

Secure P2P with File Sharing System

Group Members

- **Member 1:** Aneesh Bharadwaj K S (PES1UG23AM047)
 - **Member 2:** Akanksh Rai (PES1UG23AM031)
-

Aim

To develop a peer-to-peer (P2P) file sharing system that allows users to download and view files using both server-peer and peer-to-peer connections, providing a simple and functional interface for efficient file access and transfer.

Project Design

The system architecture includes two primary modes of file sharing:

1. **Server-Peer Communication**
A centralized server hosts files which can be requested and downloaded by peers.
 2. **Peer-to-Peer Communication**
Peers can discover each other, initiate direct connections, and share files among themselves.
- The frontend uses **Tkinter** for GUI-based interaction and file display.
-

Tools and Technologies Used

- **Programming Language:** Python
 - **Frontend:** Tkinter
 - **Networking:** Python's socket library
 - **Threading:** Python's threading module for handling multiple connections
 - **File Handling:** Python I/O operations
-

Implementation Procedure

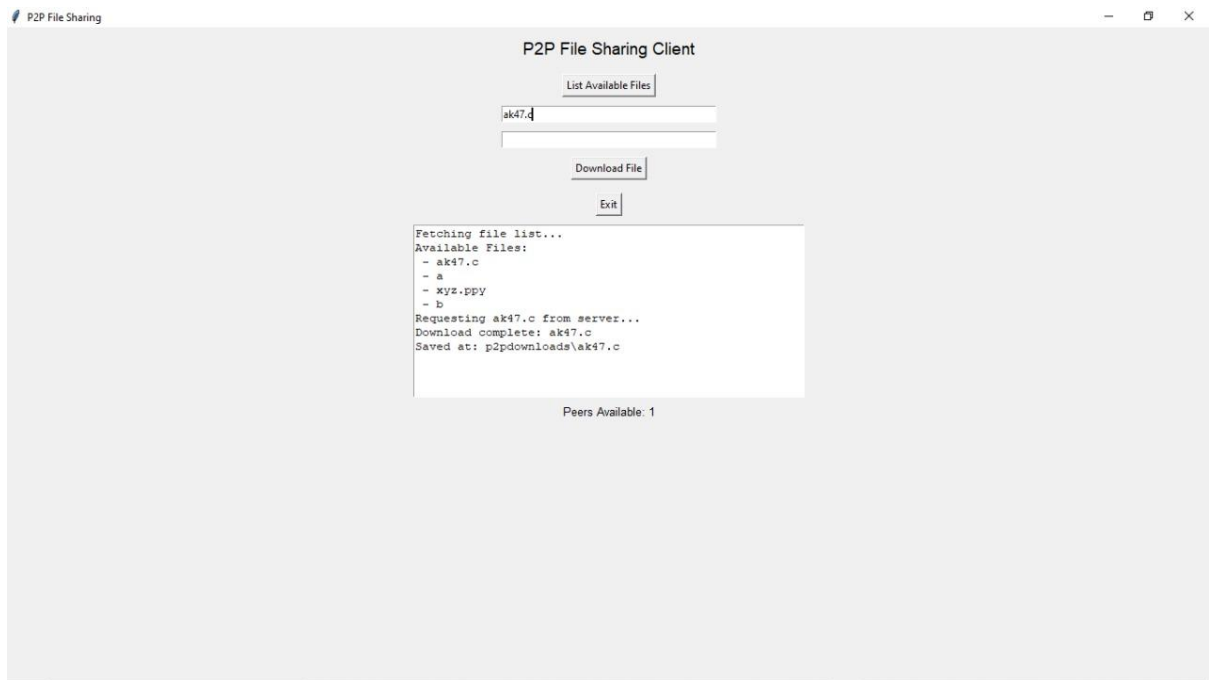
1. Server-Peer Connection

- The server maintains a list of available files.
- A peer can:
 - Connect to the server
 - Request a file
 - Download and display the file

Screenshot: Server-Peer File Request and Download

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Filter
[Running] python -u "c:\Users\asus\OneDrive\Desktop\aneesh\client.py"
[+] Client is running. Ready to download files.
[+] Peer server running on port 6000
```

```
> __pycache__
> .dist
v p2pdownloads
  v .dist
  ≡ 123.txt
  ≡ abc.txt
v shared_files
  ≡ a1.txt
+ client.py
+ gui.py
```



server's terminal:

```
aneesh@aneesh: ~/Desktop
Apr 10, 10:19
aneesh@aneesh: ~/Desktop
aneesh@aneesh:~/Desktop$ python3 server.py
[+] Peer discovery server running on port 5000
Debug: Received request -> LIST
Debug: Sent file list -> ak47.c;a;xyz.ppy;b
Debug: Received request -> LIST
Debug: Sent file list -> ak47.c;a;xyz.ppy;b
Debug: Received request -> GET ak47.c
Debug: Sent ak47.c successfully

aneesh@aneesh:~/Desktop$ hostname -I
192.168.251.1 2409:40f2:101f:fe54:ca18:6d72:24b1:fb92 2409:40f2:101f:fe54:eaf:dcd7:470d:6532
aneesh@aneesh:~/Desktop$
```

2. Peer-to-Peer Connection

- One peer can connect directly to another peer via IP and port.
- Files are shared without routing through the central server.

- A peer can:
 - Request a file from another peer
 - Download and display the file contents

Screenshot: Peer-to-Peer File Transfer

```

Command Prompt

Ethernet adapter Ethernet 5:

    Connection-specific DNS Suffix  . : 
    Link-Local IPv6 Address . . . . . : fe80::a54a:76a8:7623:8f67%3
    IPv4 Address. . . . . : 192.168.56.1
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 

