**Software Requirements Specification (SRS)**

Revision History:

|  |  |  |
| --- | --- | --- |
| Date | Author | Description |
| 2019.3.17 | Rui Xing | Editing system capabilities |
| 2019.3.18 | Shuihan Zhang | Editing system context |
| 2019.3.19 | Yuru Wang | Editing quality requirements (non-functional requirements) |
| 2019.3.19 | Zheng Chen | Introduction/Concept of Operation |
| 2019.3.20 | Rui Zhu | Editing fundamental assumptions |
| 2019.3.20 | Rui Xing | Editing expected subsets |
| 2019.3.21 | Rui Xing, Shuihan Zhang, Yuru Wang, Rui Zhu, Shijie Wen | Editing use cases |
| 2019.3.21 | Zheng Chen | Quality Requirements/Expected subsets |
| 2019.3.21 | Zhi Zhou | Overall block diagram |
| 2019.3.21 | Zimu Hu | Edit functional documentation |
| 2019.3.22 | Rui Xing, Shuihan Zhang, Yuru Wang, Rui Zhu, Shijie Wen | Editing use cases |
| 2019.3.22 | Zheng Chen | Behavioral Requirements |
| 2019.3.23 | Zhi Zhou | Modify functional documentation |
| 2019.3.23 | Zheng Chen | Use Cases/Behavioral Requirements |
| 2019.3.23 | Zheng Chen | Fundamental Assumption/Appendices |
| 2019.3.23 | Rui Xing, Shuihan Zhang, Yuru Wang, Rui Zhu, Shijie Wen | Adding use case |
| 2019.3.23 | Shijie Wen | Editing detailed requirements |
| 2019.3.23 | Zhang Hongfan | Introduction  Concept of Operation |
| 2019.3.23 | Zhang Hongfan | Behavioral Requirements  Expected subsets  Quality Requirements  Fundamental Assumptions  Expected Changes |
| 2019.3.23 | Rui Zhu | Editing expected changes |
| 2019.3.23 | Yuru Wang | Editing appendices |
| 2019.3.24 | Shijie Wen | Modifying detailed requirements |
| 2019.3.24 | Rui Xing | Editing introduction |
| 2019.3.25 | Zhi Zhou | Add Server System Context |
| 2019.3.25 | Zhi Zhou | Add System Input & Output |
| 2019.3.25 | Renxiang Zhu | Add Quality Requirements |
| 2019.3.25 | Renxiang Zhu | Integrate documents |
| 2019.3.25 | Yuanjin Li | Editing Software Requirements Specification |
| 2019.3.25 | Zhang Hongfan  Rui Raposo | User Case |
| 2019.3.26 | Zhang Hongfan  Rui Raposo | User Case |
| 2019.3.26 | Yifan Zhang | Editing the Detailed Requirments |
| 2019.3.26 | Zhongyu Wang | Editing the Quality Requirments |
| 2019.3.26 | Zheng Chen | Revise Use Cases and System Inputs and Outputs |
| 2019.3.26 | Qingzhong Chen | Revise Use Cases |
| 2019.3.27 | Zheng Chen | Revise Use Cases and Fundamental Assumption |
| 2019.3.28 | Zhi Zhou | Combine Learning Ducks’ Document |
| 2019.3.30 | Zhang Hongfan， Rui Raposo | User Case |
| 2019.3.31 | Zhi Zhou | Combine Revision History |
| 2019.4.1 | Zheng Chen | Remove some parts of administrator’s adding and moving functions and use cases. |
| 2019.4.1 | Yuanjin Li | Modify the Output |
| 2019.4.1 | Yifan Zhang | Modify the Input |
| 2019.4.1 | Yifan Zhang | Add the Definitions |
| 2019.4.1 | Yuanjin Li | Modify the use cases |
| 2019.4.1 | Rui Xing, Shuihan Zhang, Yuru Wang, Rui Zhu, Shijie Wen | Editing use cases |
| 2019.4.1 | Hongfan Zhang | Update User Case Diagrams |
| 2019.4.2 | Zimu Hu | Combine Double Bloom’s Document |
| 2019.4.2 | Zhi Zhou | Combine Apostle’ s Document |
| 2019.4.3 | Zheng Chen, Pedro | Revise Use Case for Customers and hardware. |
| 2019.4.3 | Zhi Zhou | Add catalogue and update user case of server. |
| 2019.4.8 | Zheng Chen | Revise System Context, Use Case for Customers and hardware and Revise System Input for Web App. |
| 2019.4.10 | Zheng Chen, Rui Raposo | Revise Use Case 2.3.1 according to Rui’s advice. |
| 2019.5.8 | Zhang Hongfan | Add user cases mentioned by Rui Zhang |
| 2019.5.8 | Zhang Hongfan | Update the Definitions |
| 2019.5.9 | Zheng Chen | Add Use Case 2.3.5-2.3.15 According To  New Requirements From Users And Revise Behavioral Requirements part. |
| 2019.5.9 | Zimu Hu | Add use case 2.4.6  Add 8.2 key technology |
| 2019.5.10 | Li Yuanjin | Second iteration |
| 2019.5.25 | Zheng Chen | 1.Add The Use Case: User checks the list of sensors and actuators(lights and sirens).  2.Revise The Use Case: Administrator changes priorities between roles.  3.Add A Use Case: Administrator wants to check the alarm logs |
| 2019.5.28 | Zimu Hu, Zhi Zhou | Rewrite Use Case 2.4.1-2.4.6 according to V2.0 |
| 2019.5.29 | Li Yuanjin | Modify the general use cases graph and use cases(delete priority checking) |
| 2019.5.30 | Li Yuanjin | Add alternate flow of some use case |
| 2019.5.30 | Li Yuanjin | Update the created time of use case |
| 2019.6.4 | Zheng Chen and Hongfan Zhang | Revise the part 2.3: Use Case for Customers |

**Catalogue**

1.  Introduction 7

2. System Capabilities 7

2.1. System Context 7

2.2. System capabilities 7

2.3. Use cases for Customers 8

2.3.1 User Wants to Login 9

2.3.2 User Wants to Quit the Application 10

2.3.3 User wants to checks building list or room list. 11

2.3.4 User checks the list of sensors and actuators(lights and sirens). 12

2.3.5 User checks the state of lights or light sensors or checks whether someone is in room. 13

2.3.6 User Wants to Turn the lights on/off 14

2.3.7 Administrator wants to add/remove buildings 15

2.3.8 Administrator wants to add a room to a building or remove rooms from a building 16

2.3.9 Administrator wants to add sensors to a room or remove sensors from a room. 17

2.3.10 Administrator wants to add actuators (lights and siren) to a room or remove actuators (lights and siren) from a room 18

2.3.11 Administrator changes priorities between roles 20

2.3.12 Administrator Wants to enable/disable sensors or to enable/disable actuators in a room or to setup time-out value or to define the default state of the lights when the system recovers from an emergency, for every room. 20

