

SDE Assignment-1 Report

Bikash Dutta (D22CS051)

System Design:

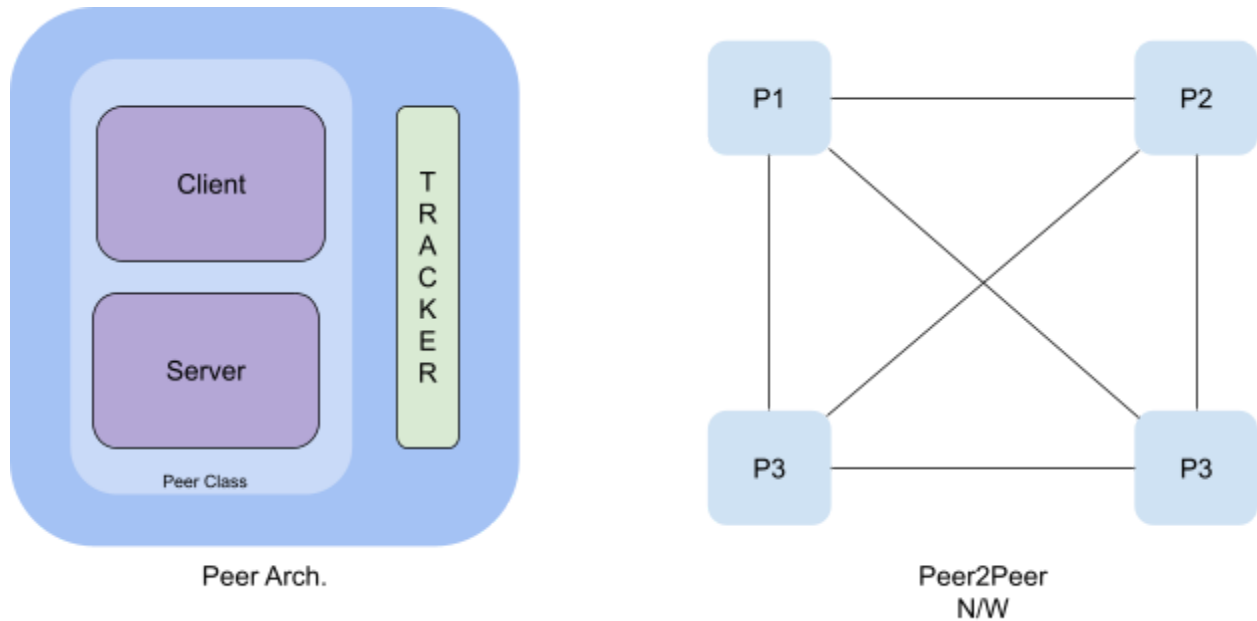


Fig: Peer Architecture & Network Topology for P2P system

- Architecture: Within a peer-to-peer (P2P) system, the absence of a central server is mentioned. In contrast, each peer (device) possesses equivalent status and has the capability to function as both a client and a server. In a mesh network, peers establish direct connections with one another, enabling the exchange of files without dependence on a central body.
- Peer connectivity is established through the exchange of IP addresses and port numbers among peers. An instance of Peer A can establish a connection with an instance of Peer B by utilizing the IP address and port associated with Peer B. This mode of communication facilitates the sharing of data between individuals of equal status.
- Peer discovery is a crucial aspect. Upon joining the network, a peer undergoes a registration process with its tracker server, which is responsible for maintaining a comprehensive record of all registered peers in the network. The purpose of this tracker is to function as a directory service that aids in the process of peer discovery. When an individual within a network desires to locate additional peers, they initiate a query to the tracker to ascertain the availability of these peers.

- The security measures implemented in the system are based on the principle of restricting participation to registered peers. Consequently, individuals within the network are required to register with the tracker in order to become members. Nevertheless, the implementation of encryption and authentication procedures can significantly bolster security in practical scenarios.

Implementation:

