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

* 🔵 Client Objectives
* User Authentication and Session Management
* Hash password input using SHA256.
* Convert login form data into JSON dictionary.
* Base64 encode JSON packet for secure transmission.
* Open TCP socket and send login packet to port 8000.
* Receive and decode server response.
* On success, flash status and redirect to dashboard.
* Handle 401 and 404 errors with user alerts.
* Device Registration
* Fetch MAC address from client device.
* Send device name, username, and MAC address to server.
* Receive confirmation and store metadata in local DB.
* Folder Creation and Sharing
* Generate unique folder ID.
* Create directory using `os.makedirs()`.
* Store metadata in `dir.json` and database.
* Prepare share requests with usernames and device names.
* Send share metadata to server to write to `invites.json`.
* Sync Queue and Folder Traversal
* Use stack-based preorder traversal for folder indexing.
* Push file metadata (hash, size, path) to sync queue.
* Persist events in `sync\_queue.json` for tracking.
* Update file changes using `watchdog` observers.
* SocketIO Real-time Updates
* Emit alerts to frontend via WebSocket.
* Implement `check\_sync\_status`, `request\_folders`, etc.
* Use structured emit/receive patterns.
* 🔴 Server Objectives
* Socket Server Initialization
* Bind TCP socket to port 8000.
* Listen for client messages and accept connections.
* Parse incoming JSON data and dispatch to handler.
* Database Management with SQLAlchemy
* Define models: User, Device, Folder, Share, File.
* Use composite primary keys where necessary.
* Commit and rollback sessions using `sessionmaker`.
* User and Device Operations
* Create new users if username doesn't exist.
* Check hash on login and respond with status.
* Register devices only if MAC and name are unique.
* Folder and Share Management
* Validate folder path using regex for OS compatibility.
* Check for existing folder\_id before creation.
* Write share requests into `invites.json`.
* Insert new Folder and Share records in DB.
* Event Processing via SyncWorker
* Accept file sync connection on port 9000.
* Use `recv\_exact()` to read header and payload.
* Unpack JSON metadata and determine `event\_type`.
* Handle File Events
* Created
* Dispatch to `CreateDir.apply()` if directory.
* Dispatch to `CreateFile.apply()` if file.
* Modified
* Dispatch to `Modify.apply()` with updated metadata.
* Verify checksum before overwriting existing file.
* Deleted
* Dispatch to `Delete.apply()` to remove files/directories.
* Update database and filesystem accordingly.
* Moved
* Dispatch to `Move.apply()` to update path.
* Ensure `path` is changed in both FS and DB.
* File Integrity and Packet Verification
* Base64 decode and hash file blocks with MD5.
* Compare against transmitted checksum.
* Send ACK/ERR based on hash match result.