2.3.13 Administrator Wants to Turn Alarm off 22

2.3.14 Administrator Wants to See the List of Alarms 23

2.4. Use cases of Server 24

2.4.1 Allocate Raspberry PI Unique ID 25

2.4.2 Update Raspberry PI GPIO Settings and Value 26

2.4.3 Client Updates Data in Database 26

2.4.4 Client Queries Data in Database 27

2.4.5 Client Controls the Hardware 28

2.5. Use cases of Intelligent Controller 30

2.5.1 Initialize the system 30

2.5.2 Automatic control mode 31

2.5.3 Command-light mode 32

2.5.4 Setting mode 33

2.5.5 Rules setting mode 34

2.6 Use Cases of Database 38

2.6.1 Server Wants to Register an Account for End Users 38

2.6.2 Server Wants to Delete a User Account 39

2.6.3 Server Wants to Change a User’s Password 40

2.6.4 Server Wants Authentication of the User ID and Password 41

2.6.5 Server Wants to Add New Lights 42

2.6.6 Server Wants to Remove Lights from a Room 43

2.6.7 Server Wants to Add New Sensors 45

2.6.8 Server Wants to Remove Sensors from a Room 46

2.6.9 Server Wants to Add New Rooms 47

2.6.10 Server Wants to Remove Existing Rooms 48

2.6.11 Server Wants to Change the User's Permissions 49

2.6.12 Server Wants to Add New Actuators 50

2.6.13 Server Wants to Remove Actuators from a Room 51

2.7 Use Cases of Hardware 52

2.7.1 Sensors & Lights Wants to Send the Status 53

2.7.2 hardware sends signals and gets command 53

2.7.3 Server gets signals from communication module 54

1.1    Intended Audience and Purpose 55

1.2    How to use the document 56

3.    Detailed Requirements 56

3.1 System Inputs and Outputs for Customers 56

3.1.1 Inputs for Web 56

3.1.2 Outputs for Web 57

3.1.3 Inputs for APP 57

3.1.4 Outputs for APP 58

3.2 Detailed Output Behavior for Customers 58

3.2.1 For Web 58

3.2.2 For APP 59

3.3 System Inputs and Outputs for Developer 59

3.3.1 Inputs 59

3.3.2 Outputs 61

4.   Quality Requirements (Non-functional Requirements) 65

5. Expected Subsets 65

6.   Fundamental Assumptions 66

7.    Expected Changes 66

8.    Appendices 66

8.1    Definitions and acronyms 66

8.1.1    Definitions 66

8.1.2    Acronyms and abbreviations 66

8.2 key technology 67

8.2.1 socket instead of web 67

# 1.  Introduction

# 2. System Capabilities

## 2.1. System Context

Requires a system with a GUI display and browser because all of the operations are performed through a GUI and a browser.

The Web APP can run on browsers which are chrome or Firefox.

The Android APP can run on Android 4.0+.

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+

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)

## 2.2. System capabilities

Intelligent light control system Web APP is a web program that supports user interaction. On the web page, the user logins the account according to his personal ID and password, and then carries on the concrete operation to the intelligent light control system. Different kinds of users have different rights to intelligent light control system. There are three different permissions: students, teachers and administrators. The system functions are as follows:

1.User login. Users must be students, teachers or administrators of some schools.

2.Check the state of the light. All users have this permission.

3.Check whether a room is occupied. All three users have this permission.

4.Check the state of the light sensor. In this function, users can see the situation of ambient light.

5.Turn on/off the lights. Student users can only turn on the light when it is off and the classroom is occupied, and turn off the light when it is on and the classroom is empty. When the relevant operation cannot be carried out, a window will pop up to show the reasons: For example, *There are people in the classroom, so you cannot turn off the lights*. Teachers and administrators directly force the lights to be on/off. Students, teachers and administrators can operate the switch of a light or the main switch of all lights.

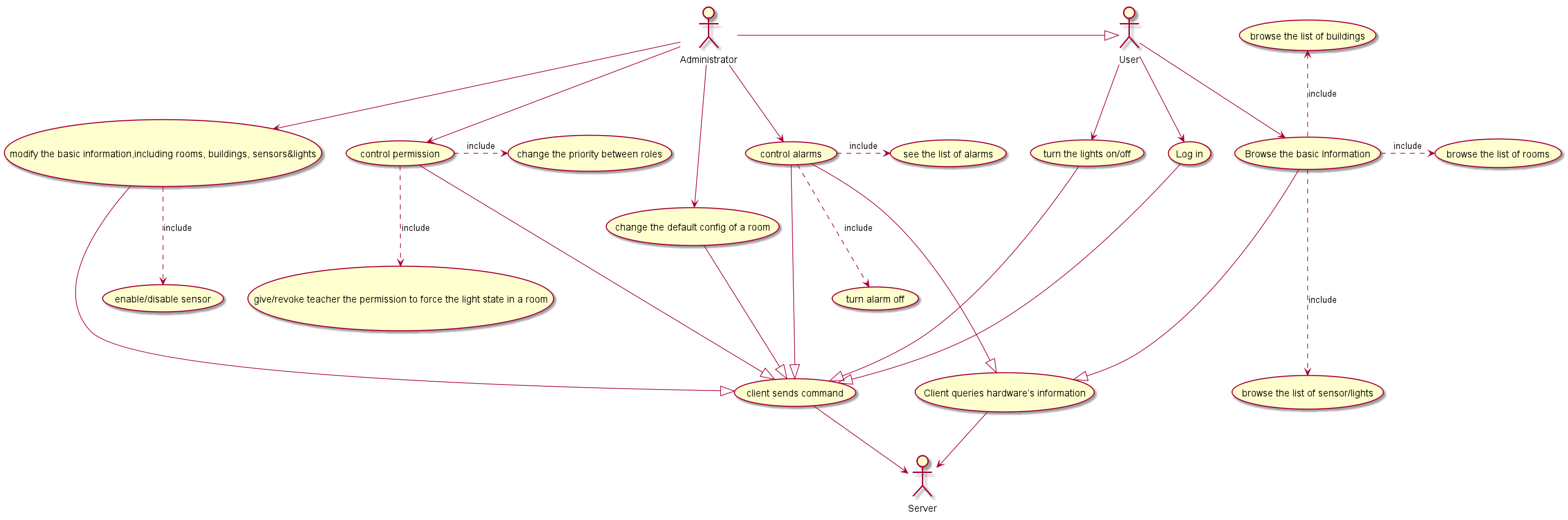
6.Add/delete new rooms. Administrators have this permission.

7.Add/delete sensors. Administrators have this permission. There are three kinds of sensors: switch sensor, light sensor and Presence sensor.

8.Add/delete actuators (lights). Administrators have this permission.

## 2.3. Use cases for Customers





### 2.3.1 User Wants to Login

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | user wants to login | | |
| Version | 1.0 | Created | 3-23-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | User login and go into the web application’s user interface. | | |
| Goals | User login and go into the web application’s user interface. | | |
| Summary | Login by inputting account number, password and press login button. | | |
| Actors | user | | |
| Trigger | Inputting account number, password and press login button. | | |
| Precondition | None | | |
| Basic Flow | Actor | | System |
| 1 | User(student, teacher and administrator)input account number and password. | |  |
| 2 | User press login button | |  |
| 3 |  | | system will process the answer from the server. If the login was successful the user will be sent to his homepage for web app or home screen for android app, otherwise the system will alert the user that his password or account is not correct. |
| 4 | User will get into the homepage or home screen, or will get the alert for wrong account information | |  |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The web page is displayed. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
|  |  | |  |

### 2.3.2 User Wants to Quit the Application

|  |  |  |
| --- | --- | --- |
| Use Case | User Wants to Quit the Application | |
| Version | 1.4 | |
| Author | Rui Raposo, Hongfan Zhang | |
| Source | Directly from Portuguese teacher | |
| Purpose | Quit | |
| Goals | Close the application and save the username and password | |
| Summary | Save the username and password, and terminate the application | |
| Actors | User | |
| Trigger | User presses “back” twice in two seconds. | |
| Precondition | The application is open and running. | |
| Basic Flow | User | System |
| 1 | Press “back” twice in two seconds. |  |
| 2 |  | Save username and password. |
| 3 |  | Terminates itself. |
| Exception Flows |  |  |
| 2.2 | User Forces shutdown (by shutting down their machine, using Android's Force Quit, etc.). |  |
| 3.2 |  | Do nothing. |
| Postconditions | If a user explored a room, app should save this room for “rooms” interface. | |
| Use case diagram |  | |

### 2.3.3 User wants to checks building list or room list.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | User checks building list or room list | | |
| Version | 1.0 | Created | 3-23-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | check building list or room list. | | |
| Goals | check building list or room list. | | |
| Summary | Check building list or room list. | | |
| Actors | User | | |
| Trigger | User Choose a teaching building/ a room | | |
| Precondition | Login  sensors” | | |
| Basic Flow | Actor | | System |
| 1 | User pressed *see all buildings* button or choose a teaching building from a menu. | |  |
| 2 |  | | UI will return a list of teaching buildings or rooms. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The check result of light are displayed. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.4 User checks the list of sensors and actuators(lights and sirens).

