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

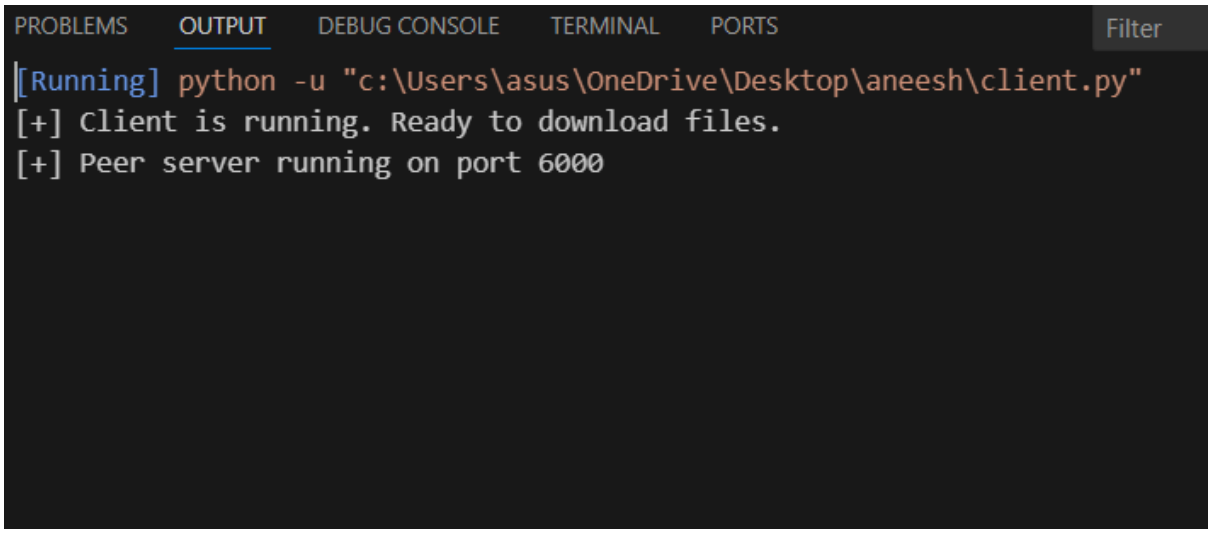
- **Programming Language**: Python
- **Frontend**: Tkinter
- **Networking**: Python's socket library
- **Threading**: Python’s threading module for handling multiple connections
- **File Handling**: Python I/O operations