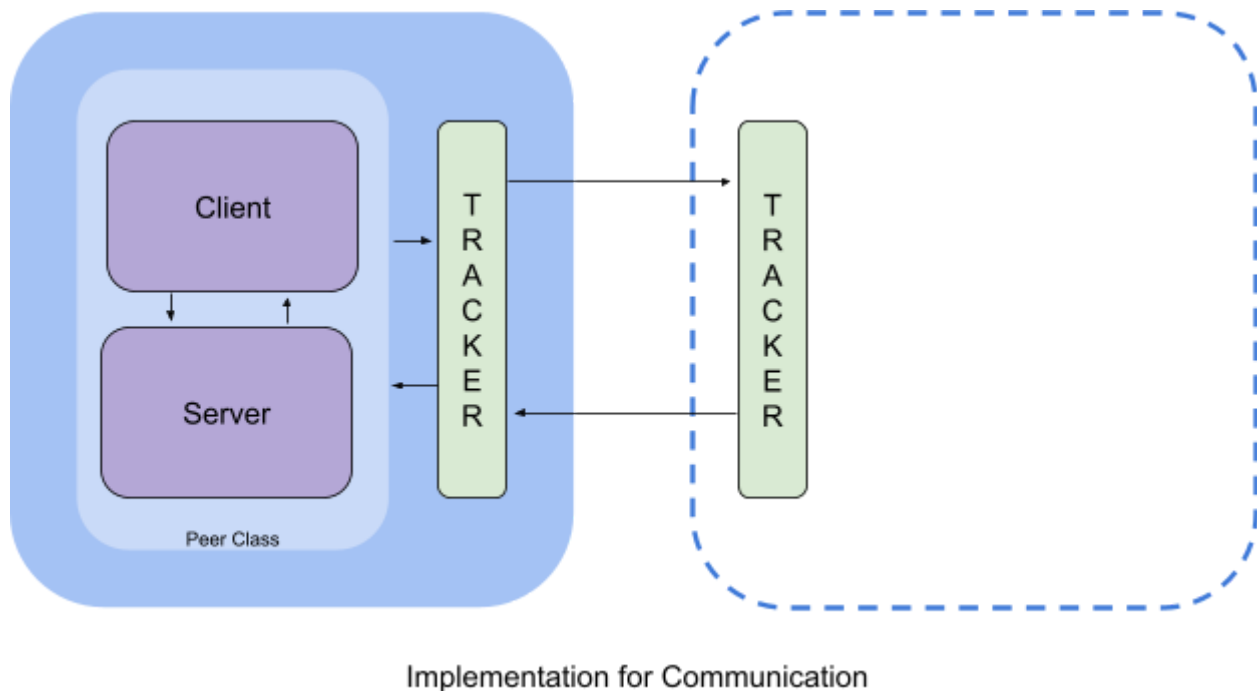


Fig: Communication b/w tracker for exchanging IPs

- Implementation is done in Python with the help of Socket Programming, multi-threading, FastAPI & PyQT5.
- To become a member of the network, individuals can participate by furnishing their IP address, port number, and a distinct identifier(name). The registration process serves the purpose of ensuring that only individuals who have been granted authorization are able to become members of the network. Prevention of unauthorized access is ensured.
- File sharing is a process whereby individuals can exchange files by employing the "SHARE" function, which is afterwards followed by the specific filename. As an illustration, in the scenario where Peer A intends to distribute a file to Peer B, it initiates the transmission by sending a message labeled "SHARE: filename" to Peer B(handled by code itself). Subsequently, the file is transmitted directly from Peer A to Peer B through a socket connection.

- Error handling is an integral component of the system, as it is designed to effectively manage and resolve errors that may arise in various scenarios. An illustrative instance is when the software adeptly manages problems associated with network operations, such as instances of connection failures and timeouts. Furthermore, it has the capability to detect the user's unresponsive peers.
- User-Friendly Interactions: The utilization of a graphical user interface (GUI) facilitates an intuitive means for users to engage with the system. Individuals have the ability to exchange files, conduct searches for files, and retrieve them. The graphical user interface (GUI) provides instantaneous feedback and updates on the system's current condition, enhancing its user-friendliness.

Efficiency and Scalability:

- System Performance: The system has been specifically engineered to effectively manage an increasing quantity of peers. The system incorporates methods for regular monitoring of unresponsive peers and subsequent removal from the peer list, so ensuring the maintenance of a manageable list of peers.
- File Discovery: In order to achieve efficient file discovery, the system employs a tracker server for the purpose of peer finding. When a peer initiates a search for files, it sends a query to the tracker, which in turn provides a response with a comprehensive list of peers that have the requested files available. The scalability of this approach is noteworthy; nonetheless, it is crucial to take into account the capacity and redundancy of the tracker when dealing with networks of significant size.
- The process of downloading files is considered efficient due to its direct peer-to-peer nature. This concurrent downloading process serves to improve both the speed and reliability of the download.

User Interface:

