_forever()

setup_thread = Thread(target=setup)
setup_thread.start()


ftp_thread = Thread(target=ftp)
ftp_thread.start()
```

| | |
|---|---|
| Now that the server is set up, the time has come for the client side<br><br>Making functions has become one of our specialties,<br><br>So can we make another function<br><br>Basically, we have to access the file from our system, which means we have to browse the file and then need to print the path on the filepathLabel to make it work according to our function.<br><br>Let's make the function ***browseFiles():***<br><br>● Declare variable global textarea, filepathLabel<br><br>● To access a file from our system, we need to take a file from our system for that we will use filedialog module<br><br>● ***askopenfilename()*** is method to access the file<br><br>● We will configure the path inside filePathLabel<br><br>● A HOSTNAME, a USERNAME and a PASSWORD must be declared as part of the authentication process<br><br>● Our FTP_server requires a HOSTNAME, a USERNAME, and a PASSWORD, which we have also specified on the server.<br><br>● The server will encode it using UTF-8<br><br>● Meanwhile, server will create a folder named sharing files, where accessed files will be saved<br><br>● NTpath is used to access a path for a file that is stored in fname<br><br>● Open method is used to read files, and rb is used to | **ESR**<br><br>**Yes!** |

read binary data

- storbinary() initiates the transfer of a binary file from an FTP client to an FTP server with the FTP command STOR.
- Ftp_server will create a directory for the accessed path using ***dir()*** and save a list of the current directory
- After making directory it will quit using ***quit()***

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

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

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

To make our Attach & Send button work correctly, we must call this function from the User Interface

```
attach=Button(window,text="Attach & Send",bd=1, font = ("Calibri",10), command = browseFiles)
attach.place(x=10,y=305)
```

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

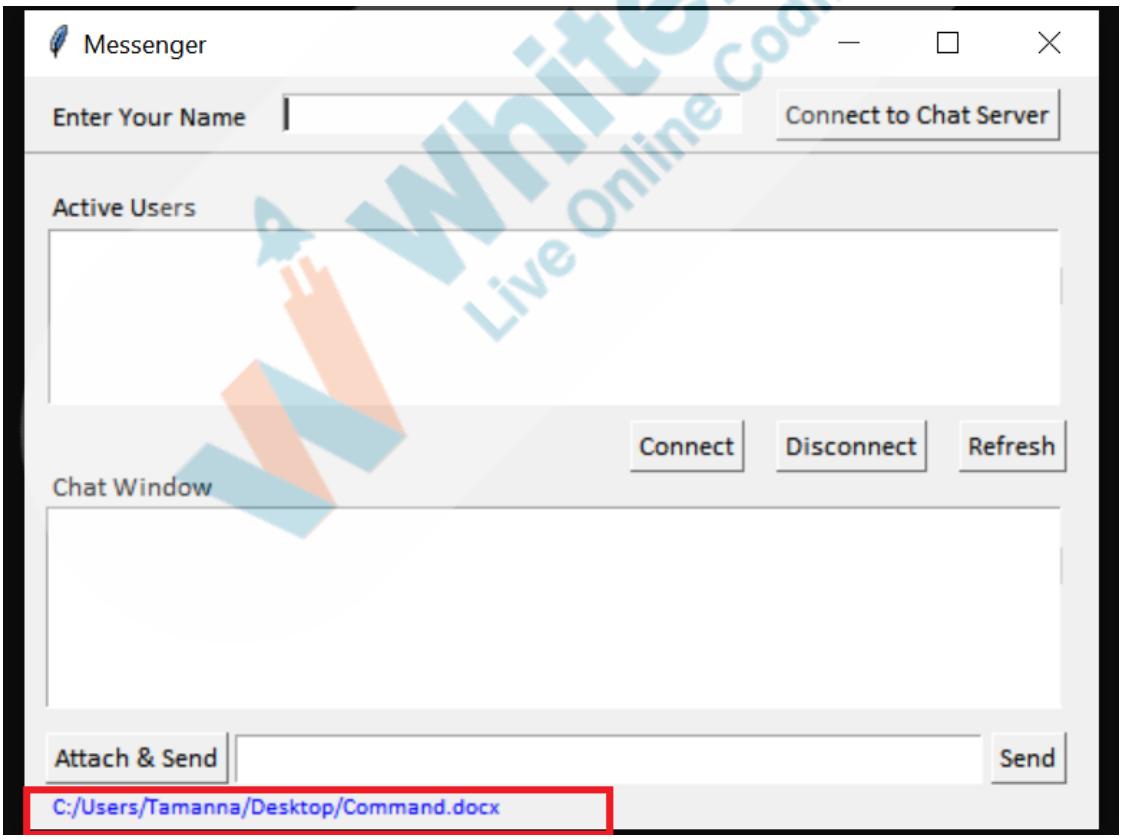
```
                                IP MESSENGER

