**Software Design Specification (SDS)**

Revision History: (Double-Bloom for unifications)

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | | Description |
| 2019.4.8 | Renxiang Zhu | | The sds document for server service |
| 2019.4.8 | Zhi Zhou | | The Detailed design for server |
| 2019.4.10 | Wenniu Zhang | | The sds document for IC service |
| 2019.4.11 | Yifan Zhang | | Test cases document for ICservice |
| 2019.4.8 | Shuihan Zhang | | Editing services provided (database module) |
| 2019.4.8 | Rui Xing | | Editing access method (database module) |
| 2019.4.8 | Rui Zhu | | Editing access method effects (database module) |
| 2019.4.10 | Rui Xing | | Editing services (database module) |
| 2019.4.11 | Yuru Wang | | Editing ER chart (database module) |
| 2019.4.12 | Shijie Wen | | Editiong class chart (database module) |
| 2019.4.8 | Haohua Duan | The Client(Android) Detailed Design | |
| 2019.4.10 | Haohua Duan | The IS document for Android | |
| 2019.4.11 | Haohua Duan | Revise the Client (Android) Detailed Design | |
| 2019.4.11 | Haohua Duan | Revise the IS document for Android | |
| 2019.4.12 | Haohua Duan | Revise the SDS document for Android | |
| 2019.4.12 | Zhi Zhou | Update Server Required & TC Names | |

Contents

[Software Design Specification (SDS) 1](#_Toc1184)

[1. Introduction 3](#_Toc22977)

[1.1. Intended Audience and Purpose 3](#_Toc11823)

[1.2. How to use the document 3](#_Toc4702)

[2. System Design 3](#_Toc27289)

[2.1. Context 3](#_Toc6148)

[2.2. Design Pattern 4](#_Toc24603)

[2.3. Architecture 4](#_Toc32443)

[2.3.1. <Component Diagram> 4](#_Toc18016)

[2.3.2. <Deploy Diagram> 4](#_Toc11352)

[3. Module Interface Design 5](#_Toc3435)

[3.1 The hardware build socket connection with server. 5](#_Toc5165)

[3.2 When actuator received a command. 5](#_Toc6435)

[4. Detailed Design 5](#_Toc14590)

[4.1. Server Detailed Design 5](#_Toc7656)

[4.1.1 Services Provided 6](#_Toc28587)

[4.1.2 Access Method 6](#_Toc20243)

[4.1.3 Access Method Effects 7](#_Toc28729)

[4.1.4 Services Required 8](#_Toc22108)

[4.2. Client (Web) Detailed Design 8](#_Toc28706)

[4.2.1 Services Provided 8](#_Toc882)

[4.2.2 Access Method 8](#_Toc6157)

[4.2.3 Access Method Effects 9](#_Toc23881)

[4.2.4 Services Required 9](#_Toc30814)

[4.3. Client (Android) Detailed Design 10](#_Toc19181)

[4.3.1 Service Provided 10](#_Toc31558)

[4.3.2 Access Method 10](#_Toc30683)

[4.3.3 Access Method Effects 11](#_Toc2094)

[4.3.4 Services Required 12](#_Toc30765)

[4.4. Intelligent Control Detailed Design 12](#_Toc27580)

[4.4.1 Services Provided 13](#_Toc4892)

[4.4.2 Access Method 14](#_Toc28867)

[4.4.3 Access Method Effects 15](#_Toc3753)

[4.4.4 Services Required 15](#_Toc14806)

[4.5 Database Detailed Design 16](#_Toc28007)

[4.5.1 ER chart 16](#_Toc10330)

[4.5.2 Class chart 17](#_Toc22462)

[4.5.3 Services 18](#_Toc17741)

[A.    Appendices 23](#_Toc29727)

[A.1    Definitions and acronyms 23](#_Toc283)

[A.1.1    Definitions 24](#_Toc19969)

[A.1.2    Acronyms and abbreviations 24](#_Toc2866)

[A.2    References 24](#_Toc10656)

## Introduction

## Intended Audience and Purpose

This document is write for intelligent control light system, in detail, for software design specification. Both user and developers can read this document for their desire. For users, the document is used for check functions for each module. For developers, this document is used for clarify the border of each module, as well as their fundamental design guidance.

The document includes operating system,

## How to use the document

The document includes system design (Section 2), which is consisted with programming language, operating system, internet protocol, (Section 2.1), as well as design pattern(2.2), architecture(2.3) and deploy diagram.

For section 3, it declare the interface for hardware connection. If you are looking for method to connect with hardware, you can see this part for details.

Besides, section4 is most detailed part for developers to look for functions and methods in each module. This module includes five basic part: web server(4.1), client for Android(4.2), client for web(4.3), Intelligent Control(4.4) and database(4.5). Each part is basically consisted by service provided, access method, access method effects, service required. Additionally, some chart will be included as well.

In the end, you may see the appendices(A), for definitions and acronyms(A.1) and references(A.2).