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | User checks building list or room list | | |
| Version | 1.0 | Created | 5-25-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | User checks the list of sensors and actuators(lights and sirens). | | |
| Goals | User checks the list of sensors and actuators(lights and sirens). | | |
| Summary | User checks the list of sensors and actuators(lights and sirens). | | |
| Actors | User | | |
| Trigger | User chooses a building from buildings’ drop-down menu and then chooses a room from rooms’ drop-down menu. | | |
| Precondition | Login  sensors” | | |
| Basic Flow | Actor | | System |
| 1 | User chooses a building from buildings’ drop-down menu and then chooses a room from rooms’ drop-down menu. | |  |
| 2 |  | | UI part will send building name, room number and command to server. |
| 3 | UI will get a list of sensors and actuators (lights and sirens). | |  |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The check result of light are displayed. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.5 User checks the state of lights or light sensors or checks whether someone is in room.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | User checks the state of lights or light sensors or checks whether someone is in room | | |
| Version | 1.0 | Created | 3-23-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | check the state of lights or light sensors or check whether someone is in room | | |
| Goals | check the state of lights or light sensors or check whether someone is in room | | |
| Summary | Check all states of lights and sensors and whether someone is in room by choosing room number and choosing teaching building. | | |
| Actors | user | | |
| Trigger | choosing room number and choosing teaching building | | |
| Precondition | Login and press “lights and  sensors” | | |
| Basic Flow | Actor | | System |
| 1 | User chooses teaching building name and room number from drop-down list and press enter button. | |  |
| 2 |  | | To server: UI part will send account number, token(for identity validation of connection with server), room number, teaching building and checking command. |
| 3 | The user checks result. | |  |
| 4 |  | | The server return lights' and light sensors' information, switch sensors and presence sensors’ information and whether someone is in room. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The check result of light are displayed. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.6 User Wants to Turn the lights on/off

|  |  |  |
| --- | --- | --- |
| Use Case | User Wants to Turn the lights on/off | |
| Version | 1.0 | |
| Author | Rui Raposo, Hongfan Zhang | |
| Source | Directly from Portuguese teacher | |
| Purpose | Turn the lights on/off. | |
| Goals | Turn the lights on/off. | |
| Summary | Upload commands to server and receive the result of commands. | |
| Actors | User | |
| Trigger | User click specific switch in “sensors&lights” tab. | |
| Precondition | The application is open and running. User is logged. User has chosen a building. User has chosen a room. | |
| Basic Flow | User | System |
| 1 | Click specific switch in rooms interface. |  |
| 2 |  | Upload commands to the server. |
| 3 |  | Receive the result of commands. |
| 4 |  | Display the result of commands. |
| Exception Flows |  |  |
| Postconditions | None | |
| Use case diagram |  | |

### 2.3.7 Administrator wants to add/remove buildings

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Administrator wants to add/remove buildings | | |
| Version | 1.0 | Created | 2019-5-8 |
| Author | Zheng Chen | | |
| Source | New requirements from users | | |
| Purpose | Administrator adds/removes buildings | | |
| Goals | Administrator adds/removes buildings | | |
| Summary | Administrator adds building by inputting buildings’ information. Administrator removes buildings by choosing buildings. | | |
| Actors | Administrator | | |
| Trigger | Administrator presses *add buildings* button or presses *remove buildings* button. | | |
| Precondition | Administrator logins | | |
| Basic Flow | Actor | | System |
| 1 | Administrator presses *add buildings*  buttonor presses *remove buildings* button. | |  |
| 2 |  | | If *remove buildings* button is pressed, UI will update the page with several buildings’ information and let the administrator choose which buildings he/she wants to remove.  If *add buildings* button was pressed, UI  will let you input the new buildings’ information. |
| 3 | If *remove buildings* button is pressed, the administrator chooses buildings and presses *confirm* button.  If *add buildings* button is pressed, the administrator inputs the new buildings’ information and presses *confirm* button. | |  |
| 4 |  | | UI will send building information and command to server |
| 5 | Administrator gets the adding/removing result, such as “remove successfully!”. | |  |
| Type | Primary | | |
| Postconditions | None | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.8 Administrator wants to add a room to a building or remove rooms from a building

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Administrator wants to add/remove rooms | | |
| Version | 1.0 | Created | 2019-5-8 |
| Author | Zheng Chen | | |
| Source | New requirements from users | | |
| Purpose | Administrator adds/removes rooms | | |
| Goals | Administrator adds/removes rooms | | |
| Summary | Administrator adds a room to a building by inputting rooms’ information and choosing a teaching building. Administrator removes rooms by choosing rooms and choosing a building. | | |
| Actors | Administrator | | |
| Trigger | Administrator presses *add a room to a building* buttonor presses *remove rooms from a building* button. | | |
| Precondition | Administrator logins | | |
| Basic Flow | Actor | | System |
| 1 | Administrator presses *add a room to a building* buttonor presses *remove rooms from a building* button. | |  |
| 2 |  | | UI will update the page with a drop-down menu  and let the administrator choose one building. |
| 3 | If *add a room* button is pressed, the administrator will input room number and press *confirm* button.  If *remove rooms* button is pressed, the administrator will choose rooms and press *confirm* button. | |  |
| 4 |  | | UI will send adding/removing command, room number(s) and building name to the server. |
| 5 | Administrator gets the adding/removing result, such as “remove successfully!”. | |  |
| Type | Primary | | |
| Postconditions | None | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.9 Administrator wants to add sensors to a room or remove sensors from a room.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Administrator wants to add sensors to a room or remove sensors from a room | | |
| Version | 1.0 | Created | 2019-5-8 |
| Author | Zheng Chen | | |
| Source | New requirements from users | | |
| Purpose | Administrator adds/removes sensors | | |
| Goals | Administrator adds/removes sensors | | |
| Summary | Administrator adds sensors to a room by pressing *add sensors* button and inputting sensors’ information and choosing a building and a room. Administrator removes sensors by pressing *remove sensors* button and choosing a building a room and sensors. | | |
| Actors | Administrator | | |
| Trigger | Administrator presses *add sensors* buttonor presses *remove sensors* button. | | |
| Precondition | Administrator logins | | |
| Basic Flow | Actor | | System |
| 1 | Administrator presses *add sensors* buttonor presses *remove sensors* button. | |  |
| 2 |  | | UI will update the page with two drop-down menus and let the administrator choose one building and a room. |
| 3 | Administrator chooses a building and a room from two drop-down menus.  If *add sensors* button is pressed, the administrator inputs sensors’ information and presses *confirm* button.  If *remove sensors* button is pressed, administrator chooses sensors and finally presses *confirm* button. | |  |
| 4 |  | | UI will send adding/removing command, room number(s) and building name and sensor’s information to the server. |
| 5 | Administrator gets the adding/removing result, such as “remove successfully!”. | |  |
| Type | Primary | | |
| Postconditions | None | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.10 Administrator wants to add actuators (lights and siren) to a room or remove actuators (lights and siren) from a room

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Administrator wants to add actuators (lights and siren) to a room or remove  Actuators (lights and siren) from a room. | | |
| Version | 1.0 | Created | 2019-5-8 |
| Author | Zheng Chen | | |
| Source | New requirements from users | | |
| Purpose | Administrator adds/removes actuators | | |
| Goals | Administrator adds/removes actuators | | |
| Summary | Administrator adds actuators (lights and siren) to a room by pressing *add actuators* button and inputting actuators’ information and choosing a building and a room. Administrator removes actuators by pressing *remove actuators* button and choosing a building a room and actuators. | | |
| Actors | Administrator | | |
| Trigger | Administrator presses *add actuators* buttonor presses *remove actuators* button. | | |
| Precondition | Administrator logins | | |
| Basic Flow | Actor | | System |
| 1 | Administrator presses *add actuators* buttonor presses *remove actuators* button. | |  |
| 2 |  | | UI will update the page with two drop-down menus and let the administrator choose one building and a room. |
| 3 | Administrator chooses a building and a room from two drop-down menus.  If *add actuators* button is pressed, the administrator inputs actuators’ information and presses *confirm* button.  If *remove actuators* button is pressed, administrator chooses actuators and finally presses *confirm* button. | |  |
| 4 |  | | UI will send adding/removing command, room number(s) and building name and actuators’ information to the server. |
| 5 | Administrator gets the adding/removing result, such as “remove successfully!”. | |  |
| Type | Primary | | |
| Postconditions | None | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.11 Administrator changes priorities between roles

