**A computer screen shot of text

AI-generated content may be incorrect.**

**main()**

**📌 Purpose:  
Initializes the server, binds it to an IP and port, and continuously listens for incoming client connections.**

**🧠 What it does:**

1. **Creates a TCP socket using IPv4 addressing.**
2. **Sets socket options to allow quick reuse of the address.**
3. **Binds the server to the specified HOST and PORT.**
4. **Starts listening for incoming connections.**
5. **For each new client, spawns a new thread using handle\_client().**

**The main function is the entry point of the server application. It configures and launches the server, making it capable of accepting multiple concurrent client connections using threading. This allows the server to handle each client independently and continuously, supporting parallel file-sharing interactions across the network.**

**A computer screen shot of a program code

AI-generated content may be incorrect.**

**✅ handle\_client(conn, addr)**

**📌 Purpose:  
Handles all interactions (LIST, UPLOAD, DOWNLOAD) with a single connected client.**

**🧠 What it does:**

1. **Logs the new client connection.**
2. **Waits to receive a command from the client.**
3. **Splits the command and identifies whether it is LIST, UPLOAD, or DOWNLOAD.**
4. **Dispatches the request to the appropriate function.**
5. **Closes the connection and logs disconnection or errors.**

**The handle\_client function serves as the command interpreter for each connected client. It listens for requests, processes them accordingly, and ensures proper communication handling and logging. Each client’s connection is managed in its own thread, enabling smooth multiclient support.**

**A computer screen shot of a program

AI-generated content may be incorrect.**

**✅ handle\_upload(conn, parts, addr)**

**📌 Purpose:  
Receives a file from the client and stores it on the server, handling version conflicts.**

**🧠 What it does:**

1. **Extracts filename and file size from the command.**
2. **Automatically renames the file using generate\_versioned\_filename() if it already exists.**
3. **Sends "READY\_REC" to the client.**
4. **Receives the file in chunks (1024 bytes).**
5. **Writes the data to disk and logs the upload.**

**The handle\_upload function enables the server to receive files from clients. It ensures file name conflicts are avoided by renaming duplicate files, and manages the data reception process efficiently using buffered chunks. This function contributes to the robustness of file storage and prevents unintentional overwrites.**

**A computer screen shot of text

AI-generated content may be incorrect.**

**✅ download\_file\_handler(conn, cmd\_parts, addr)**

**📌 Purpose:  
Sends a requested file to the client after verifying its existence.**

**🧠 What it does:**

1. **Extracts the filename from the command.**
2. **Checks if the file exists in the server's directory.**
3. **If it exists, sends the file size and waits for the client's readiness.**
4. **Sends the file in binary form using conn.sendfile().**
5. **If the file doesn't exist, returns an error message.**

**The download\_file\_handler function handles client requests to retrieve specific files. It checks for availability, communicates file metadata, and streams the file efficiently. This helps provide reliable and fast downloads to clients, while handling errors gracefully.**

**A computer screen with text on it

AI-generated content may be incorrect.**

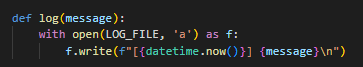
**✅ handle\_list(conn)**

**📌 Purpose:  
Sends a list of all available files on the server to the client.**

**🧠 What it does:**

1. **Reads the list of filenames in the files/ directory.**
2. **Sends "ACK\_LIST" to confirm readiness.**
3. **Waits for the client's "READY\_FOR\_LIST" acknowledgment.**
4. **Sends the file list or a message indicating no files are available.**

**The handle\_list function allows the server to share a directory listing of stored files with connected clients. It ensures protocol synchronization by waiting for the correct client response before proceeding, preventing miscommunication or data loss during transmission.**

****

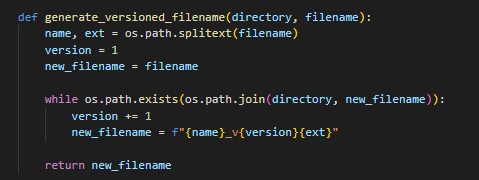
**✅ log(message)**

**📌 Purpose:  
Logs server-side actions such as file transfers, client connections, and errors.**

**🧠 What it does:**

1. **Opens the server log file in append mode.**
2. **Adds a timestamp to the message.**
3. **Writes the log entry into logs\_server/server\_log.txt.**

**The log function is a reusable utility that records all significant server activities. By maintaining a timestamped log of operations, it facilitates debugging, auditing, and system monitoring.**

****

**✅ generate\_versioned\_filename(directory, filename)**

**📌 Purpose:  
Prevents file overwriting by automatically generating a new versioned filename.**

**🧠 What it does:**

1. **Splits the filename into name and extension.**
2. **Checks if a file with the same name exists in the target directory.**
3. **If yes, appends \_vX (e.g., \_v2, \_v3, etc.) to the filename until a unique name is found.**
4. **Returns the unique, versioned filename.**

**The generate\_versioned\_filename function is essential for maintaining data integrity by preventing file overwrites. It ensures that every uploaded file is stored safely, even if a file with the same name was previously uploaded.**

**A computer screen shot of text

AI-generated content may be incorrect.**

**✅ File & Directory Initialization (at the top)**

**📌 Purpose:  
Ensures that necessary directories and log files exist before the server starts.**

**🧠 What it does:**

1. **Defines folders: files/ for uploaded files and logs\_server/ for logs.**
2. **Uses os.makedirs() with exist\_ok=True to create them if missing.**
3. **Initializes the server\_log.txt file with a creation timestamp if it doesn’t already exist.**

**This setup guarantees the server always has the correct folder structure and logging in place before serving client requests, reducing the risk of crashes due to missing paths.**