## System Design

## Context

Programming languages:python

Operating system:

Linux:

* Red Hat Enterprise Linux 5.5+1, 6.x (32-bit), 6.x (64-bit)2
* Red Hat Enterprise Linux 7.x (64-bit)2 (8u20 and above)
* Ubuntu Linux 12.04 LTS, 13.x
* Ubuntu Linux 14.x (8u25 and above)
* Ubuntu Linux 15.04 (8u45 and above)
* Ubuntu Linux 15.10 (8u65 and above)

Windows:

* Windows 10 (8u51 and above)
* Windows 8.x (Desktop)
* Windows 7 SP1
* Windows Vista SP2
* Windows Server 2008 R2 SP1 (64-bit)
* Windows Server 2012 and 2012 R2 (64-bit)

Mac OS X:

* Intel-based Mac running Mac OS X 10.8.3+, 10.9+

Internet protocol:IPV4

## Design Pattern

<Specifies the technical details of the software system: i.e., model-view-control division, restful service pattern, etc.>

## Architecture

## <Component Diagram>

The whole system is divided into five parts according to the SRS document, known as the Intelligent Control(IC), the Server, the Database(DB), the Web Client and the App Client. The server part mainly deals with the interaction functions with hardware, intelligent control modules, applications and databases. Its role can be considered as a transit station, as a hub for various information processing and interaction. According to the SRS file, the server mainly provides six services: (1) stable link between hardware (2) receiving hardware response (3) accepting hardware status data (4) accepting end user requirements (5) responding to end user query (6) End user access to the database to provide access.

There are totally 5 ways for other parts to get access to the server, which are server\_report, server\_register, server\_query, server\_command, server\_DB. All the details are implemented in the fourth quarter.

## <Deploy Diagram>

None temporarily.

## Module Interface Design

### 3.1 The hardware build socket connection with server.

When hardware is running, it should try to build sockets with server, in the following steps:

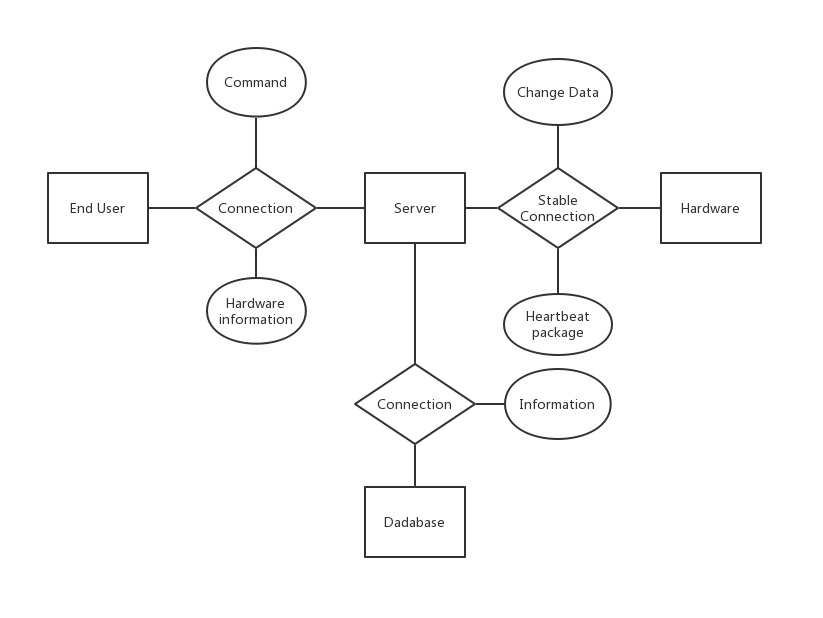
1. Try to build a reporting socket with server.(If the hardware is a actuator, it should build another receiving socket for receiving command.)
2. When socket is built, the hardware should sent a package to server using this socket for authenticating it’s identification.
3. The hardware will received a package in form of ‘{‘status’:0, ‘msg’:’...’}’ indicating whether the socket is admitted by the sever.

### 3.2 When actuator received a command.

When actuator received a command from server and this command let the actuator’s state changed. The actuator should report this command with its up-to-date state to server using the ‘server\_report’ method through the socket.

## Detailed Design

## Server Detailed Design



### 4.1.1 Services Provided

|  |  |  |  |
| --- | --- | --- | --- |
| # | Service | Provided By | Tested By |
| 1 | Hardware want to build stable connections with server. | server\_register | Server - T1 -5.1 |
| 2 | Hardware report data to server when it changes. | server\_report | Server - T3 -TC 5.3 |
| 3 | Hardware send heartbeat package to server. | server\_report |  |
| 4 | End user want to query hardware’s information. | server\_query, DB\_checkAuthority | Server - T2 -TC 5.2 |
| 5 | End user want to send command to hardware. | server\_command, DB\_queryHardware, DB\_checkAuthority | Server - T2 -TC 5.2 |
| 6 | End user want to communicate with DB.(When it comes to something about user, building, room and so on.) | server\_DB, DB’s other functions. | Server - T4 -TC 5.4 |

### 4.1.2 Access Method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| server\_report | Hardware use the socket which built in register stage to send data. | data | Text in JSON format | The content of data could be different when hardware want to report its data or send heartbeat package. |  | 2, 3 |
| server\_register | Hardware built a socket with server on ‘IP:443’. And then send data. | data | Text in JSON format | The JSON format data should contains the following fields:  ‘hid’: Hardware’s unique ID.  ‘type’: The type of this socket. (Report / Receive)  ‘auth’: The authenticating key which will be confirmed by server. | Wrong hardware’s ID or authenticating key. | 1 |
| server\_query | Send post/get request to ‘IP:80/api/hardware’ | data | HTTP request parameters package | The JSON format data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. | The user don’t have authority to access the hardware. | 4 |
| server\_command | Send post/get request to ‘IP:80/api/command’ | data | HTTP request parameters package | The JSON format data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID.  ‘cmd’: The user command. | The user don’t have authority to access the hardware. | 5 |
| server\_DB | Send post/get request to ‘IP:80/interface/<type>/<task>’ | data | HTTP request parameters package | The data should contain everything that the interface of DB need. | According to the DB’s response. | 6 |

