📦 Module Structure Overview

- **Server Modules:** Focus on managing devices, processing requests, and handling network communication.
- **Client Modules:** Handle user interaction, send commands, and receive responses.
- **Shared Modules:** Contain common code used by both server and client, including the protocol, network simulation, and logging.

🟢 Server Modules (`/server`)

These modules are only relevant to server-side operations.

1. **Server Core**
   - `Server.cpp`/ `Server.h` → Main class for server operations.
   - Handles client connections and requests.
2. **Device Management**
   - `Device.cpp`/ `Device.h` → Represents individual devices.
   - `DeviceManager.cpp`/ `DeviceManager.h` → Manages all devices.
3. **Thread Management**
   - `ThreadManager.cpp`/ `ThreadManager.h` → Manages multithreading for concurrent requests.
4. **Configuration Management**
   - `ConfigurationManager.cpp`/ `ConfigurationManager.h` → Handles configuration settings for the server.
5. **Logging** *(Optional if more detailed logging is needed)*
   - `Logger.cpp`/ `Logger.h` → Logs system activities for debugging and monitoring.
6. **Routing**
   - `Router.cpp`/ `Router.h` → Handles packet routing between subnets.

🟠 Client Modules (`/client`)

These modules are focused on user interaction and communication with the server.

1. **Client Core**
   - `Client.cpp`/ `Client.h` → Manages client connections and commands.
2. **User Interface**
   - `UI.cpp`/ `UI.h` → Provides a simple text-based interface for users to control devices.
3. **Protocol Handling**

- Uses the shared `ProtocolHandler.cpp`/ `ProtocolHandler.h`to encode/decode commands.

🟡 Shared Modules (`/shared`)

These modules will be used by both the server and the client. They provide essential functionalities that should be consistent across both applications.

1. **Protocol Management**
   - `ProtocolHandler.cpp`/ `ProtocolHandler.h` → Manages request and response encoding/decoding using a simple HTTP-like protocol.
2. **Networking**
   - `Network.cpp`/ `Network.h` → Simulates network communication (TCP/IP, ARP, ICMP).
3. **Logger** *(If you want a shared logging system)*
   - `Logger.cpp`/ `Logger.h` → Provides basic logging utilities.
4. **Utilities** *(Optional)*
   - `Utils.cpp`/ `Utils.h` → General helper functions for validation, string manipulation, etc.

Heading 2