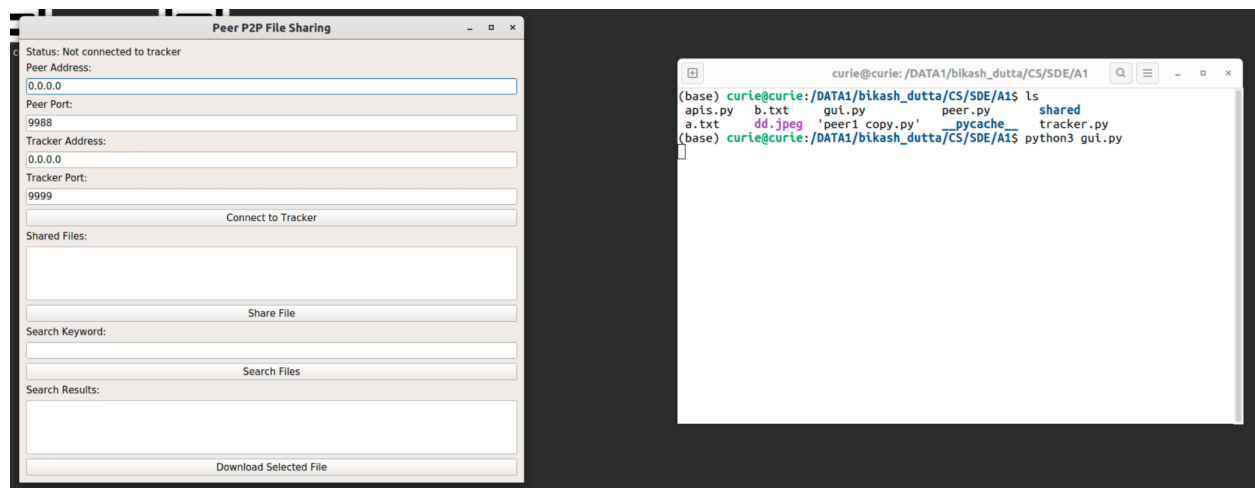
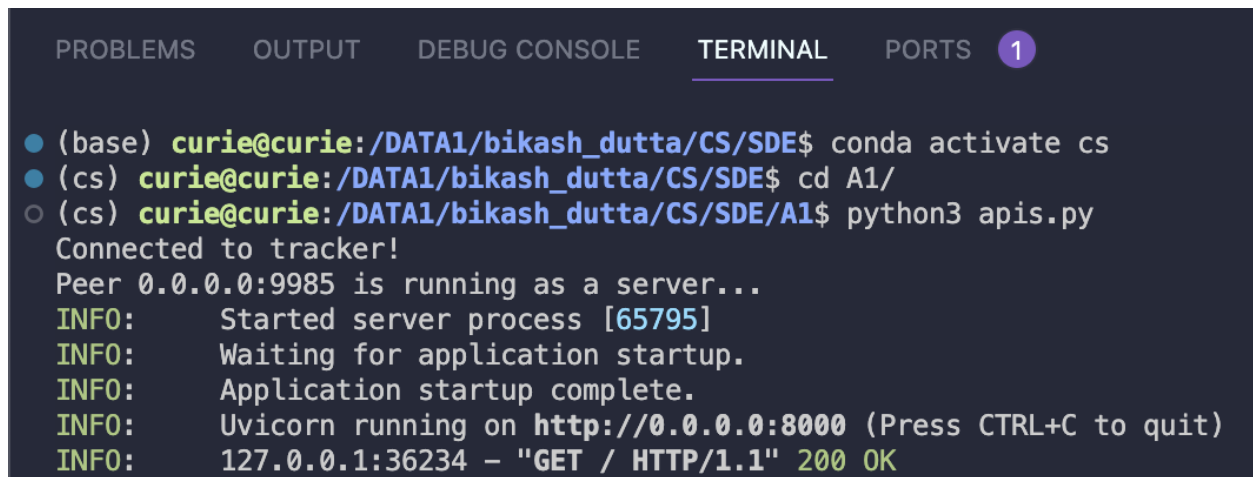


Fig: GUI Interface & Running GUI



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
● (base) curie@curie:/DATA1/bikash_dutta/CS/SDE$ conda activate cs
● (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE$ cd A1/
○ (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE/A1$ python3 apis.py
Connected to tracker!
Peer 0.0.0.0:9985 is running as a server...
INFO: Started server process [65795]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
INFO: 127.0.0.1:36234 - "GET / HTTP/1.1" 200 OK
```

Fig: Running APIs

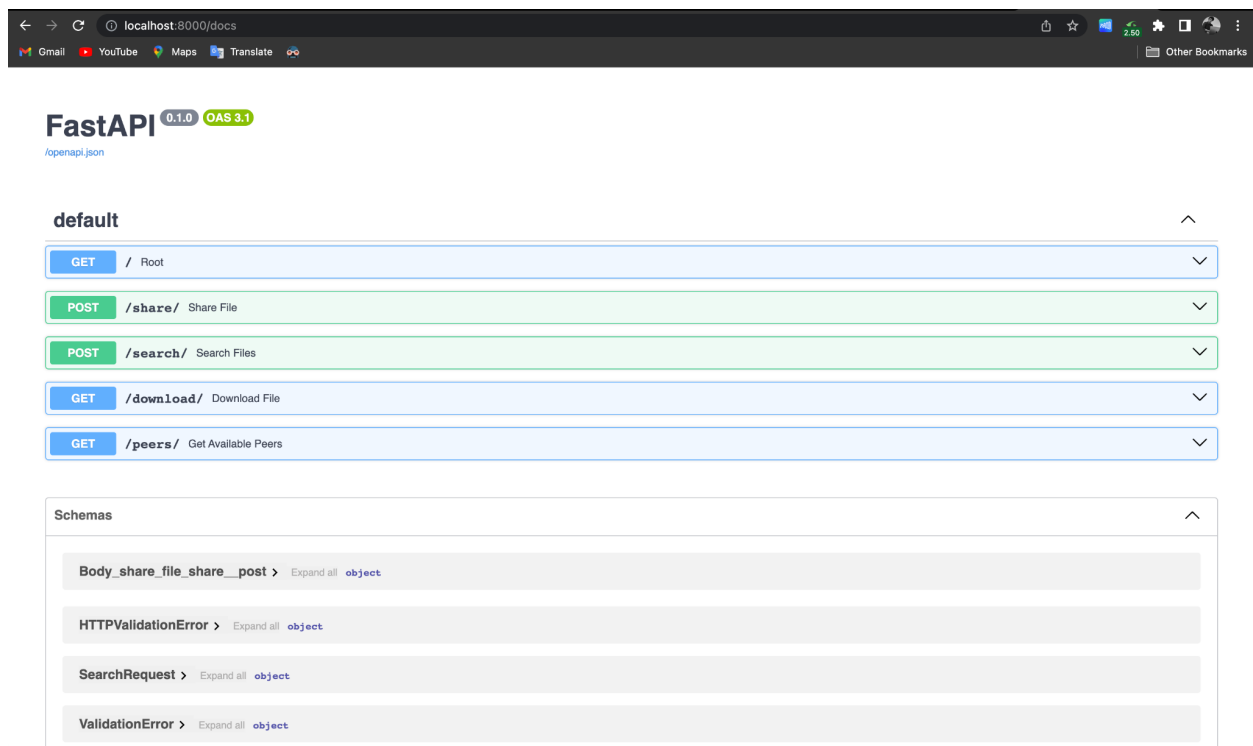
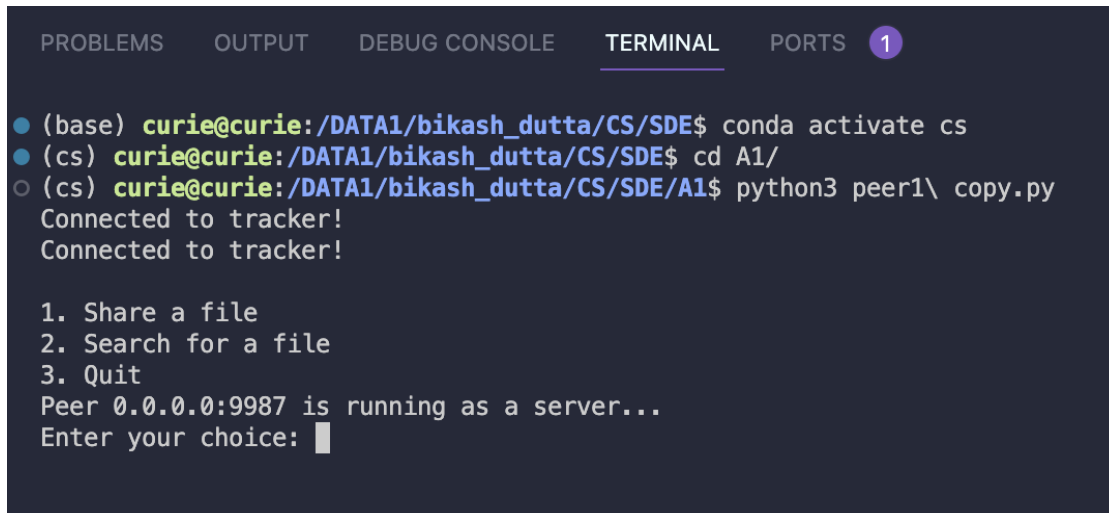


Fig: APIs Endpoints and Documentation



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1
• (base) curie@curie:/DATA1/bikash_dutta/CS/SDE$ conda activate cs
• (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE$ cd A1/
○ (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE/A1$ python3 peer1\ copy.py
Connected to tracker!
Connected to tracker!

1. Share a file
2. Search for a file
3. Quit
Peer 0.0.0.0:9987 is running as a server...
Enter your choice: █
```