### 4.1.3 Access Method Effects

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| server\_report | If the received JSON data is empty which indicating that hardware sent a heartbeat package, the server will update the hardware’s last updating time to keep it in a online state.  If the received JSON only contains ‘data’ filed indicating that the sensor want to report it latest data, the server will record the data in the RAM or Redis.  If the received JSON data contains ‘data’ and ‘cmd’ filed which indicating that the actuator’s state is change by a command, the server will record it’s latest data and the latest efficient command in RAM or Redis. |
| server\_register | The server will check whether the authenticating key is correct or not.  After, the server will check the hardware’s identification according to the the information which got from DB.  The server will build a report / receive socket according to the filed ‘type’. |
| server\_query | The server will check whether the user has authority to access this hardware.  Next, the server will request hardware’s persistent information like it’s nickname, type and so on from DB.Combining the real-time data which stored in server, the server will return this to clients. |
| server\_command | The server will check whether the user has authority to access this hardware.  Next, server will ask the IC for command which need to be sent to hardware.  Finally, the server will send the command to hardware. |
| server\_DB | The server will redirect this request to DB’s API and return what DB response to client. |

### 4.1.4 Services Required

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| DB\_getUser | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. |  | 4, 5 |
| DB\_queryHardware | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. |  | 4 |
| DB\_getAllRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | None |  |  |
| DB\_getRoomByHID | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘hid’: The target hardware’s unique ID. |  |  |
| DB\_getSensorOfRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘rid’: The target room’s unique ID. |  |  |
| DB\_getDeviceOfRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘rid’: The target room’s unique ID. |  |  |
| DB’s other functions. | Send post/get request to ‘...’ | data | HTTP request parameters package | According to the functions of DB. |  | 6 |

## Client (Web) Detailed Design

### 4.2.1 Services Provided

|  |  |  |
| --- | --- | --- |
| Service | Provided By | Tested By |
| User login to the system through two text boxes, username and password, then click a button to submit it to the server | login | WEB\_T1 |
| User click the light’s or the light sensor’s icon to let the web list the light’s state or light sensor’s state and whether someone is in the room | loadLights,checkLightState | WEB\_T2 |
| User click a radio button to turn on/off the lights | turnOn, turnOff | WEB\_T3 |
| List the rooms in the window | loadRoom | WEB\_T2 |
| List the lights and light sensors in the window | loadLights | WEB\_T2 |
| List the buildings in the window | loadBuilding | WEB\_T2 |

### 4.2.2 Access Method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| Login | Username,password | String,String | All is String type | timeout | 1 |
| loadBuilding |  |  |  |  | 1 |
| loadRoom | buildID | Int | buildID is the id of the building whose rooms will be loaded |  | 2 |
| loadLights | roomID | Int | roomID is the id of the room whose lights and light sensors will be loaded |  | 1 |
| checkLightState | lightID | Int | lightID is the id of the light which will be checked |  |  |
| turnOn | lightID | Int | lightID is id of the light to be turned on |  |  |
| turnOff | lightID | Int | lightID is the id of light to be turned off |  |  |
| quit | void |  |  |  |  |

### 4.2.3 Access Method Effects

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| login | User type their username and password in the textbox, then the function encrypt the password and send them to the serve to check whether the user is valid. |
| LoadBuilding | After login successfully, the web will execute the function automatic, then the user can view the building list. |
| loadRoom | After selecting a building, the web will execute the function, then the user can view the rooms of the building |
| loadLights | After selecting a room, the web will execute the function, then the user can view the lights and light sensors in the room and whether someone is in the room |
| checkLightState | After selecting a light, the web will execute the function to display the light’s state |
| turnOn | Turn on the light |
| turnOff | Turn off the light |
| quit | Log out the login information about user |

### 4.2.4 Services Required

|  |  |  |
| --- | --- | --- |
| Service | Provided By | Tested By |
| login | login | WEB\_T1 |
| Check the state of lights or lightsensors or checks whether someone is in room | loadLights,checkLightState | WEB\_T2 |
| Turn on/off the lights | turnOn, turnOff | WEB\_T3 |
| View the list of room | loadRoom | WEB\_T2 |
| View the list of Sensors and lights | loadLights | WEB\_T2 |

## Client (Android) Detailed Design

### 4.3.1 Service Provided

|  |  |  |
| --- | --- | --- |
| Service | Provided By | Tested By |
| User input username and password into two different EditText to login | login | T1 |
| User input the roomID and the lightID into two different EditText to check the state of lights or lightsensors or checks whether someone is in room | loadLights,checkLightState | T2 |
| Through the Switch to turn on/off the lights | turnOn, turnOff |  |
| User input the buildID into a EditText to view the list of room | loadRoom |  |
| User input the roomID into a EditText to view the list of Sensors and lights | loadLights |  |
| User input the position of the building which is going to be added into a EditText to add a new building | AddBuilding |  |
| User input the buildID which is the id of a existing building into a EditText to delete a existing building | DelBuilding |  |