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Administrator changes priorities between roles | | |
| Version | 1.0 | Created | 2019-5-8 |
| Author | Zheng Chen | | |
| Source | New requirements from users | | |
| Purpose | Administrator changes priorities between roles | | |
| Goals | Administrator changes priorities between roles | | |
| Summary | Administrator changes priorities between roles by pressing *change priorities between roles* button, pressing one role’s *modify* button, inputting a new priority value. | | |
| Actors | Administrator | | |
| Trigger | Administrator presses *change user’s authority* button. | | |
| Precondition | Administrator logins | | |
| Basic Flow | Actor | | System |
| 1 | Administrator presses *change priorities between roles* button | |  |
| 2 | Administrator presses one role’s *modify* button | |  |
| 3 |  | | UI will let you input a priority value between 0 and 99. |
| 4 | Administrator inputs a priority value to change the role’s priority. | |  |
| 5 | Administrator gets this operation’s result, such as “operation successfully!”. | |  |
| Type | Primary | | |
| Postconditions | None | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.3.12 Administrator Wants to enable/disable sensors or to enable/disable actuators in a room or to setup time-out value or to define the default state of the lights when the system recovers from an emergency, for every room.

|  |  |  |
| --- | --- | --- |
| Use Case | Administrator Wants to enable/disable sensors or to enable/disable actuators in a room or to setup time-out value or to define the default state of the lights when the system recovers from an emergency | |
| Version | 1.0.1 | |
| Author | Hongfan Zhang | |
| Source | Rui Zhang | |
| Purpose | Enable/disable sensors or to enable/disable actuators in a room or to setup time-out value | |
| Goals | Enable/disable sensors or to enable/disable actuators in a room or to setup time-out value | |
| Summary | Administrator change some attributes of a room. | |
| Actors | Administrator | |
| Trigger | Administrator Wants to enable/disable sensors or to enable/disable actuators in a room or to setup time-out value or to define the default state of the lights when the system recovers from an emergency | |
| Precondition | The application is open and running. User log in as Administrator. | |
| Basic Flow | Administrator | System |
| 1 | Administrator Wants to enable/disable sensors or to enable/disable actuators in a room or to setup time-out value or to define the default state of the lights when the system recovers from an emergency |  |
| 2 |  | Get and display the information of rooms form servers |
| 3 | Setup |  |
| 4 |  | Ask administrator to confirm |
| 5 | Confirm |  |
| 6 |  | Send the new information of rooms to server |
| 7 |  | Get and display up-to-date information of the room |
| Exception Flows |  |  |
| 5 | Decide not to confirm |  |
| 6 |  | Return 2 |
| Postconditions | None | |
| Use case diagram |  | |

### 2.3.13 Administrator Wants to Turn Alarm off

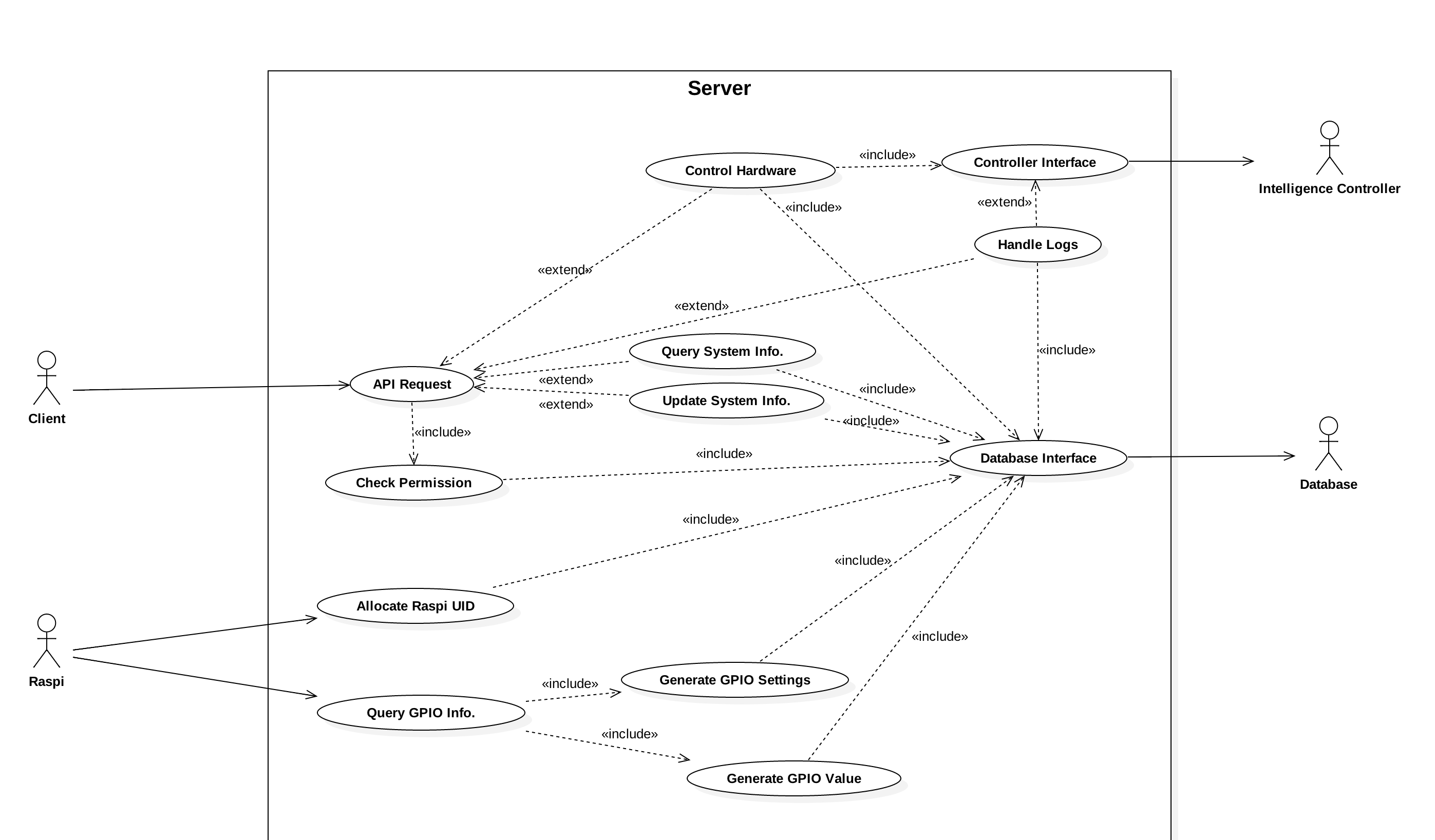
|  |  |  |
| --- | --- | --- |
| Use Case | Administrator wants to see what alarms are on and in which room the panic button was pressed | |
| Version | 1.0.1 | |
| Author | Hongfan Zhang | |
| Source | Rui Zhang | |
| Purpose | Turn alarm off. | |
| Goals | Turn alarm off. | |
| Summary | Administrator turn alarm off as he wants. | |
| Actors | Administrator | |
| Trigger | Administrator wants to turn alarm off. | |
| Precondition | The application is open and running. User log in as Administrator. There are alarms more than zero. | |
| Basic Flow | Administrator | System |
| 1 |  | Display the list of alarms and rooms where the panic button was pressed |
| 2 | Turn alarm off. |  |
| 3 |  | Ask administrator to confirm |
| 4 | Confirm |  |
| 5 |  | Send information about alarm-turn-off to server |
| 6 |  | Get and display new list of alarms and rooms where the panic button was pressed |
| Exception Flows |  |  |
| 4 | Decide not to confirm |  |
| 5 |  | Return 1 |
| Postconditions | None | |
| Use case diagram |  | |

### 2.3.14 Administrator Wants to See the List of Alarms

|  |  |  |
| --- | --- | --- |
| Use Case | Administrator Wants to See the List of Alarms | |
| Version | 1.0.1 | |
| Author | Hongfan Zhang | |
| Source | Rui Zhang | |
| Purpose | See the list of alarms. | |
| Goals | See the list of alarms. | |
| Summary | Administrator could see the list of alarms. | |
| Actors | Administrator | |
| Trigger | Administrator wants to see the list of alarms. | |
| Precondition | The application is open and running. User log in as Administrator. | |
| Basic Flow | Administrator | System |
| 1 | See the list of alarms |  |
| 2 |  | Get the list of alarms from server |
| 3 |  | Display the list of alarms |
| Exception Flows | None |  |
| Postconditions | None | |
| Use case diagram |  | |

## 2.4. Use cases of Server

This section is written for developer who wants to know the functions of server. The following graph describes the whole Use Cases of Server. And the following sections describe each parts of this graph.



