**MANUAL**

**Content:**

1. Registering a PEER.

2. Searching for a particular file from the indexing server.

3. Downloading the file from the peer of choice.

4. Deregistering the file when the user deletes the file in the directory.

5. Concurrently searching the file and downloading the file .

**1.Registering the PEER**

First, we need start the server(run the Server.java) and then when we run the Client.java(Add new console window and pin it) then it asks for the port number and directory and once the user gives both two information then the peer is registered to the server(Indexing Server).Then user will be prompted whether he wants to register a file or just search for a file in the network and download. Let us first discuss about registering a file.

For example, let’s consider the screenshot below :

Graphical user interface, text, application, email

Description automatically generated

In the above screenshot when the peer1 choose to register a file to the server then it asked for peer id and after the user provided peer id as 1 then it asked for number of files the peer1 wanted to register and user entered as 3 and it allowed the peer to register 3 files namely 1.txt,100.txt and 9.txt to register to the server.

All these three files are registered on server, and it would print the message on console as shown below:Graphical user interface, text, application, email

Description automatically generated

For multiple peers we just need to run Client.java multiple times and open new console for every new peer and then pin it.

**2.Searching for a particular file from the indexing server**

If you are not willing to register any file, then you can directly choose option 2 and start searching for a particular file and downloading it. We can also search and download a file after registering the files also.

Let’s us consider the below screen shot:

**Graphical user interface, text, application, email

Description automatically generated**

In the above screen shot as soon as you choose option 2 it will prompt you to enter your peer id and after that it will as you for the filename which you are looking to search and download. Once you enter the filename , then the peer sends the request to the server and then the server returns the list of peers having that file.

**3.Downloading the file from the peer of choice**

In the previous screenshot the server sends back the list of peers having the file then it asks the user to choose the peer from whom the user is willing to download the file and enter the respective peer ID.

Let’s consider the below screenshot:

Graphical user interface, text, application, email

Description automatically generated

Here the PEER1(user) choose Peer ID 3 and as shown the file is downloaded from PEER3 to PEER1.

**4.Deregistering the file when the user deletes the file in the directory**

After you search for a file or download for a file or after finishing registering the files then if the user deletes any file from the directory of any peer and then watch service will detect that and it will immediately call the de-register function of the server which will in turn remove that peer id for that file and updates its list.

For example, let’s discuss the below screenshot:

Graphical user interface, text, application, email

Description automatically generated

Here in the above screenshot the user deletes the 100.txt file from the PEER1. So, here in the peer side the message is displayed as “File affected 100.txt” and then in the server side the 100.txt file is de-registered from the PEER1 as shown in the below screenshot.

Graphical user interface, text, application, email

Description automatically generated

**5.Concurrently searching the file and downloading the file**

Please run Server.java and then run Client.java only once and register as PEER4 and register the 100.txt(100 kb file) with source directory as /Users/sagarshekhargoudapatil/Desktop/PEER4. Once this is completed then run Concurrent.java. Here in we are running 3 peers i.e. PEER1,PEER2,PEER3 and all the three peers are concurrently searching for 100.txt file from the server and then server sends back the list and all the three peers choose the same peer 4 to download the 100.txt file and here we are trying to measure the time taken for searching and downloading the file from PEER4 to all the three peers.

Below is the screenshot for reference:

Graphical user interface, text, application, email

Description automatically generated

**Considerations**:

1. Server is always running on port 6025(I Have hard coded it for now as we are using only one server)
2. The de-registering of the files automatically works either after you complete registering the file or downloading the file or else if you choose option 0 in the beginning or option 0 after registering the file.