### 4.3.2 Access Method

The details about each access method are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| Login | Username,password | String,String | All is String type | timeout | 1 |
| loadBuilding |  |  |  |  | 1 |
| loadRoom | buildID | Int | buildID is the id of the building whose rooms will be loaded |  | 2 |
| loadLights | roomID | Int | roomID is the id of the room whose lights and light sensors will be loaded |  | 1 |
| checkLightState | lightID | Int | lightID is the id of the light which will be checked |  |  |
| turnOn | lightID | Int | lightID is id of the light to be turned on |  |  |
| turnOff | lightID | Int | lightID is the id of light to be turned off |  |  |
| quit | void |  |  |  |  |

### 4.3.3 Access Method Effects

Users are divided into two categories, namely students and administrators. There are six services offered, such as login, check the state of lights or light sensors or checks whether someone is in room, turn on/off the lights, view the list of room, view the list of Sensors and lights. And there are eight access methods, the details are as follows:

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| login | User type their username and password in the textbox, then the function encrypt the password and send them to the serve to check whether the user is valid. |
| LoadBuilding | After login successfully, the web will execute the function automatic, then the user can view the building list. |
| loadRoom | After selecting a building, the web will execute the function, then the user can view the rooms of the building |
| loadLights | After selecting a room, the web will execute the function, then the user can view the lights and light sensors in the room and whether someone is in the room |
| checkLightState | After selecting a light, the web will execute the function to display the light’s state |
| turnOn | Turn on the light |
| turnOff | Turn off the light |
| quit | Log out the login information about user |

This module interacts with other modules via json packets.

### 4.3.4 Services Required

|  |  |  |
| --- | --- | --- |
| Service | Provided By | Tested By |
| login | login | T1 |
| Check the state of lights or lightsensors or checks whether someone is in room | loadLights,checkLightState | T2 |
| Turn on/off the lights | turnOn, turnOff |  |
| View the list of room | loadRoom |  |
| View the list of Sensors and lights | loadLights |  |
| Add a new building | AddBuilding |  |
| Delete a existing building | DelBuilding |  |

## Intelligent Control Detailed Design

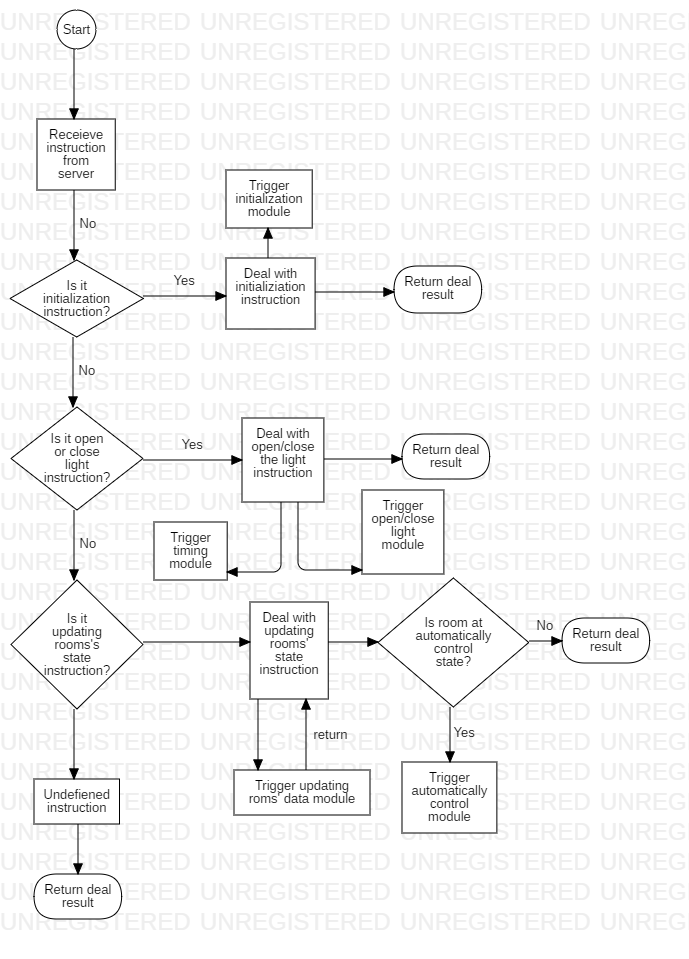


Diagram 4.4.1 Flow Chart for Intelligent Control Module

### 4.4.1 Services Provided

|  |  |  |  |
| --- | --- | --- | --- |
| # | Service | Provided By | Tested By |
| 1 | The server request an command for Intelligent Controller. | Analysis\_check\_data, Judge\_command, Return\_information\_and\_errors | TC 1 |
| 2 | Hardware report data to server when it changes. | IC\_initialize | TC 2 |
| 3 | The server update room information for Intelligent Controller. | IC\_update | TC 3 |