### 2.4.1 Allocate Raspberry PI Unique ID

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Allocate Raspberry PI Unique ID | | |
| Version | V2.1 | Created | *2019.5.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Raspberry PI | | |
| Purpose | Request a unique ID from server | | |
| Goals | Connect to server and request server to give Raspi a unique ID | | |
| Summary | Raspi connect to server and request the allocate API to get a unique ID. | | |
| Actors | Raspberry PI | | |
| Trigger | When Raspi boot for the first time. | | |
| Precondition | Server is running | | |
| Basic Flow | *Actor* | | System |
| 1 | Request allocate API | |  |
| 2 |  | | Generate a unique ID under the help of database. |
| 3 |  | | Return unique ID to Raspi. |
| Frequency | Seldom | | |
| Type | Primary | | |
| Postconditions | Raspi got its unique ID. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
|  |  | |  |

### 2.4.2 Update Raspberry PI GPIO Settings and Value

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Update Raspberry PI GPIO settings and value | | |
| Version | V2.1 | Created | *2019.5.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Raspberry PI | | |
| Purpose | Update the setting and value of each GPIO on this Raspi. | | |
| Goals | Sync GPIO setting and value. | | |
| Summary | Report Raspi GPIO Value (For Sensor Port)  Update Raspi GPIO Setting  Update Raspi GPIO Value (For Device Port) | | |
| Actors | Raspberry PI | | |
| Trigger | Timer | | |
| Precondition | Server is online.  Raspi got its unique ID.  Raspi is online. | | |
| Basic Flow | *Actor* | | System |
| 1 | Report GPIO value (Sensor Port) | |  |
| 2 |  | | Return GPIO settings and GPIO value (Device Port) |
| 3 | Update GPIO settings and value | |  |
| Frequency | Frequently | | |
| Type | Primary | | |
| Postconditions | GPIO settings and value is synced. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
|  |  | |  |

### 2.4.3 Client Updates Data in Database

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Client updates data in database | | |
| Version | V2.1 | Created | *2019.5.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Client | | |
| Purpose | Update data in database | | |
| Goals | Update data in database, such as room, user, building, hardware and so on. | | |
| Summary | Client raise a request to update data.  After checking by server, the modification is applied. | | |
| Actors | Client | | |
| Trigger | Client request to update data. | | |
| Precondition | Server is running and client is login. | | |
| Basic Flow | *Actor* | | System |
| 1 | Send request to server | |  |
| 2 |  | | Check client’s permission. (Move to alternate flow 1 when failed.) |
| 3 |  | | Check request is valid or not. (Move to alternate flow 2 when failed.) |
| 4 |  | | Apply the modifications. |
| Frequency | Seldom | | |
| Type | Primary | | |
| Postconditions | Modifications are applied. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | | Tell client that the request need other permission. End. |
| 2 |  | | Tell client that the request is illegal. End. |

### 2.4.4 Client Queries Data in Database

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Client queries data in database | | |
| Version | V2.1 | Created | *2019.5.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Client | | |
| Purpose | Client queries some data in database. | | |
| Goals | Client queries some data in database, such as information of room, building, hardware and so on. | | |
| Summary | Client raise a request to update data.  After checking by server, the queries are answered. | | |
| Actors | Client | | |
| Trigger | Client raises a request. | | |
| Precondition | Server is running and client is login. | | |
| Basic Flow | *Actor* | | System |
| 1 | Raise a query request. | |  |
| 2 |  | | Check client’s permission. (Move to alternate flow 1 when failed.) |
| 3 |  | | Answer queries |
| Frequency | Seldom | | |
| Type | Primary | | |
| Postconditions | Client’s queries are answerd. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | | Tell client that the request need other permission. End. |

### 2.4.5 Client Controls the Hardware

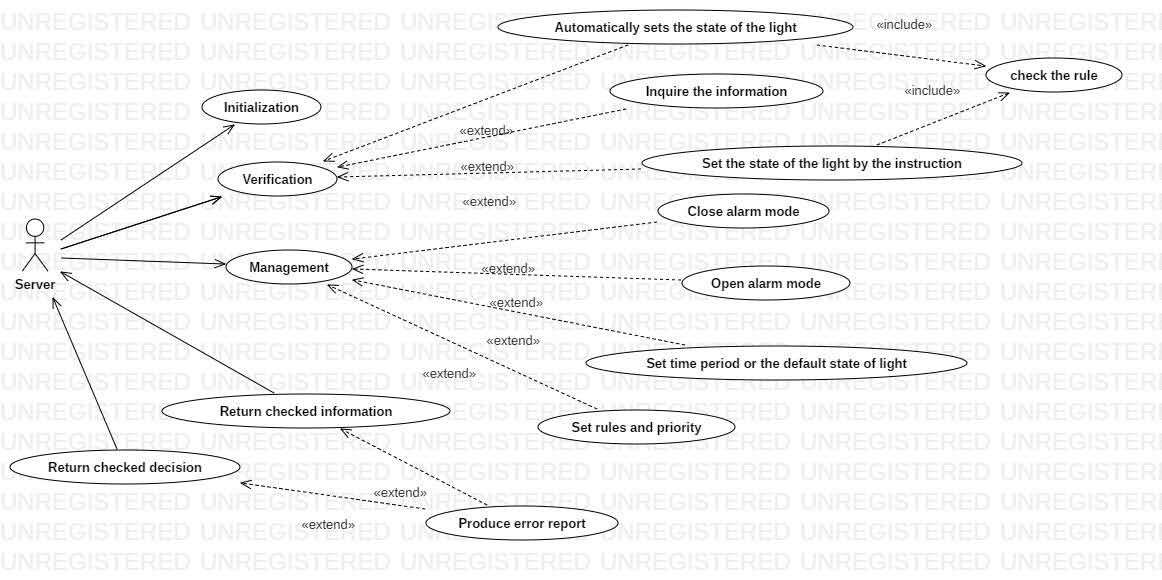
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Client control the hardware | | |
| Version | V2.1 | Created | *2019.5.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Client | | |
| Purpose | Client control the hardware. | | |
| Goals | Client turn on / off the light. | | |
| Summary | Server queries the result of command from Intelligence Controller and applied the command when needed. | | |
| Actors | Client | | |
| Trigger | Client raises a command request. | | |
| Precondition | Server is running and client is login. | | |
| Basic Flow | *Actor* | | System |
| 1 | Raise a command request. | |  |
| 2 |  | | Check client’s permission. (Move to alternate flow 1 when failed.) |
| 3 |  | | Query IC the result of this command |
|  |  | | Apply the command when needed and reply the clients. |
| Frequency | Seldom | | |
| Type | Primary | | |
| Postconditions | Command is apllied. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | | Tell client that the request need other permission. End. |

**2.4.6 Client Read / Solve Emergency Logs**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case  Case | Client Read / Solve Emergency Logs | | |
| Version | V2.1 | Created | *2019.05.28* |
| Author | Zhi Zhou, Zimu Hu | | |
| Source | Client | | |
| Purpose | Client get emergency logs or mark one log as solved. | | |
| Goals | Client get emergency logs or mark one log as solved. | | |
| Summary | Server check client’s permission and the give the logs to client or mark the log as solved. | | |
| Actors | Client | | |
| Trigger | Client raise a request. | | |
| Precondition | Server is running and client is login. | | |
| Basic Flow | *Actor* | | System |
| 1 | Client raise a log related request. | |  |
| 2 |  | | Check client’s permission. (Move to alternate flow 1 when failed.) |
| 3 |  | | Handler log tasks from clients and reply to client. |
| Frequency | Seldom | | |
| Type | Primary | | |
| Postconditions | The log is read by client or the log is marked as solved. | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | | Tell client that the request need other permission. End. |

## 2.5. Use cases of Intelligent Controller

This section is written for developer who wants to know the functions of intelligent controller.