Fig: CLI interface

- The graphical user interface (GUI) provides a user-friendly means of interacting with the peer-to-peer (P2P) system. Users have the ability to engage in many operations, such as effortlessly exchanging files, conducting file searches, and downloading files. The graphical user interface (GUI) effectively communicates feedback regarding the status of actions and presents shared files and search results, thereby boosting the overall user experience.
- APIs have also been made available such that extension to other applications is at ease. Ex: GUI is a System Application, and APIs can be used to develop Mobile/Web Apps.
- CLI is Also available for system administration if needed, to run on Server systems that have no GUIs.

Follow-up Question:

- The decentralized peer-to-peer (P2P) architecture offers several advantages.
 - Scalability: The decentralized nature of peer-to-peer (P2P) networks facilitates easy scalability, as the incorporation of new peers does not impose any additional burden on a central server. Every individual within the network makes a contribution towards enhancing its overall capacity.
 - Data redundancy and robustness are intrinsic features of decentralized networks due to the distribution of files among different peers. In the event that a peer becomes disconnected from a network, it is still possible for other peers to offer the same files.
 - The decentralized architecture of a system enhances user privacy by restricting access to user data from centralized servers.

- The challenges associated with decentralized peer-to-peer (P2P) architecture are manifold.
 - The task of guaranteeing secure and confidential communication inside a decentralized network is a considerable challenge. In order to safeguard data, it is imperative for peers to employ various measures such as encryption and authentication.
 - The task of maintenance involves the management of peer lists and the maintenance of data consistency, which can present inherent complexities. Regular monitoring of unresponsive peers is crucial.
 - Efficiency may be compromised in extensive networks as a result of the substantial number of participants, leading to decreased effectiveness in locating specific files or peers. When designing a system, it is important to take into account scalability methods such as load balancing.

The hybrid architecture is characterized by the integration of components from both centralized and decentralized systems. This technology has the potential to enhance security and provide greater control, all while ensuring scalability and redundancy. Nevertheless, the introduction of complexity and the possibility of single points of failure are noteworthy concerns.

Running Figures:

- **Sharing File:**
 - CLI

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  1
• (base) curie@curie:/DATA1/bikash_dutta/CS/SDE$ conda activate cs
• (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE$ cd A1/
◦ (cs) curie@curie:/DATA1/bikash_dutta/CS/SDE/A1$ python3 peer1\ copy.py
Connected to tracker!
Connected to tracker!

1. Share a file
2. Search for a file
3. Quit
Peer 0.0.0.0:9987 is running as a server...
Enter your choice: 1
Enter the name of the file to share: a.txt
a.txt has been shared.