### 4.4.2 Access Method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| Analysis\_check\_data | Server makes a request for user’s command. Server send data to IC, IC deploy the data to find its correction. | data | Text in JSON format | The json format data should contains the following fields: ‘auth: 1(1 for student, 2 for teacher).  ‘camera’: True(True for someone in the room. False for opposite. )  ‘light’: on(The current state for light in room. )  ‘brightness’ : 234(The brightness in the room)  ‘button\_pressed’: true(If the button is pressed, this it would be true. )  ‘command’: on(To open or close the light, command is on or off) | The same packages are send for network delay. Or the data format is wrong. | 1, 2, 3 |
| Judge\_command | To decide whether accept command or not, this method judge whether this command can be executed. | Auth, camera, light, brightness, button\_pressed, command | Integer, bool, string, integer, bool, string | This method will run the rule diagram and decide whether this command can be executed or not. Before that, rule diagram have already existed. | Command parameters are less than excepted. | 1 |
| Return\_information\_and\_errors | Return to the sever whether the command can be accepted. | Judgement, information | String, String | If yes, return command to be executed. If no, return wrong information. |  | 1, 2, 3 |
| Set\_room\_inf | The server initialize the IC module. | data | Text in JSON format | The JSON format data should contains the following fields:  ‘rooms’: ‘’ (rooms of the system and their states. )  For each room, the JSON data should contain following context.  ‘camera’: True(True for someone in the room. False for opposite. )  ‘light’: on(The current state for light in room. )  ‘brightness’ : 234(The brightness in the room)  ‘button\_pressed’: true(If the button is pressed, this it would be true. ) | Wrong argument or missed this step. The update is so frequent that the bind width limits its transfer. | 2, 3 |
| Produce\_graph | Initialize rules graph for each time IC module starts. | data | Text in JSON format | The JSON format data should contains the following fields:  Have\_People (True or false),Authority\_Current (1 or 2), Authority\_Previous (1or2), Authority\_Current\_High (True or False), Room\_Light (integer), Button\_Pressed (True or false), Instruction (On or Off), Nobody\_For\_Set\_Time (True or false), Nobody\_Set\_Time (integer), No\_Instruction\_For\_Set\_Tim (True or false), Instruction\_Return (On or off) | The rules missed. | 2 |

### 4.4.3 Access Method Effects

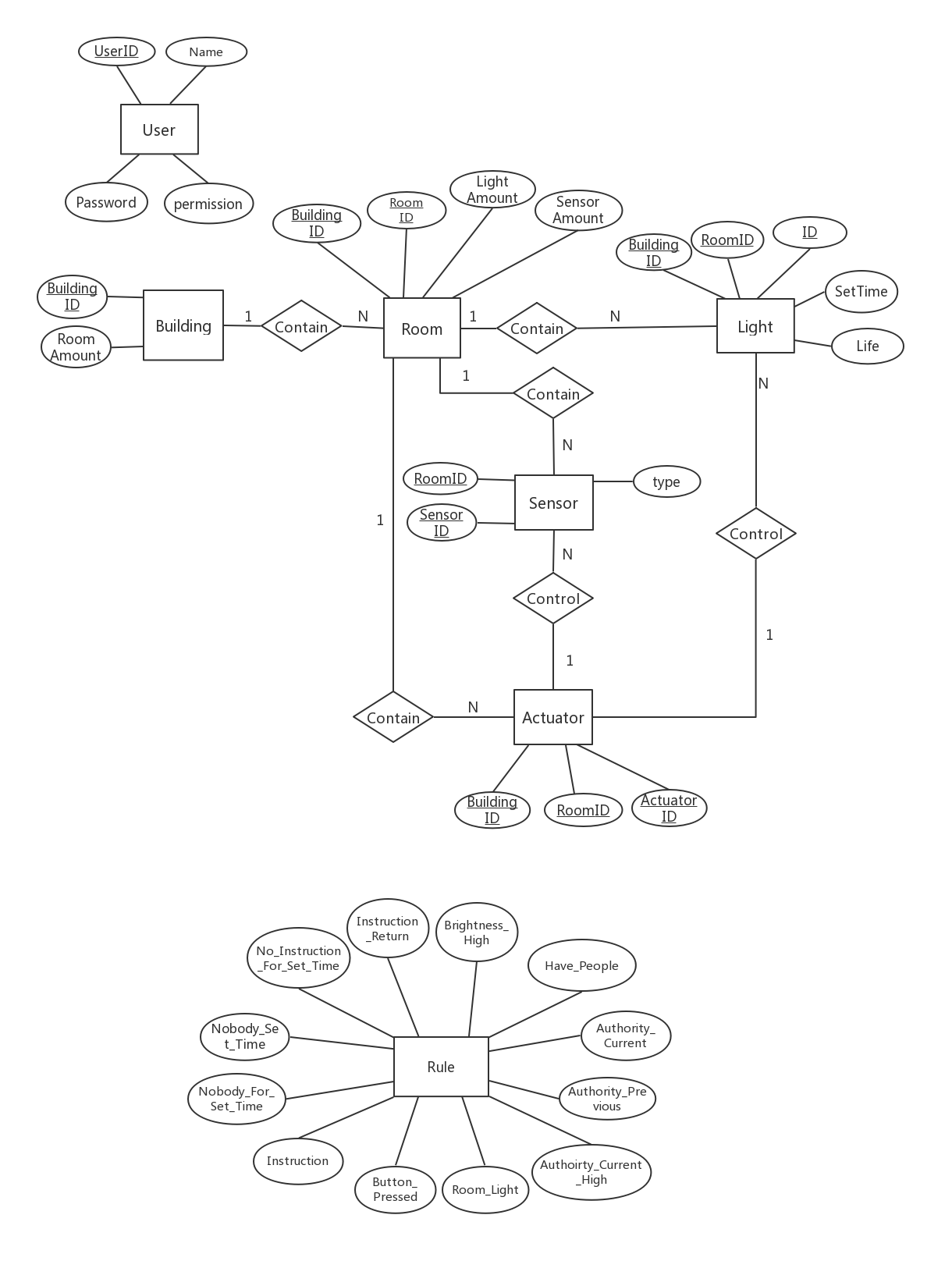
|  |  |
| --- | --- |
| **Access Method** | **Description** |
| Analysis\_check\_data | Every time the server put data to IC module, IC will use this method to check the data is in right format or not. If this method judge its error, the error message will be returned. |
| Judge\_command | Every time the server use IC module, it will send instruction to IC module , the IC module will deploy this instruction and judge whether it is a right instruction or not. If this instruction is in right format, IC will execute it, or IC returns error message.  Each time, the server should send a package of an room and its command, as well as the commander and information of the room to be tested. For this command, only the user with authority to access room can send its command. (This kind of filtration will be done in the server and database. )  The command is divided by the authority, that is student < teacher = administrator. If one command effects, another command should wait for at least one minute to change light state again. |
| Return\_information\_and\_errors | For each service, this method will be reused. For request, it will returns whether accept command or not. For initialize service, it will return the message about initialization. For updating, it will returns the rooms which state should be changed. |
| Set\_room\_inf | This method will refresh rooms’ state in IC memory. At the same time, if some room’s state should be changed, it will be reported to the service. |
| Produce\_graph | The server will initialize the IC module for rules. The rules are saved in the database, the server will fetch rules for IC module. |