### 2.5.1 Initialize the system

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Initialize the system | | |
| Version | 2.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | Initialize the system | | |
| Goals | Make the system start to work | | |
| Summary | Server give a signal and data package to make the system initialized. | | |
| Actors | Server | | |
| Trigger | Customer start the system | | |
| Precondition | None | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package to initialize the system | |  |
| 2 |  | | Initialization and sends a reply to server(If failed, move to alternate flow 2.1) |
| Frequency | Do it when customer want | | |  |
| Type | Primary | | |
| Chart | <<extend>>  Server  Initialization  Produce error report  Return checked information | | |
| Alternate Flow | *Actor* | | System |
| 2.1 |  | | sends an error report to server |

### 2.5.2 Automatic control mode

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Automatic control mode | | |
| Version | 3.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | Power saving intelligently | | |
| Goals | Control the status of the light Automatically | | |
| Summary | Automatically sets the state of the light. | | |
| Actors | Server | | |
| Trigger | None | | |
| Precondition | Automatic control mode | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a heartbeat data package. | |  |
| 2 |  | | IC verifies the situation (if be lack of sensor), checks rules then sends the command back to the server or sends an error report |
| Frequency | When the heartbeat data package comes once 30 minutes | | |  |
| Type | Primary | | |
| Chart  <<extend>>  Server  Produce error report  Return checked decisions  Check rules  <<include>>  Automatically sets the state of the light |  | | |
| Alternate Flow | *Actor* | | System |

### 2.5.3 Command-light mode

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Use Case | | Command-light mode | | | | |
| Version | | 3.1 | | Created | *2019-5-30* | |
| Author | | Li Yuanjin | | | | |
| Source | | Requirement | | | | |
| Purpose | | Turn the light on or off correctly by instruction | | | | |
| Goals | | Change the status of the light or give the error report | | | | |
| Summary | | A user issues an instruction to change the light through the server, then the Intelligent Control System (our system) make a judgement and return the result. | | | | |
| Actors | | Server | | | | |
| Trigger | | Someone gives an instruction to change the status of the light. | | | | |
| Precondition | | None | | | | |
| Basic Flow | | *Actor* | | | System | |
| 1 | | Server sends a data package which including instruction to change the state of the light  of the light | | |  | |
| 2 | |  | | | IC verifies the situation (if be lack of sensor), checks rules then sends the command or information back to the server | |
| Frequency | | Do it when server sends a data package | | | | |
| Type | | Primary | | | | |
| Chart  <<extend>>  Server  Produce error report  Return checked information and decisions  Check rules  <<include>>  Set the state of the light by the instruction | |  | | | | |
| Alternate Flow | | *Actor* | | | System | |

### 2.5.4 Setting mode

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Setting mode | | |
| Version | 3.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | (The administrator) Set the time period that during these time slots our system will keep the light on or off all the time, until a teacher’s or administrator’s command change the state. Or set the default state of the light. | | |
| Goals | Set the time period or set the default state of the light. | | |
| Summary | An administrator issues a command to change the time periods or set the default state of the light through the Server, then the Intelligent Control System (our system) make a judgement and return the results or the reason why he can’t do it. (IC should record the new rules) | | |
| Actors | Server | | |
| Trigger | A command to change the time periods | | |
| Precondition | The command must come from an administrator. | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package | |  |
| 2 |  | | Record the rule and send decision to Server |
| Frequency | Do it when server sends a data package | | |
| Type | Primary | | |
| Chart  <<extend>>  Server  Produce error report  Return checked information  Set time period or the default state of light |  | | |
| Alternate Flow | *Actor* | | System |

### 2.5.5 Rules setting mode

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Rules setting mode | | |
| Version | 3.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | (The administrator) Set the rules of our system, including permissions, priorities and the time of instruction coverage and shutdown time of light | | |
| Goals | Set the rules | | |
| Summary | A user issues a command to change the rules through the Server, then the  Intelligent Control System (our system) make a judgement and return the results or the reason why he can’t do it. (IC should record the new rules) | | |
| Actors | Server | | |
| Trigger | A command to set the rules. | | |
| Precondition | The command came from an administrator. | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package | |  |
| 2 |  | | Record the rules and send report to Server |
| Frequency | Do it when server sends a data package | | |
| Type | Primary | | |
| Chart | <<extend>>  Server  Produce error report  Return checked information  Set rules and priority | | |
| Alternate Flow | *Actor* | | System |

**2.3.6 Open the alarm mode**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Open the alarm mode | | |
| Version | 1.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement of the second iteration | | |
| Purpose | Turn on all the lights and sirens in the same building and keep the status until Administrator close the alarm mode | | |
| Goals | IC get the alarm status and refuse the other request and command in this building unless the administrator close the alarm mode. | | |
| Summary | Server sends a data packages (which should be including all ids of lights and sirens), IC record the status. | | |
| Actors | Server | | |
| Trigger | Someone presses the panic button | | |
| Precondition | None | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package | |  |
| 2 |  | | System records the status and sends a reply. |
| Frequency | Do it when server sends a data package | | |  |
| Type | Primary | | |
| Chart  <<extend>>  Server  Produce error report  Return checked information  Open alarm mode |  | | |
| Alternate Flow | *Actor* | | System |

**2.3.7 Close the alarm mode**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Close the alarm mode | | |
| Version | 1.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement of the second iteration | | |
| Purpose | Close the alarm mode and turn in to the normal situation. | | |
| Goals | Turn off all the siren in this building, record the time, turn off all the sirens and can receive the request from all user from now. | | |
| Summary | Server sends a data package (which should be including all ids of lights and sirens) to IC, IC record the time, turn off all the sirens and can receive the request from all user from now. | | |
| Actors | Server | | |
| Trigger | Administrator turn off the alarm | | |
| Precondition | None | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package | |  |
| 2 |  | | System record the time, turns off all the sirens and can receive the request from all user from now and send a reply to server. |
| Frequency | Do it when server sends a data package | | |  |
| Type | Primary | | |
| Chart  <<extend>>  Server  Produce error report  Return checked information  Close alarm mode |  | | |
| Alternate Flow | *Actor* | | System |

**2.3.8 Inquire the information**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Inquire the information | | |
| Version | 1.1 | Created | *2019-5-30* |
| Author | Li Yuanjin | | |
| Source | Requirement of the second iteration | | |
| Purpose | Administrator inquire the information of the system | | |
| Goals | Return the information or error report to server | | |
| Summary | Server sends a inquire package. By checking the permission, IC send the needed information or error report to server. | | |
| Actors | Server | | |
| Trigger | Administrator inquire the information of the alarm or panic button and so on | | |
| Precondition | None | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends a data package | |  |
| 2 |  | | System send the needed information or error report to server. |
| Frequency | Do it when server sends a data package | | |  |
| Type | Primary | | |
| Chart  <<extend>>  Server  Produce error report  Return checked information  Inquire the information |  | | |
| Alternate Flow | *Actor* | | System |

## 2.6 Use Cases of Database

This section is written for developer who wants to know the functions of database.