1. Share a file
2. Search for a file
3. Quit
Enter your choice: █

```


API

localhost:8000/docs#/default/share_file_share__post

POST /share/ Share File

Parameters

No parameters

Request body ^{required}

multipart/form-data

files ^{required}
array

Choose file forming.webp

Add string item

Execute

Responses

Code	Description	Links
200	Successful Response	No links
Media type: application/json		
Controls Accept header.		
Example Value Schema		
"string"		
422	Validation Error	No links
Media type: application/json		
Example Value Schema		

localhost:8000/docs#/default/share_file_share__post

Curl

```
curl -X 'POST' \
  'http://localhost:8000/share/' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'files=@forming.webp;type=image/webp'
```

Request URL

http://localhost:8000/share/

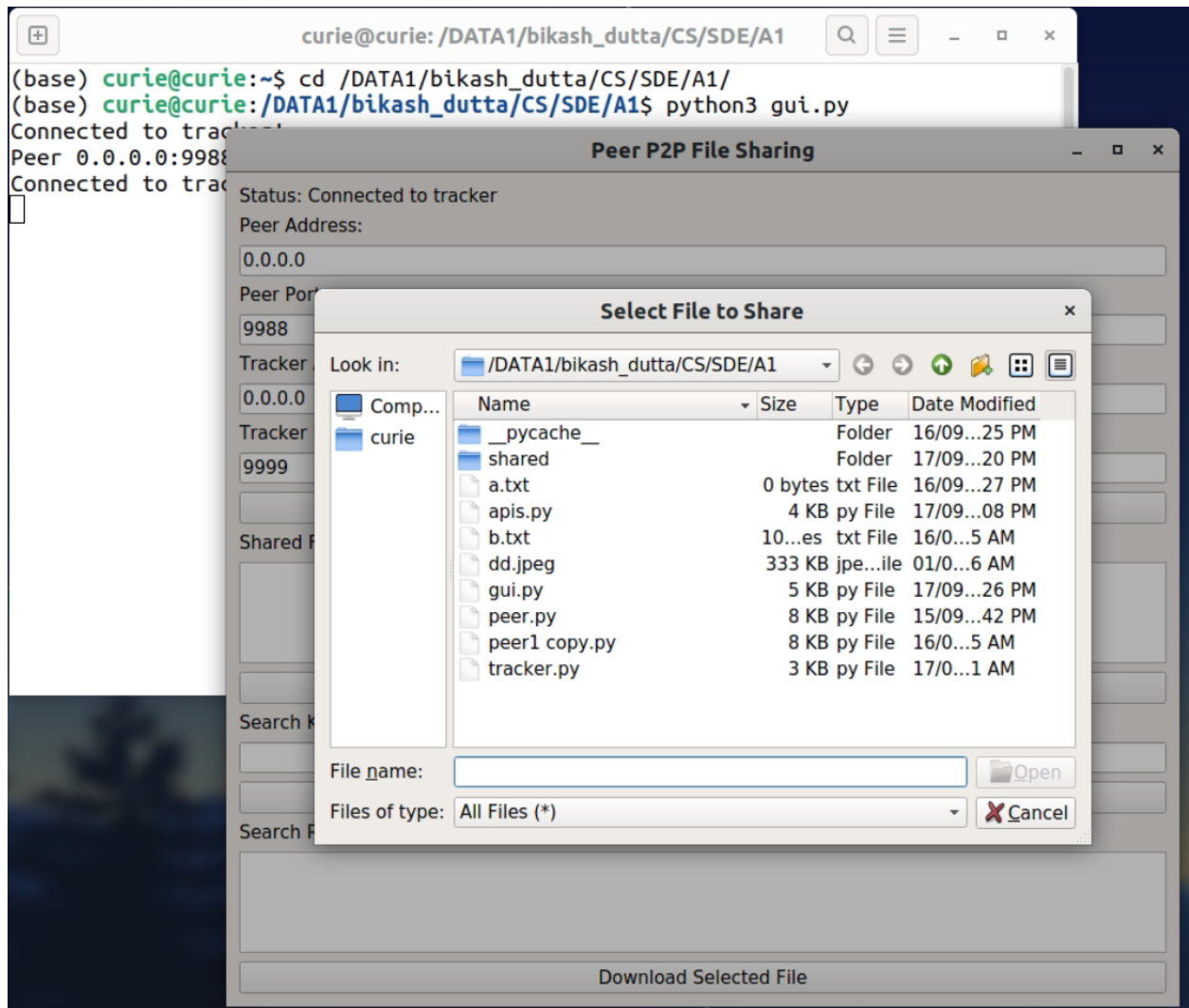
Server response

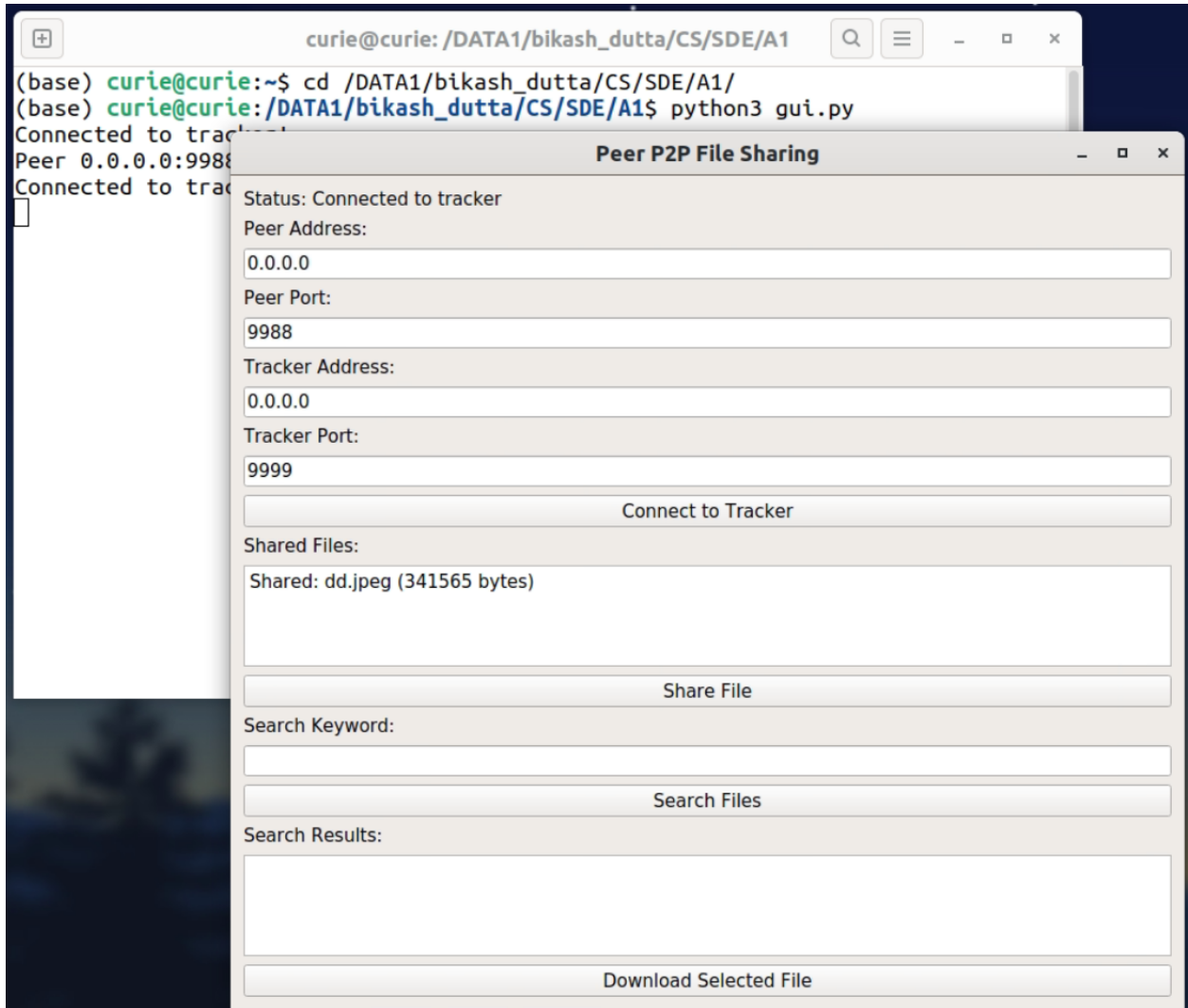
Code	Details
200	<p>Response body</p> <pre>{ "message": "Files have been shared.", "shared_files": [{ "filename": "forming.webp", "size": 193288 }] }</pre> <p>Response headers</p> <pre>content-length: 96 content-type: application/json date: Sun, 17 Sep 2023 16:50:48 GMT server: uvicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links
Media type: application/json		
Controls Accept header.		
Example Value Schema		
"string"		
422	Validation Error	No links
Media type: application/json		

- GUI





- Searching & Downloading:
 - APIs

POST /search/ Search Files

Parameters

No parameters

Request body required application/json