### 4.4.4 Services Required

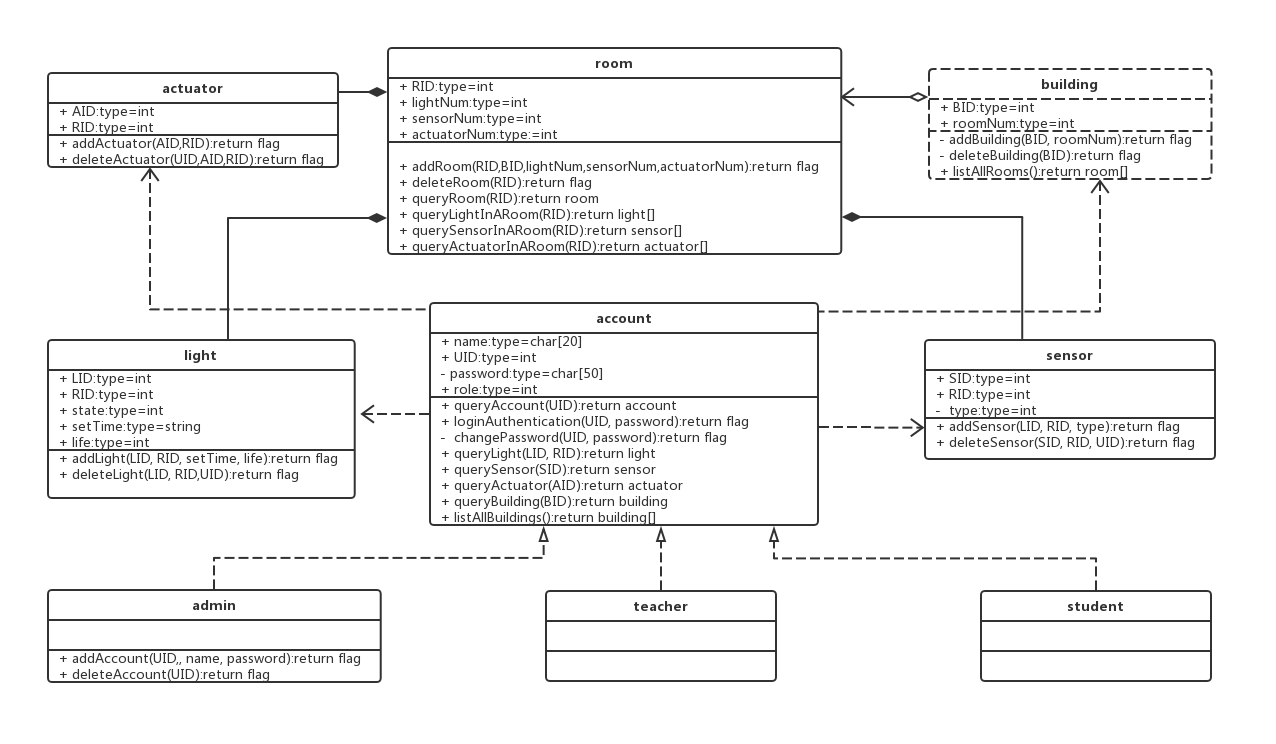
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| Server\_initialize\_IC | Initialize the rules and rooms’ state. | data | HTTP request parameters package | The JSON format data should contains the following fields: ‘auth’: 1(1 for student, 2 for teacher).  ‘camera’: True(True for someone in the room. False for opposite. )  ‘light’: on(The current state for light in room. )  ‘brightness’ : 234(The brightness in the room) |  | 2 |
| DB\_get\_rules | Database get rules of IC module. | data | HTTP request parameters package | The data should contains the following fields:  Have\_People (True or false),Authority\_Current (1 or 2), Authority\_Previous (1or2), Authority\_Current\_High (True or False), Room\_Light (integer), Button\_Pressed (True or false), Instruction (On or Off), Nobody\_For\_Set\_Time (True or false), Nobody\_Set\_Time (integer), No\_Instruction\_For\_Set\_Tim (True or false), Instruction\_Return (On or off) |  | 2 |
| Server\_update\_to\_IC | Updates the rooms’ state to IC. | data | HTTP request parameters package | Each time, one room’s state will be submitted. |  | 3 |

