**SHARED FILE TRANSFER NETWORK**

File transfer implementing multi-threading application, let’s clients upload or download files from a central server. Additionally, clients can create or connect to temporary rooms and send messages and files to other clients. The server application will authenticate the users through password validation when a user tries to login. All message passing happens using TCP via server and file transfers happen via UDP protocol for better performance.

**Features**

* **Client Authentication:** A unique username is required of each client while logging in, that enforces password-based authentication for access.
* **List**: The client can list all the files from the server.
* **Upload**: The client can upload files to the server.
* **Download**: The client can download the files from the server.
* **Delete:** The authorized client can delete the files from the server. When an authorized user is deleting a file, other users will not be able to perform any other action on the file.
* **File Access:** Users are classified into three access groups where authorized users can upload, download and delete the files, the other group users have access to only upload and download the files and some users have only access to download the files.
* **Multithreading:** With the help of multithreading, the server can handle multiple clients concurrently. The server assigns each client a thread to handle working for that client.
* **Logs:** The server maintains an audit report which consists of login details, timestamp and client’s IP address of connected clients.
* **Temporary file transfer:** The server provides the list of active users to the client, and they can share files with other clients in a temporary room and these files will no longer be available once all the users have left the room.

**Getting Started**

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes.

**Prerequisites**

You need to have Python 3.1 to run the application.

**Installing**

####Using Github

git clone <https://github.com/Prudhvi-Chekka/Shared-File-Transfer-Network-Group>

**Understanding how the code works:**

Here is a block diagram showing how clients connect to a specific chatroom on a server in the application:

Following is a flow chart that depicts working of a server:

Text, letter

Description automatically generated

**How to run the code**

**Start**

On the device we wish to run the server on, open a terminal window and obtain the IP address.

Now, on the device we need to start the Server, the server can be run with command line arguments. The options made available to us are:-

1. Example to start server:-

cd .\server\

To access all the files in client folder

python .\server.py

To execute server python file

Now “Server is listening at 50000” is displayed.

1. Next, on the device we need to start the client, run the client can be run with command line arguments. The options made available to us are:-

Example to start client:-

cd .\client\

To access all the files in client folder

python .\client.py

To execute client python file

Now “Connected to 192.168.1.111 at port 50000” is displayed.

1. Next in the client window Signup\Login option is displayed   
   If we select Signup next it will display to enter Username and Password.
2. Now it will prompt to request for admin access.

If the client selects Yes, Server will be asked for approval.

Please Send Active ACK will be displayed

Now it will display client needs admin access. Please approve.

Once its approved client is given Admin access, then they have access to delete files.

1. Client is now prompted to Login by entering Username and password.

Next displays “Welcome Username”,

What do you want from the server.

1.List Files.

2.Download a File.

3.Upload a File.

4.Delete a File.

5.Create a chatroom.

6.Join a chatroom.

1. Now clients can select an option from above list

If client selects “1” – List Files  
List of all the files will be displayed

If client selects “2” – Download a file

Firstly, List of all files will be displayed,

Next “Enter the file name” will be displayed and the entered file will be downloaded.

If client selects “3”- Upload a file

Enter the file path will be displayed and once client enter the file path along with file name, that file will be uploaded.

If client selects “4” – Delete a file

Firstly, List of all files will be displayed,

Next “Enter the file name” will be displayed and the entered file will be deleted if you have admin access along with the list of files. If you do not have admin access, then it will display “You don't have admin access to delete the file”.

If client selects “5” – Create a chatroom

If you select this option it will display “OK|Specify a chatroom name to create.”

After you enter a name for your chatroom it will ask “OK|Chatroom created. Do you want to password protect the room? [yes/no]”

Next you are all set to pass messages and files in the chat room.

If client selects “6” – Join a chatroom

It will display “OK|Here is a list of chatrooms you can join.” And gives the list of chatroom available. Now the client can select any chatroom to enter by typing its name from the list. Next it displays “Welcome to chatroom. You are all set to pass messages”.

Here you can chat, send messages to all member by using @all, to server using @server, to one particular client using @clients username.

You can upload, download and delete files in the chatroom by entering

@server|Upload, @server|download, @server|delete.

1. While entering the above option 1 – 6, the client can continue entering the next option after finishing one.
2. While entering options 1- 6, the client can enter Help if they want the list stating 1 – 6 commands to be displayed.

**Chat Options**

Chat commands made available to the user:-

| **Command** | **Description** |
| --- | --- |
| `@username | chat` |
| `@all | chat` |
| `@server | chat` |
| `@server | get\_peers` |
| `@server | exit` |
| @server | list |
| @server | Download |
| @server | Upload |

Commands made available to the server:-

| **Command** | **Description** |
| --- | --- |
| `@username | chat` |
| `@all | chat` |
| `@server | chat` |
| exit | Kill all client connections and exit application gracefully |

**How to interpret the results**

1. Chat : Check for chat messages in terminal (Format: #sender|message from user)
2. File Transfer: Check for file existance inside [folder](https://github.com/danisfermi/chat-based-file-transfer/blob/master/code/code/folder) or click on list and check it.
3. users.db: Check the database output and observe the timestamp, port number and username of the user.