```
{
  "keyword": "a.txt"
}
```

Execute

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls: Accept header.

Example Value | Schema

```
"string"
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8000/search/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "keyword": "a.txt"
  }'
```

Request URL

```
http://localhost:8000/search/
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "results": [["a.txt", 0, "0.0.0.0:9987"]] }</pre> <p>Response headers</p> <pre>content-length: 40 content-type: application/json date: Sun, 17 Sep 2023 17:01:10 GMT server: uvicorn</pre>

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls: Accept header.

Example Value | Schema



GET /download/ Download File

Parameters

Cancel

Name	Description
filename <small>required</small>	
string (query)	<input type="text" value="a.txt"/>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/download/?filename=a.txt' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/download/?filename=a.txt
```

Server response

Code

Details

200

Response headers

```
content-disposition: attachment; filename=a.txt
content-length: 0
content-type: application/octet-stream
date: Sun, 17 Sep 2023 17:01:50 GMT
server: uvicorn
```

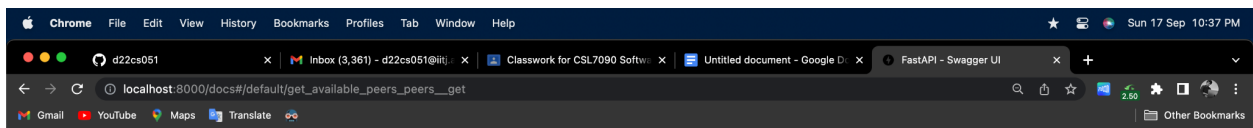
Responses

Code	Description	Links
200	Successful Response	No links

Media type

application/json

Protektio API-ko dokumenta



GET /peers/ Get Available Peers

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/peers/' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/peers/
```

Server response

Code

Details

200

Response body

```
{
  "peers": [
    {
      "name": "Peer1",
      "address": "0.0.0.0",
      "port": 9988
    },
    {
      "name": "Peer1",
      "address": "0.0.0.0",
      "port": 9985
    },
    {
      "name": "pi_gui_test",
      "address": "0.0.0.0",
      "port": 9987
    }
  ]
}
```