## 4.5 Database Detailed Design

### 4.5.1 ER chart



### 4.5.2 Class chart



### 4.5.3 Services

#### 4.5.3.1 Services Provided

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Service** | **Provided By** | **Tested By** |
| 1 | The server wants to register a non-existent account before for end users. | addAccount | T1 |
| 2 | The end user wants to register a new account and fill in his/her personal information. | deleteAccount | T2 |
| 3 | The server wants to query existent accounts. | queryAccount | T3 |
| 4 | The server would like to change user’s password. | changePassword | T4 |
| 5 | The server would like to search for username and password. | loginAuthentication | T1-T23 |
| 6 | The server wants to change one’s role. | changeUserRole | T5 |
| 7 | The server wants to know all users’ accounts. | listAllAccounts | T6 |
| 8 | The server wants to query for information about a light in a room. | queryLight | T7 |
| 9 | The server wants to query for information about all lights in a room. | queryLightInARoom | T8 |
| 10 | The server wants to add new lights to the list of lights he or she can control. | addLight | T9 |
| 11 | The server would like to delete some lights from light table. | deleteLight | T10 |
| 12 | The server wants to query for information about a sensor in a room. | querySensor | T11 |
| 13 | The server wants to query for information about all sensors in a room. | querySensorInARoom | T12 |
| 14 | The server wants to add new sensors to the list of sensors he or she can control. | addSensor | T13 |
| 15 | The server would like to delete some sensors from sensor table. | deleteSensor | T14 |
| 16 | The server wants to query for information about a room. | queryRoom | T15 |
| 17 | The server wants to add new rooms to the list of rooms he or she can control. | addRoom | T16 |
| 18 | The server would like to delete some rooms from room table. | deleteRoom | T17 |
| 19 | The server would like to get information about all rooms. | listAllRooms | T18 |
| 20 | The server wants to query for information about an actuator. | queryActuator | T19 |
| 21 | The server wants to query for information about all actuators in a room. | queryActuatorInARoom | T20 |
| 22 | The server would like to delete some actuators from actuator table. | deleteActuator | T21 |
| 23 | The server wants to add new actuators to the list of actuators he or she can control. | addActuator | T22 |
| 24 | The server wants to query for information about a building. | queryBuilding | T23 |
| 25 | The server wants to add new buildings to the list of buildings he or she can control. | addBuilding | T24 |
| 26 | The server would like to delete some buildings from building table. | deleteBuilding | T25 |
| 27 | The server would like to get information about all buildings. | listAllBuildings | T26 |

#### 4.5.3.2 Access Method

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Excep-tions** | **Map to services** |
| addAccount | UID, name, password | integer, string, string | UID is the integer which indicates user’s account number; name is string which shows user’s name and password is a string which represents user’s password. |  | 1 |
| deleteAccount | UID | integer | UID is the integer which indicates user’s account number. |  | 2 |
| queryAccount | UID | integer | UID is the integer which indicates user’s account number. |  | 3 |
| changePassword | UID, password | integer, string | UID is the integer which indicates user’s account number and password is a string which represents user’s new password. |  | 4 |
| loginAuthentication | UID, password | integer, string | UID is the integer which indicates user’s account number and password is a string which represents user’s password. |  | 5 |
| changeUserRole | UID, role | integer, integer | UID is the integer which indicates user’s account number and role is the integer which represents user’s role which means that he/she is a student, teacher, or administrator. |  | 6 |
| listAllAccounts | none | none | none |  | 7 |
| queryLight | LID, RID | integer, integer | LID is the integer which indicates the light's number in a room and RID is the integer which shows a room’s number. |  | 8 |
| queryLightInARoom | RID | integer | RID is the integer which shows a room’s number. |  | 9 |
| addLight | LID, RID, setTime, life | integer, integer, string, integer | LID is the integer which indicates the light's number in a room; RID is the integer which shows a room’s number; setTime is the integer which represents the installation time of the light and life is the integer which suggests the life of a light in hours. |  | 10 |
| deleteLight | LID, RID,  UID | integer,  integer,  integer | LID is the integer which indicates the light's number in a room; RID is the integer which shows a room’s number and UID is the integer which indicates user’s account number. |  | 11 |
| querySensor | SID | integer | SID is the integer which shows the sensor’s number. |  | 12 |
| querySensorInARoom | RID | integer | RID is the integer which shows a room’s number. |  | 13 |
| addSensor | LID, RID, type | integer, integer, integer | LID is the integer which indicates the light's number in a room; RID is the integer which shows a room’s number and type is the integer which suggests the type of sensor. |  | 14 |
| deleteSensor | SID, RID, UID | integer, integer, integer | SID is the integer which shows the sensor’s number; RID is the integer which shows a room’s number and UID is the integer which indicates user’s account number. |  | 15 |
| queryRoom | RID | integer | RID is the integer which shows a room’s number. |  | 16 |
| addRoom | RID, lightNum, sensorNum | integer, integer, integer | RID is the integer which shows a room’s number; lightNum is the integer which indicates the required number of lights in a room and sensorNum is the required number of sensors. |  | 17 |
| deleteRoom | RID | integer | RID is the integer which shows a room’s number. |  | 18 |
| listAllRooms | none | none | none |  | 19 |
| queryActuator | AID | integer | AID is the integer which indicates actuator’s number. |  | 20 |
| queryActuatorInARoom | RID | integer | RID is the integer which shows a room’s number. |  | 21 |
| deleteActuator | AID, RID, UID | integer, integer, integer | AID is the integer which indicates actuator’s number; RID is the integer which shows a room’s number and UID is the integer which indicates user’s account number. |  | 22 |
| addActuator | AID, RID | integer, integer | AID is the integer which indicates actuator’s number and RID is the integer which shows a room’s number. |  | 23 |
| queryBuilding | BID | integer | BID is the integer which shows a building’s number. |  | 24 |
| addBuilding | BID, roomNum | integer, integer | BID is the integer which shows a building’s number and roomNum is the integer which indicates the required number of lights in a room. |  | 25 |
| deleteBuilding | BID | integer | BID is the integer which shows a building’s number. |  | 26 |
| listAllBuildings | none | none | none |  | 27 |

