# SMART Objectives for Peer-to-Peer File Sync System

## Client-Side Objectives

1. Implement user authentication system using POST request and hashed password.
2. Encode login data as JSON and send it over TCP socket to the server.
3. Receive login response and update Flask session and flash message accordingly.
4. Use Flask-SocketIO to send and receive real-time folder and device updates.
5. Enable client to send IP address and MAC info to server on dashboard load.
6. Track pending sync queue using folder traversal and MD5 hashing.
7. Store sync events into a queue and trigger sync only if `sync\_active` is set.
8. Respond to 'accept\_share' by creating local folders and updating `dir.json`.
9. Track folder sync status using file count vs expected count and update UI.
10. Send add\_folder requests with properly expanded user paths and create directory.

## Server-Side Objectives

1. Start TCP socket server on port 8000 and listen for incoming client messages.
2. Handle JSON-encoded login/register requests and return appropriate status codes.
3. Track and persist device MAC addresses and names per user in SQLAlchemy database.
4. Store and retrieve users, devices, and folders using ORM mappings and filters.
5. Process add\_folder action, validate directory paths, and format Linux-style paths.
6. Create folder in file system and store path and folder metadata in DB.
7. Update invites.json with new share info and send share invitation to users.
8. Handle file sync on port 9000 with checksum verification and MD5 integrity check.
9. Dispatch received file metadata to appropriate handler: Create, Modify, Move, or Delete.
10. Log each event's success or failure with packet retries on checksum mismatch.