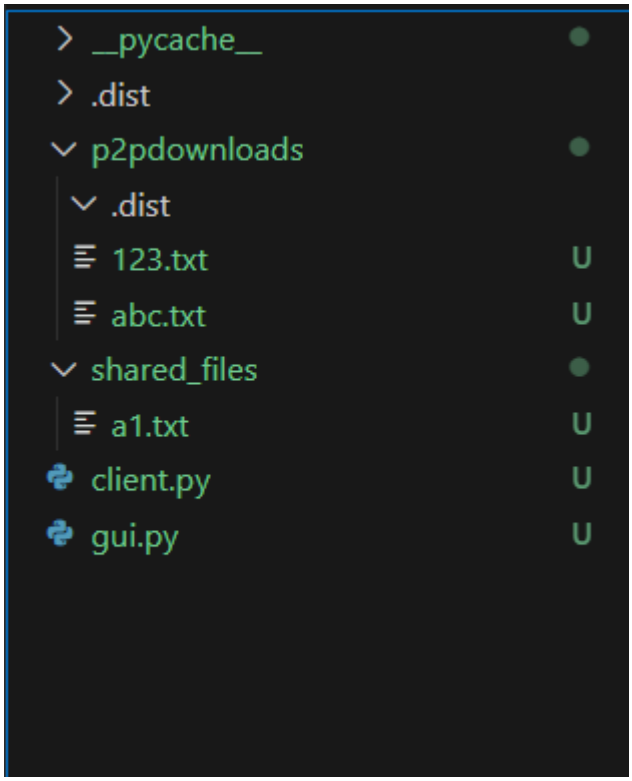
Implementation Procedure

1. Server-Peer Connection

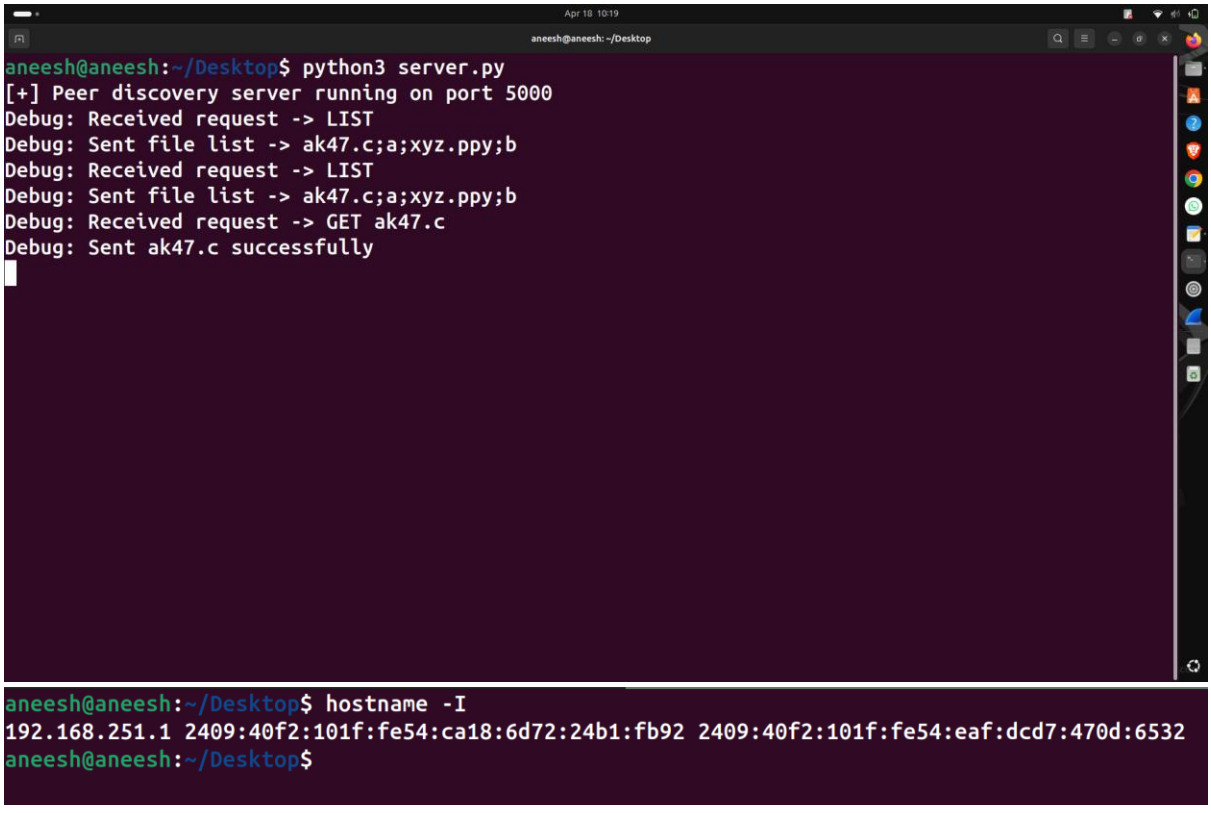
- The server maintains a list of available files.
- A peer can:
    - Connect to the server
    - Request a file
    - Download and display the file

Screenshot: Server-Peer File Request and Download





server's terminal:

