1. 用户管理模块

用户类（`User`）

```
class User {
public:
    enum class Role { Admin, User };

    User(int id, const std::string& username, const std::string&
password_hash, Role role)
        : id(id), username(username), password_hash(password_hash),
role(role) {}

    int getId() const { return id; }
    const std::string& getUsername() const { return username; }
    const std::string& getPasswordHash() const { return password_hash; }
    Role getRole() const { return role; }

private:
    int id;
    std::string username;
    std::string password_hash;
    Role role;
};
```

用户管理类（`UserManager`）

```
class UserManager {
public:
    UserManager();
    bool registerUser(const std::string& username, const std::string&
password);
    bool loginUser(const std::string& username, const std::string&
password);
    void logoutUser(int userId);
    User* getUser(int userId);

private:
    std::map<int, User> users;
    std::map<int, std::string> sessions;  // 记录会话信息，如登录时间等
};
```

2. 设备管理模块

设备类（`Device`）

```
class Device {
public:
    enum class DeviceType { Light, Thermostat, Camera, Socket };

    Device(int id, DeviceType type)
```

```
        : id(id), type(type), status("off") {}

    int getId() const { return id; }
    DeviceType getType() const { return type; }
    const std::string& getStatus() const { return status; }
    void setStatus(const std::string& newStatus) { status = newStatus; }

private:
    int id;
    DeviceType type;
    std::string status;  // 设备状态，如 "on", "off", "adjusted"
};
```

设备管理类（`DeviceManager`）

```
class DeviceManager {
public:
    DeviceManager();
    void addDevice(const Device& device);
    void removeDevice(int deviceId);
    Device* getDevice(int deviceId);
    void updateDeviceStatus(int deviceId, const std::string& newStatus);

    std::vector<Device> getAllDevices() const;
private:
    std::map<int, Device> devices;
};
```

3. 日志管理模块

日志类（`Log`）

```
class Log {
public:
    enum class LogType { UserAction, DeviceAction, SystemError };

    Log(LogType logType, int userId, int deviceId, const std::string&
content)
        : logType(logType), userId(userId), deviceId(deviceId),
content(content) {}

    LogType getLogType() const { return logType; }
    int getUserId() const { return userId; }
    int getDeviceId() const { return deviceId; }
    const std::string& getContent() const { return content; }

private:
    LogType logType;
    int userId;
    int deviceId;
    std::string content;
};
```

日志管理类（`LogManager`）

```
class LogManager {
public:
    LogManager();
    void logAction(Log::LogType logType, int userId, int deviceId, const
std::string& content);
    void saveLogsToFile(const std::string& filename);
    std::vector<Log> getLogsByUser(int userId) const;
    std::vector<Log> getLogsByDevice(int deviceId) const;

private:
    std::vector<Log> logs;
    std::mutex logMutex;  // 用于线程安全的日志写入
};
```

4. 配置管理模块

配置管理类（`ConfigManager`）

```
class ConfigManager {
public:
    ConfigManager(const std::string& configFile);
    void loadConfig();
    void saveConfig();
    void updateConfig(const std::string& key, const std::string& value);
    std::string getConfig(const std::string& key);

private:
    std::map<std::string, std::string> configData;
    std::string configFile;
};
```

5. 数据库管理模块

数据库操作类（`DatabaseManager`）

```
class DatabaseManager {
public:
    DatabaseManager(const std::string& dbPath);
    bool connect();
    bool disconnect();
    bool executeQuery(const std::string& query);
    std::vector<std::string> fetchResults(const std::string& query);

private:
    std::string dbPath;
    sqlite3* db;
};
```

6. 线程池管理模块

线程池类（`ThreadPool`）

```cpp
class ThreadPool {
public:
    ThreadPool(size_t numThreads);
    void enqueueTask(std::function<void()> task);
    ~ThreadPool();

private:
    std::vector<std::thread> workers;
    std::queue<std::function<void()>> tasks;
    std::mutex queueMutex;
    std::condition_variable condition;
    bool stop;
};
```

7. 异常处理模块

异常处理类（`ExceptionHandler`）

```cpp
class ExceptionHandler {
public:
    static void handleException(const std::exception& e);
    static void logException(const std::exception& e);
};
```