Response headers

Download

- CLI

```
1. Share a file
2. Search for a file
3. Quit
Enter your choice: 2
Enter a keyword to search for files:
Connected to tracker!
response dd.jpeg:341565 from (0.0.0.0:9988)
response shared/formimg.webp:193280 from (0.0.0.0:9985)
results: [('dd.jpeg', 341565, '0.0.0.0:9988'), ('shared/formimg.webp', 193280, '0.0.0.0:9985')]
Search Results:
1. dd.jpeg (341565 bytes)
2. shared/formimg.webp (193280 bytes)
Enter the number of the file to download (or press Enter to go back): █
```

SSH: curie 0 0 1

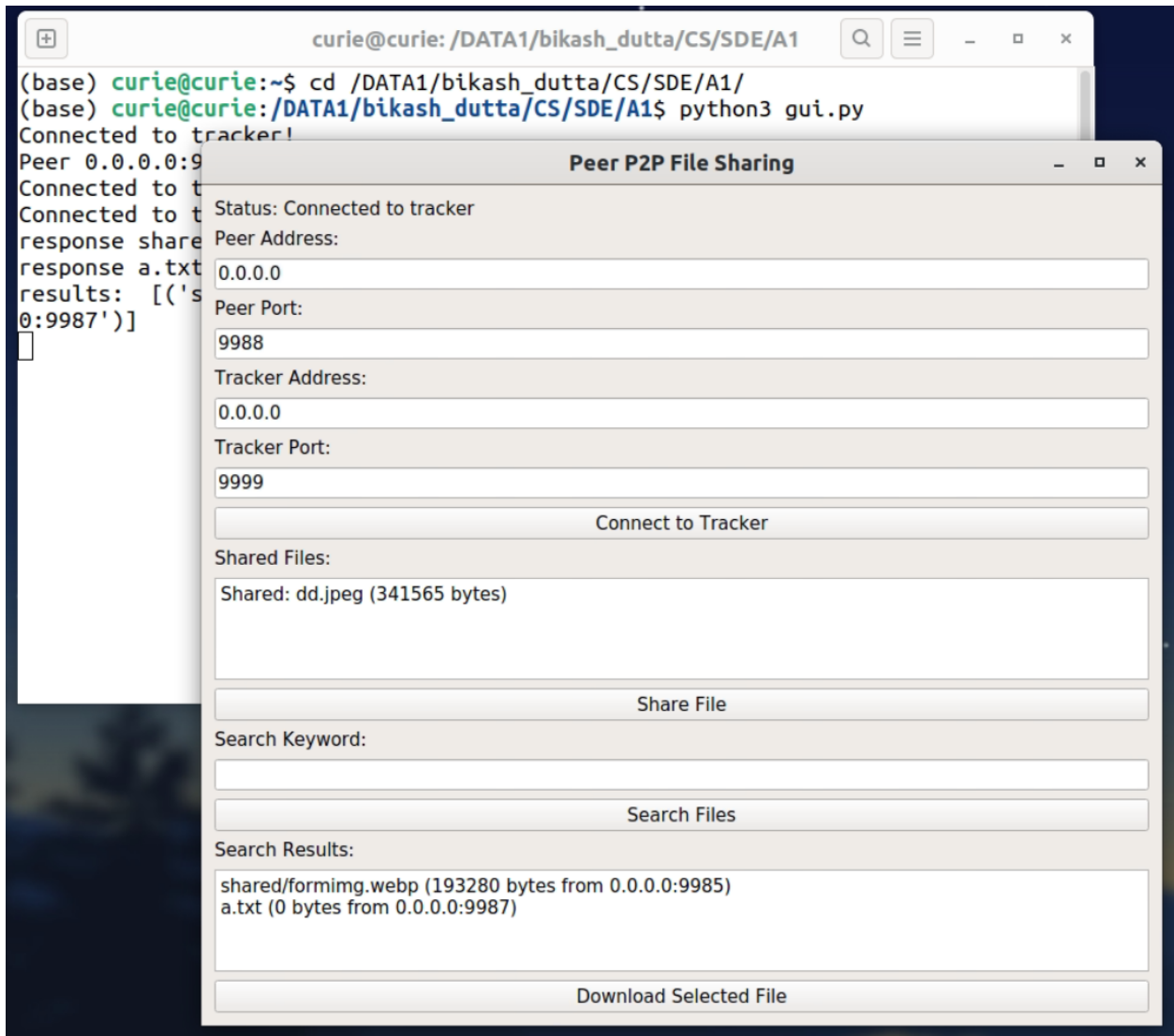
```
1. Share a file
2. Search for a file
3. Quit
Enter your choice: 2
Enter a keyword to search for files:
Connected to tracker!
response dd.jpeg:341565 from (0.0.0.0:9988)
response shared/formimg.webp:193280 from (0.0.0.0:9985)
results: [('dd.jpeg', 341565, '0.0.0.0:9988'), ('shared/formimg.webp', 193280, '0.0.0.0:9985')]
Search Results:
1. dd.jpeg (341565 bytes)
2. shared/formimg.webp (193280 bytes)
Enter the number of the file to download (or press Enter to go back): 1
0.0.0.0:9988 <class 'str'>
In Download, peer ip:0.0.0.0, port: 9988
downloading.....
File 'dd.jpeg' downloaded successfully.
```