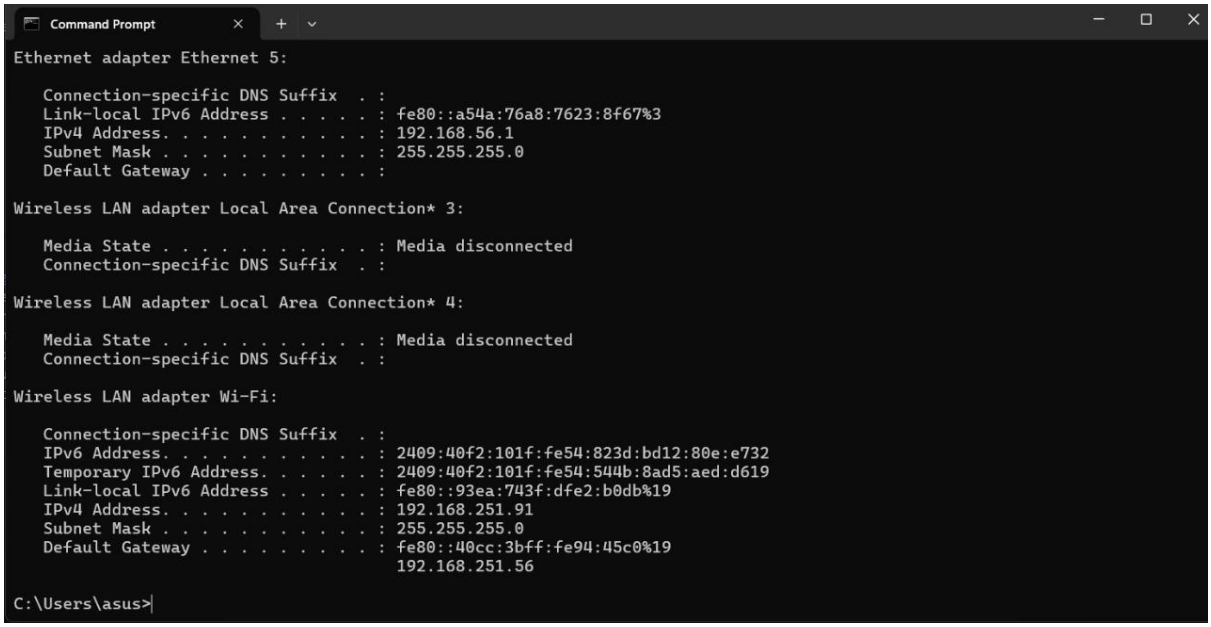


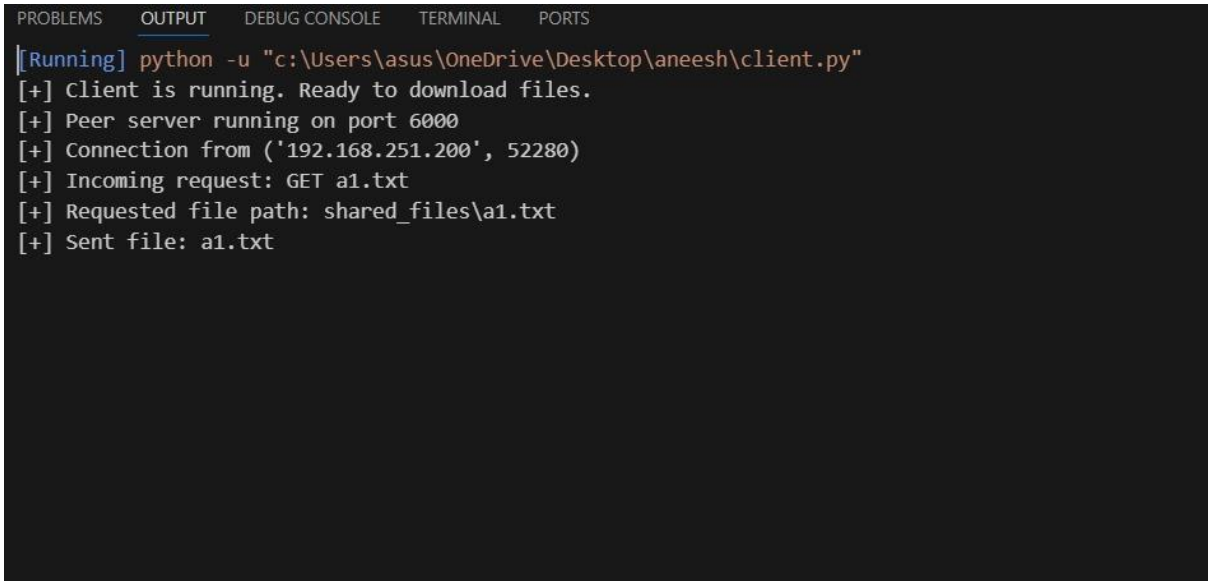
2. Peer-to-Peer Connection

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

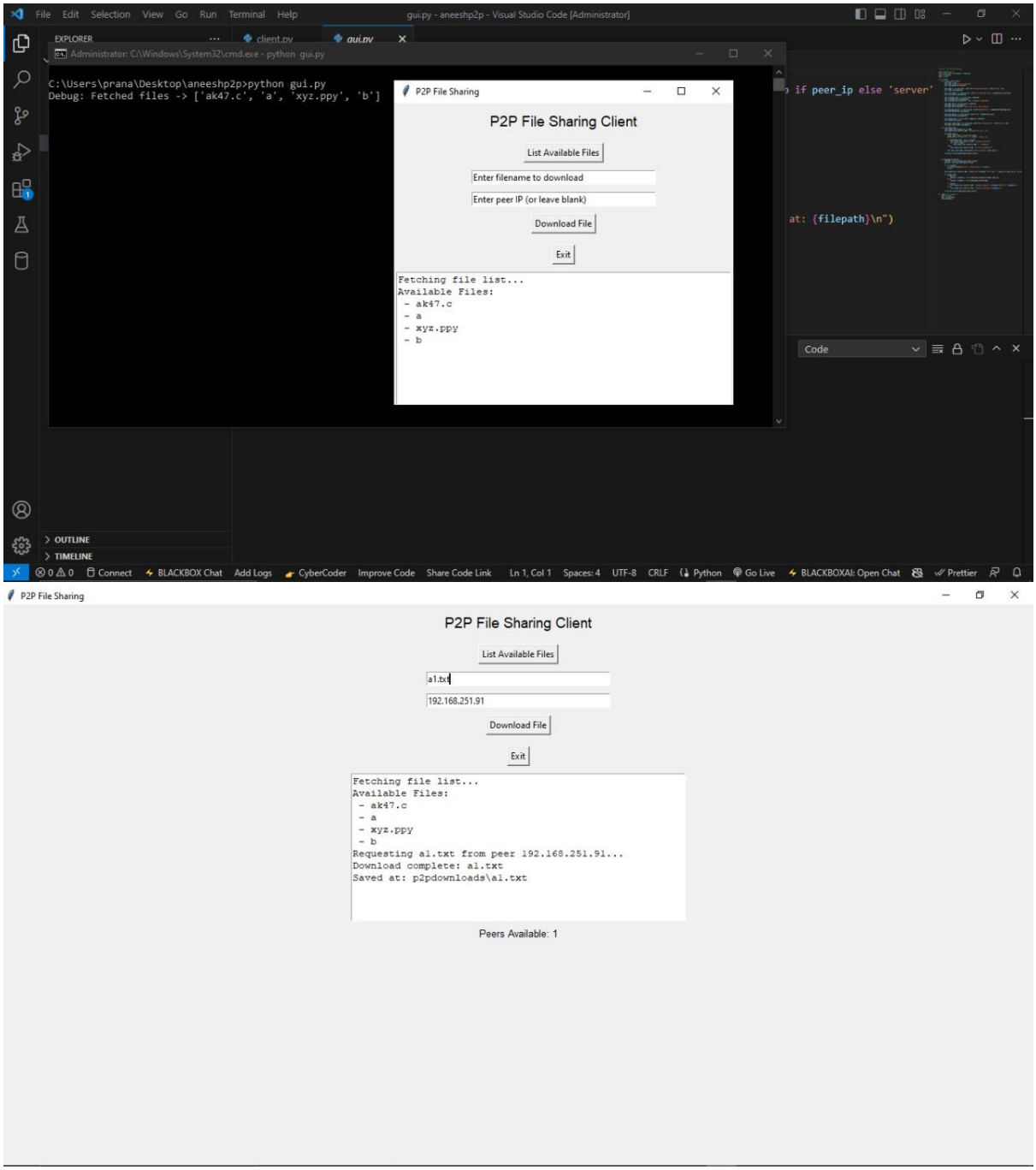
- A peer can:
    - Request a file from another peer
    - Download and display the file contents

Screenshot: Peer-to-Peer File Transfer





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.