### 2.6.1 Server Wants to Register an Account for End Users

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Register an Account for End Users* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Xing, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server wants to register a non-existent account before. | | |
| Summary | The server wants to register a non-existent account before. And then the server calls the add account function. | | |
| Actors | Server | | |
| Trigger | The server calls the add account function. | | |
| Precondition | This account does not exist before registration; the application is open and running with a client book open. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the add account function, which provides the user's ID, name, identity, and new password. | |  |
| 2 |  | | The database adds personal information to the client table. |
| 3 |  | | Update other tables. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | There is a new user in the client table. It is marked to be saved at the next save point. The user book is aware that it has been altered. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the workflow. | |  |
| 1 |  | | The application returns to its initial state. |

### 2.6.2 Server Wants to Delete a User Account

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Delete a User Account* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Xing, Yuru Wang | | |
| Source | Customer | | |
| Goals | The end user wants to register a new account and fill in his/her personal information. This information should be added to the database. | | |
| Summary | The end user wants to register a new account and fill in his/her personal information. This information should be added to the database. And the server calls the delete account function. | | |
| Actors | Server | | |
| Trigger | The server calls the delete account function. | | |
| Precondition | The server wants to delete an existing user account. The information should be deleted from the database. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the delete account function, which provides the user's ID. | |  |
| 2 |  | | Retrieve the database by ID number and find the corresponding table items. |
| 3 |  | | Delete the target table entry. |
| 4 |  | | Update other tables. |
| 5 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The database removes the user's information and the account no longer exists. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user chooses to "cancel" the process. | |  |
| 1 |  | | The user's personal information will not be removed from the database. |
| 2 | The user that be searched does not exist. | |  |
| 2 |  | | Return the flag of not exist. |

### 2.6.3 Server Wants to Change a User’s Password

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Change a User’s Password* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Xing, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to change user’s password. | | |
| Summary | The server would like to change user’s password. And the server calls the change password function. | | |
| Actors | Server | | |
| Trigger | The server calls the change password function. | | |
| Precondition | The user has registered, that is, personal information and password already exist. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the change password function, which provides the user's ID and a new password. | |  |
| 2 |  | | The database looks up the corresponding table item according to the ID. |
| 3 |  | | The database saves the encrypted password into the password property. |
| 4 |  | | Update other tables. |
| 5 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | If the user saves the change, the password will be changed and the next time the server searches his/her password, it will get a new password. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user chooses to "cancel" the process. | |  |
| 1 |  | | The database will keep the original password of current user. |
| 2 | The user that be searched does not exist. | |  |
| 2 |  | | Return the flag of not exist. |

### 2.6.4 Server Wants Authentication of the User ID and Password

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants Authentication of the User ID and Password* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Xing, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to search for username and password. | | |
| Summary | The server would like to search for username and password. And the server calls the login authentication function. | | |
| Actors | Server | | |
| Trigger | The server calls the login authentication function | | |
| Precondition | The server transfers the user ID and password. | | |
| Basic Flow | *Actor* | | System |
| 1 | The server calls the login authentication function, which gives the user ID and password. | |  |
| 2 |  | | According to the user ID, database finds out corresponding user item. |
| 3 |  | | Determine whether the password is the same.  property. |
| 4 |  | | If the user ID and password are correct, return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The server receives the authentication result. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user that be searched does not exist. | |  |
| 1 |  | | Return the flag of not exist |
| 2 |  | | If the user ID and password are not correct, return the flag of error. |

### 2.6.5 Server Wants to Add New Lights

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to add new lights* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Zhu, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server wants to add new lights to the list of lights he or she can control. | | |
| Summary | The server calls the corresponding add function and transmits the information about the bulb that needs to be added. The database service program adds the light bulb to the data. | | |
| Actors | Server | | |
| Trigger | The server calls the add light function. | | |
| Precondition | User is an administrator; the application is open and running with a light book open. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the add light function, which provides the light's ID, roomID, settime, and Life. | |  |
| 2 |  | | The database adds light information to the light table. |
| 3 |  | | Update other tables. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | There is a new light in the light list. It is marked to be saved at the next save point. The light book is aware that it has been altered. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the workflow. | |  |
| 1 |  | | The light book he or she controls return to the initial state. |

### 2.6.6 Server Wants to Remove Lights from a Room

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Server Wants to Remove Lights from a Room | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Rui Zhu, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to delete some lights from light table. | | |
| Summary | The server calls the corresponding delete function and transmits the information about the bulb that needs to be deleted. The database service program deletes the light bulb to the data. | | |
| Actors | Server | | |
| Trigger | The server calls the delete light function. | | |
| Precondition | User is an administrator; the application is open and running with a light book open. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the delete light function, which gives the light ID, room ID and user ID. | |  |
| 2 |  | | According to the user ID, database determines the current user’s attribute and judge whether he has the permission. |
| 3 |  | | According to the light ID and room ID, database finds out target light. |
| 4 |  | | Remove the target light. |
| 5 |  | | Update the other table. |
| 6 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The database removes the target light and return the flag of result. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The current user has no authority to delete the light. | |  |
| 1 |  | | Return the flag of no permission. |
| 2 | The light that be searched does not exist. | |  |
| 2 |  | | Return the flag of not exist. |

### 2.6.7 Server Wants to Add New Sensors

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to add new sensors* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shijie Wen, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server wants to add new sensors to the list of sensors he or she can control. | | |
| Summary | The server calls the add sensor function and transmits the information about the sensors that needs to be added. The database service program adds the sensor to the sensor-list in database. | | |
| Actors | Server | | |
| Trigger | The server calls the add sensor function. | | |
| Precondition | User is an administrator; the application is open and running with a sensor book open. | | |
| Basic Flow | Actor | | System |
| 1 | Server calls add sensor functions, which provide the light's ID, roomID, and type. | |  |
| 2 |  | | The database adds sensor information to the light table. |
| 3 |  | | Update other forms. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | There is a new sensor in the sensor list. It is marked to be saved at the next save point. The sensor book is aware that it has been altered. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the workflow. | |  |
| 1 |  | | The sensor books he or she controls return to the initial state. |

### 2.6.8 Server Wants to Remove Sensors from a Room

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Remove Sensors from a Room* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shijie Wen, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to delete some sensors from sensor table. | | |
| Summary | The server calls the delete sensors function and transmits the information about the bulb that needs to be deleted. The database service program deletes the sensor bulb to the data. | | |
| Actors | Server | | |
| Trigger | The server calls the delete sensor function. | | |
| Precondition | User is an administrator; the application is open and running with a sensor book open. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the delete sensor function, which gives the sensor ID, room ID and user ID. | |  |
| 2 |  | | According to the user ID, database determines the current user’s attribute and judge whether he has the permission. |
| 3 |  | | According to the sensor ID and room ID, database finds out target sensor. |
| 4 |  | | Remove the target sensor. |
| 5 |  | | Update the other table. |
| 6 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The database removes the target sensor and return the flag of result. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The current user has no authority to delete the sensor. | |  |
| 1 |  | | Return the flag of no permission. |
| 2 | The sensor that be searched does not exist. | |  |
| 2 |  | | Return the flag of not exist. |

### 2.6.9 Server Wants to Add New Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to add new rooms* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shijie Wen, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server wants to add new rooms to the list of rooms he or she can control. | | |
| Summary | The server calls the add room function and transmits the information about the rooms that needs to be added. The database service program adds the room to the room-list in database. | | |
| Actors | Server | | |
| Trigger | The server calls the add room function. | | |
| Precondition | User is an administrator; the application is open and running with a room book open. | | |
| Basic Flow | Actor | | System |
| 1 | The server call adds the room function, which provides the roomID, Lightnum, and Sensornum. | |  |
| 2 |  | | The database adds the room information to the room table. |
| 3 |  | | Update other forms. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | There is a new room in the room list. It is marked to be saved at the next save point. The room book is aware that it has been altered. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the workflow. | |  |
| 1 |  | | The room books he or she controls return to the initial state. |