#### 4.5.3.3 Access Method Effects

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| addAccount | The database adds a new user’s information to the client table and update other tables. |
| deleteAccount | The database delete the target user account through his/her ID number and then update other tables. |
| queryAccount | User can query account by using this function such as his/her name, ID number and identify (teacher, student or administrator). |
| changePassword | User can change password by using this function. The database saves the encrypted password into the password property and update other tables through provided new password parameter. |
| loginAuthentication | According to the user ID, database finds out corresponding user item and determine whether the password is the same. If the user ID and password are correct, return the flag of success. |
| changeUserRole | Administrator can change user’s identity by using this function. Administrator provide user ID and the modified identity and database change the identity according to the user’s ID. |
| listAllAccounts | User can see all the users’ account information by using this function. Database return all the users’ account information to the user. |
| queryLight | User can query the light information by using this function. User provide LID and database return the RID where the light is. |
| queryLightInARoom | User can query the light in a room by using this function. User provide RID and database return the LID which is the ID of the light that is in this room. |
| addLight | Administrator can add light by using this function. Administrator provide light's ID, RID, setTime, and life and database store this information and update other tables. |
| deleteLight | User can delete light by using this function. User provide light ID, room ID and user ID. The database determines the current user’s role and judge whether he/she has the permission. According to the light ID and room ID, database finds out target light. Then database remove the target light and update the other table. |
| querySensor | User can query sensor information by using this function. User provide sensor SID and database return the other information in the table to the user from database. |
| querySensorInARoom | User can query the sensor in a room by using this function. User provide RID and database return the SID which is the ID of the light that is in this room. |
| addSensor | Administrator can add sensor by using this function. Administrator provide light's ID, RID, and type. and database store this information and update the other table. |
| deleteSensor | User can delete sensor by using this function. User provide sensor ID, room ID and user ID. The database determines the current user’s role and judge whether he/she has the permission. According to the sensor ID and room ID, database finds out target sensor. Then database remove the target sensor and update the other table. |
| queryRoom | User can query room information by using this function. User provide RID and database return the other information in the table to the user from database. |
| addRoom | Administrator can add room by using this function. Administrator provide RID, lightNum, and sensorNum and database store this information and update the other table. |
| deleteRoom | User can delete room by using this function. User provide room ID and user ID. The database determines the current user’s role and judge whether it can be deleted and find out target room. Then the database removes the target room and update the other table. |
| listAllRooms | User can see all the rooms’ information by using this function. Database return all the rooms’ information to the user. |
| queryActuator | User can query actuator information by using this function. User provide actuator AID and database return the other information in the table to the user from database. |
| queryActuatorInARoom | User can query the actuator in a room by using this function. User provide RID and database return the AID which is the ID of the actuator that is in this room. |
| deleteActuator | User can delete actuator by using this function. User provide actuator AID, room ID and user ID. The database determines the current user’s role and judge whether it can be deleted and find out target actuator. Then the database removes the target actuator and update the other table. |
| addActuator | Administrator can add light by using this function. Administrator provide the actuator AID and RID and database store this information and update the other table. |
| queryBuilding | User can query building information by using this function. User provide building ID and database return the other information in the table to the user from database. |
| addBuilding | Administrator can add building by using this function. Administrator provide building ID and the required number of rooms and database store this information and update the other table. |
| deleteBuilding | User can delete building by using this function. User provide building ID and user ID. The database determines the current user’s role and judge whether it can be deleted and find out target building. Then the database removes the target building and update the other table. |
| listAllBuildings | User can see all the buildings’ information by using this function. Database return all the buildings’ information to the user. |

#### 4.5.3.4 Services Required

*none*

# A.    Appendices

## A.1    Definitions and acronyms

### A.1.1    Definitions

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
|  |  |
|  |  |
|  |  |
|  |  |

### A.1.2    Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or**  **Abbreviation** | **Definitions** |
| IC | Intelligent Control |
|  |  |
|  |  |

## A.2    References