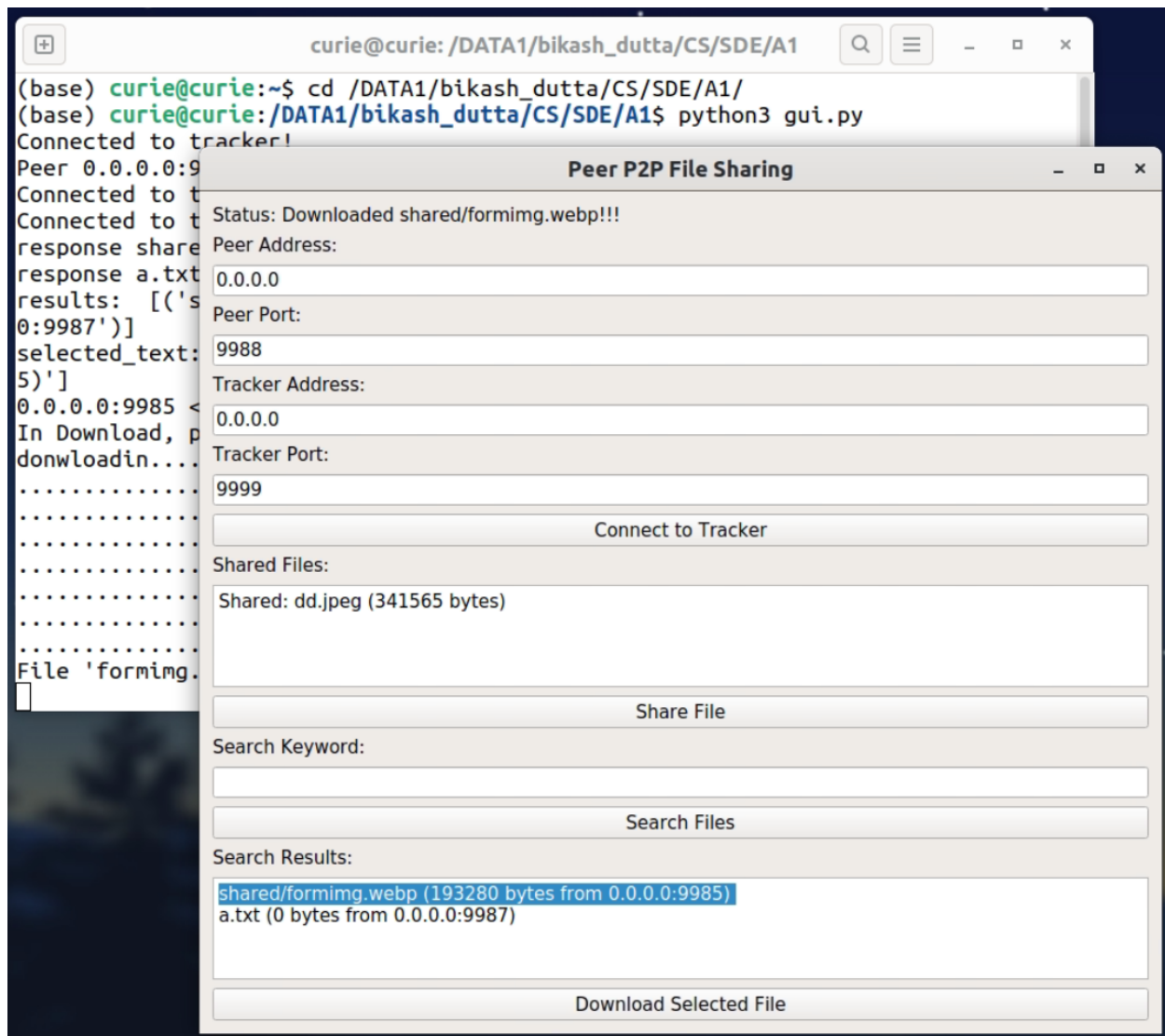
```
1. Share a file
2. Search for a file
3. Quit
Enter your choice: █
```

SSH: curie 0 0 1

- GUI

NOTE: “blank search text = All the available file in the n/w” and “select the file and click download”





References:

- <https://www.youtube.com/watch?v=2v6KqRB7adg>
- <https://youtube.com/playlist?list=PLS1QuIW01RIZGSgRsn0b8w9uoWM1gHDpo&si=LVQKd0I5MS45NtWQ>
- <https://youtu.be/7t2alSnE2-I?si=LK8znV6b5G4TmcTj>
- <https://youtube.com/playlist?list=PLs3IFJPw3G9Jhknhl-mfGxcD04IEOIZJX&si=qe6hJbFsKcndcTOK>
- <https://medium.com/@luishrsoares/implementing-peer-to-peer-data-exchange-in-python-8e69513489af>
- <https://www.youtube.com/watch?v=1Fay1pjttLg>
- <https://medium.com/hackernoon/socket-programming-in-python-client-server-and-peer-examples-a25c9782b584>
- <https://fastapi.tiangolo.com/tutorial/>

- <https://www.riverbankcomputing.com/static/Docs/PyQt5/>
- <https://doc.qt.io/qtforpython-6/>
- <https://www.geeksforgeeks.org/socket-programming-python/>
- <https://docs.python.org/3/library/socket.html>
- <https://docs.python.org/3/library/threading.html>