Network File Sharing Server and Client

1. Project Overview

This project is a networked file sharing application developed in C++ for Assignment 4 of the Capstone Project. It implements a client-server architecture using TCP/IP sockets to allow users to connect to a server, authenticate, and perform basic file transfer operations.

The server is multi-threaded, enabling it to handle multiple client connections simultaneously. A simple text-based protocol is used for communication, and a basic XOR encryption layer is included to demonstrate a security concept.

2. Features

- **Client-Server Architecture:** A robust multi-threaded TCP server and a console-based client.
- **User Authentication:** Clients must authenticate with a valid username and password (admin / password123) before accessing any features.
- **List Remote Files:** Clients can request a list of all files currently available in the server's storage directory.
- **File Download (GET):** Clients can download files from the server's storage directory to their local client_files/ directory.
- **File Upload (PUT):** Clients can upload files from their local client_files/ directory to the server's storage.
- **Simple Encryption:** All communication (commands and file data) is encrypted using a simple XOR cipher for demonstration purposes.

3. Technologies Used

- **Language:** C++ (std=c++17)
- **Networking:** POSIX Sockets (TCP/IP)
- **Concurrency:** <thread> and <mutex> for the multi-threaded server.
- **Build System:** GNU Make

4. How to Compile

A Makefile is provided for easy compilation on a Linux system with g++ installed.

Navigate to the project's root directory in your terminal and simply run:

```
make
```

This will compile and create two executables in the same directory:

- server
- client

To remove the compiled executables, you can run:

```
make clean
```

5. How to Run

You must start the server first.

5.1. 1. Start the Server

In one terminal window, run the server executable:

```
./server
```

The server will start and print a log message indicating it is waiting for connections:

```
[Server] Waiting for connections on port 8080...
```

5.2. 2. Run the Client

In a **separate** terminal window, run the client executable:

```
./client
```

5.3. 3. Usage

1. The client will connect and ask for credentials.
2. **Username:** admin
3. **Password:** password123
4. Upon successful authentication, you will see the main menu:

```
--- Client Menu ---
1. List files on server
2. Download file
3. Upload file
4. Exit
Enter choice:
```

5. **To Upload:** Place a file (e.g., MyFile.txt) into the client_files/ directory. Then, choose option 3 and type MyFile.txt when prompted.
6. **To Download:** Choose option 2 and type the name of a file on the server (e.g., MyFile.txt). It will be saved to your client_files/ directory.