                        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



| | |
|---|---|
| It's amazing! Our FTP server is created, and we have already written code for browsing the file. Next, we will learn how to download the file. | |
| **Teacher Guides Student to Stop Screen Share** | |
| **WRAP UP SESSION - 5 Mins** | |
| **Quiz time - Click on in-class quiz** | |
| **Question** | **Answer** |
| What is the purpose of our dir() method?<br><br>A. Save a list of the current directory<br>B. Show a list of directories<br>C. List the directories<br>D. None of the above | **A** |
| What is the purpose of the quit() method?<br><br>A. Ends the user's session<br>B. Start the user's session<br>C. Display the user session<br>D. None of the above | **A** |
| What is the purpose of using the OS module?<br><br>A. Display the operating system<br>B. Path to operating system<br>C. Make your operating system<br>D. To access a file from an operating system | **D** |
| **End the quiz panel** | |
| **FEEDBACK**<br>• **Appreciate the students for their efforts in the class.**<br>• **Ask the student to make notes for the reflection journal along with the code** | |

| they wrote in today's class. | |
|---|---|
| **Teacher Action** | **Student Action** |
| You get Hats off for your excellent work!<br><br>In the next class | *Make sure you have given at least 2 Hats Off during the class for:*<br>  |
| **Project Discussion**<br><br>**Goal of the Project:**<br><br>We created a File Sharing application part three. During class we created an FTP server for accessing the files from our computers.We have written functions for attach and send button and have written the code for browsing files.<br><br>**Story:**<br><br>Maria enjoys listening to music. She gets bored with youtube and other apps. She wishes to create her own music desktop app, so whenever she becomes bored, she can click on her application and listen to a song, download a playlist, or even make a new playlist. Your task is to import necessary FTP modules, create FTP Server, and write a function for an upload button function. | |

| Teacher Clicks | ✖ End Class |
|---|---|

| ADDITIONAL ACTIVITIES | |
|---|---|
| **Additional Activities**<br>*Encourage the student to write reflection notes in their reflection journal using markdown.*<br><br>Use these as guiding questions:<br>● What happened today?<br>○ Describe what happened.<br>○ The code I wrote.<br>● How did I feel after the class?<br>● What have I learned about programming and developing games?<br>● 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 Activity1 | FTP Servers | https://en.wikipedia.org/wiki/File_Transfer_Protocol |
| Teacher Activity 2 | Boilerplate Code | https://github.com/pro-whitehatjr/PRO-C211-Teacher-Boilerplate |
| Teacher Activity 3 | Reference Code | https://github.com/pro-whitehatjr/PRO-C211_Reference Code |

| | | |
|---|---|---|
| Student Activity 1 | FTP Servers | https://en.wikipedia.org/wiki/File_Transfer_Protocol |
| Student Activity 2 | Boilerplate Code | https://github.com/pro-whitehatjr/PRO-C211-Student-BoilerplateCode |