Wireless LAN adapter Local Area Connection* 3:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Wireless LAN adapter Local Area Connection* 4:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Wireless LAN adapter Wi-Fi:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2409:40f2:101f:fe54:823d:bd12:80e:e732
    Temporary IPv6 Address. . . . . : 2409:40f2:101f:fe54:544b:8ad5:aed:d619
    Link-Local IPv6 Address . . . . . : fe80::93ea:743f:dfe2:b0db%19
    IPv4 Address. . . . . : 192.168.251.91
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : fe80::40cc:3bff:fe94:45c0%19
                                192.168.251.56

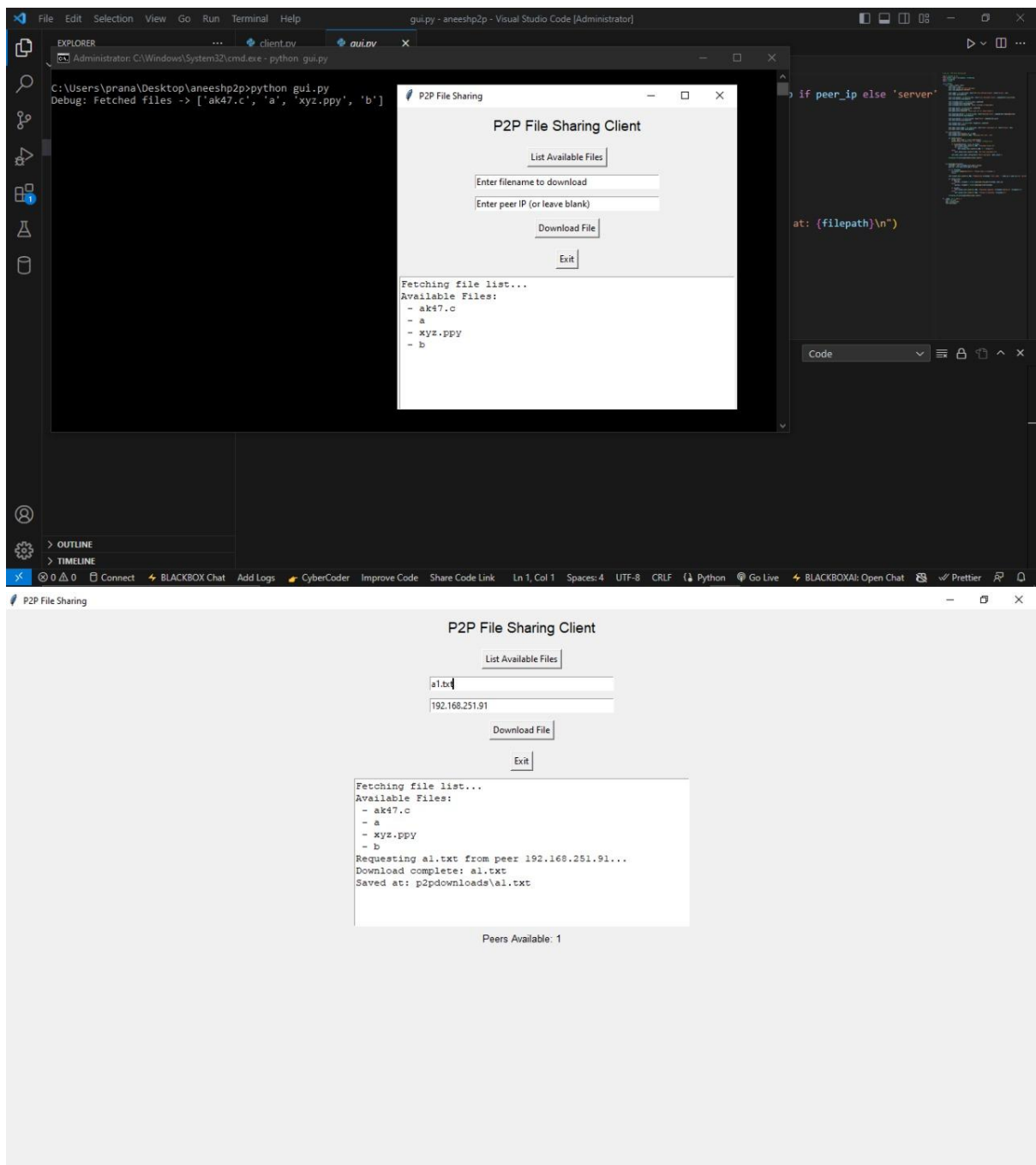
C:\Users\asus>
  
```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[Running] python -u "c:\Users\asus\OneDrive\Desktop\aneesh\client.py"
[+] Client is running. Ready to download files.
[+] Peer server running on port 6000
[+] Connection from ('192.168.251.200', 52280)
[+] Incoming request: GET a1.txt
[+] Requested file path: shared_files\a1.txt
[+] Sent file: a1.txt
  
```

other – peer:



Positive & Negative Scenarios

Positive Scenarios

- Successful file transfer from server to peer.
- File sharing between two peers.
- Proper file listing and display in GUI.

Negative Scenarios

- Attempt to download a non-existent file.
 - Peer offline/unreachable during connection attempt.
 - Server down or port conflict handling.
-

Key Learnings

- Gained hands-on experience with socket programming in Python.
 - Understood client-server vs peer-to-peer architecture differences.
 - Applied multi-threading to manage simultaneous connections.
 - Built interactive GUI with Tkinter.
-

Future Scope

- **Authentication:** Add user login and authentication before file access.
 - **Distributed File Indexing:** Replace central server with DHT (Distributed Hash Table) for decentralized peer discovery.
 - **File Integrity:** Use hashing (e.g., SHA-256) to ensure file integrity during transfers.
 - **Cross-Platform GUI:** Expand UI for web and mobile compatibility using frameworks like Flask or Electron.
-