### 2.6.10 Server Wants to Remove Existing Rooms

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Remove Existing Rooms* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shuihan Zhang, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to delete some rooms from room table. | | |
| Summary | The server wants to delete some rooms from room table. And then the server calls the delete room function. | | |
| Actors | Server | | |
| Trigger | The server calls the delete account function. | | |
| Precondition | The operator’s attribute is the administrator. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the delete room function, which gives the room ID and user ID. | |  |
| 2 |  | | The database determines the current user’s attribute and judge whether it can be deleted. |
| 3 |  | | Find out target room. |
| 4 |  | | Remove the target room. |
| 5 |  | | Update the other table. |
| 6 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The database removes the target room and return the flag of result. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the process after deciding to remove the room. | |  |
| 1 |  | | The database terminates the current operation. |

### 2.6.11 Server Wants to Change the User's Permissions

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Change the User's Permissions* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shuihan Zhang, YuruWang | | |
| Source | Customer | | |
| Goals | The server changes the user permissions. | | |
| Summary | The server wants to changes the user permissions. And then the server calls the change user identity function | | |
| Actors | Server | | |
| Trigger | The server calls the change user identity function | | |
| Precondition | Server makes a request to change the user's permissions. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the change user identity function, which provides the user ID and the modified identity. | |  |
| 2 |  | | Based on the user ID, the user is found in the client table. |
| 3 |  | | Modify the label attribute for this user. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The user is modified to specify permissions. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the process after deciding to the operation of checking the number of people in the room. | |  |
| 1 |  | | The database terminates the current operation. |

### 2.6.12 Server Wants to Add New Actuators

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Change the User's Permissions* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shuihan Zhang, YuruWang | | |
| Source | Customer | | |
| Goals | The server changes the user permissions. | | |
| Summary | The server wants to changes the user permissions. And then the server calls the change user identity function | | |
| Actors | Server | | |
| Trigger | The server calls the change user identity function | | |
| Precondition | Server makes a request to change the user's permissions. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the change user identity function, which provides the user ID and the modified identity. | |  |
| 2 |  | | Based on the user ID, the user is found in the client table. |
| 3 |  | | Modify the label attribute for this user. |
| 4 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The user is modified to specify permissions. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The user decides to "cancel" the process after deciding to the operation of checking the number of people in the room. | |  |
| 1 |  | | The database terminates the current operation. |

### 2.6.13 Server Wants to Remove Actuators from a Room

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | *Server Wants to Remove Existing Rooms* | | |
| Version | 1.0 | Created | ***2019.4.1*** |
| Author | Shuihan Zhang, Yuru Wang | | |
| Source | Customer | | |
| Goals | The server would like to delete some actuators from actuator table. | | |
| Summary | The server wants to delete some actuators from actuator table. And then the server calls the delete actuator function. | | |
| Actors | Server | | |
| Trigger | The server calls the delete actuator function. | | |
| Precondition | The operator’s attribute is the administrator. | | |
| Basic Flow | Actor | | System |
| 1 | The server calls the delete actuator function, which gives the actuator ID, room ID and user ID. | |  |
| 2 |  | | The database determines the current user’s attribute and judge whether it can be deleted. |
| 3 |  | | Find out target actuator. |
| 4 |  | | Remove the target actuator. |
| 5 |  | | Update the other table. |
| 6 |  | | Return the flag of success. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions | The database removes the target actuator and return the flag of result. | | |
| Chart |  | | |
| Alternate Flow | Actor | | System |
| 1 | The current user has no authority to delete the actuator. | |  |
| 1 |  | | Return the flag of no permission. |
| 2 | The actuator that be searched does not exist. | | . |
| 2 |  | | Return the flag of not exist. |
| 3 | The user decides to "cancel" the process after deciding to remove the actuator. | | . |
| 3 |  | | The database terminates the current operation |

## 2.7 Use Cases of Hardware

This section is written for developer who wants to know the functions of hardware.

### 2.7.1 Sensors & Lights Wants to Send the Status

|  |  |  |
| --- | --- | --- |
| Use Case | Sensors & Lights Wants to Send the Status | |
| Version | 1.0 | |
| Author | Rui Raposo, Hongfan Zhang | |
| Source | Directly from Portuguese teacher | |
| Purpose | Send the Status | |
| Goals | Sensors & Lights send the status to client. | |
| Summary | Sensors & Lights send the status to client. | |
| Actors | Sensors & Lights | |
| Trigger | Sensors & Lights send the status to client per minute. | |
| Precondition | Sensors & Lights is connected with client. | |
| Basic Flow | Sensors & Lights | Client |
| 1 | Send status to client |  |
| 2 |  | Receive status. |
| Exception Flows |  |  |
| Postconditions | None | |
| User case diagram |  | |

### 2.7.2 hardware sends signals and gets command

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | hardware sends signals and gets command | | |
| Version | 1.0 | Created | 3-23-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | hardware sends signals and gets command | | |
| Goals | hardware sends signals and gets command | | |
| Summary | hardware sends signals and gets command | | |
| Actors | user | | |
| Trigger | Sensors send their data to communication module. | | |
| Precondition |  | | |
| Basic Flow | Actor | | System |
| 1 | Communication module verify connection to the server | |  |
| 2 |  | | Server will accept the connection and  tell communication module. |
| 3 | Switch sensor tells communication module whether light was operated  or not.  Presence sensor send a picture to raspberry pi to communication module.  Light sensor send its state to communication module. | |  |
| 4 |  | | Communication module sends the switch sensor’s information and 0(not operated)/1(operated)signals to server.  Communication module uses image recognition algorithm to judge whether someone is in room. And then it sends 0(nobody) or 1(someone) signal and presence sensor's information to server.  Communication module send 0(bright) or 1(dark) signal and light sensor's information to server. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions |  | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

### 2.7.3 Server gets signals from communication module

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Server gets signals from communication module | | |
| Version | 1.0 | Created | 3-23-19 |
| Author | Zheng Chen | | |
| Source | User stories | | |
| Purpose | Server gets signals from communication module | | |
| Goals | Server gets signals from communication module | | |
| Summary | Server gets signals from communication module | | |
| Actors | user | | |
| Trigger | Sensors send their data to communication module. | | |
| Precondition |  | | |
| Basic Flow | Actor | | System |
| 1 | server verifies connection from hardware. | |  |
| 2 | Server gets the switch sensor’s information and 0(not operated)/1(operated)signals.  Server gets send 0(nobody) or 1(someone) signal and presence sensor's information.  Server gets 0(bright) or 1(dark) signal and light sensor's information. | |  |
| 3 | The Server decides whether the light should be on or not. | |  |
| 4 |  | | Communication module sends command to lights. |
| Frequency |  | | |
| Type | Primary | | |
| Postconditions |  | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |
| 1 |  | |  |

## 1.1    Intended Audience and Purpose

This document is intended to provided information guiding development process, ensuring that all system requirements are met. The following entities may find the document useful:

* Customer - This page will detail all of the Web App and Android App requirements as understood by the production team. The customer should be able to determine that their requirements will be correctly reflected in the final product through the information found on this page.
* Development Team - Details of specific requirements that the final software build must include will be located here. Developers can use this document to ensure the software addresses each of these requirements.
* QA Team - By developing testing procedures founded in the system requirements, the QA Team can create a comprehensive testing regimen that will guarantee requirements are met.

## 1.2    How to use the document

Table of Contents:

1. Introduction

2. Concept of Operations - broad description of the purpose of the application

2.1 System Context - details any specific system requirements the application will require to run

2.2 System Capabilities - description in prose of all capabilities available to the user in the address book

2.3 Use cases - A detailed look at each functional requirement, describing the application context both before and after an action is taken

3. Behavioral Requirements - How the application will interact with a user

3.1 Input and output requirements - A description of allowed inputs and generated outputs

3.1.1 Input - Describes any restrictions that will be placed on allowed input

3.1.2 Output - Describes the range of outputs that can be generated

3.2 Detailed Output Behavior - Output descriptions in prose

4. Quality Requirements - Requirements not pertaining to the function of the application will be listed here

5. Expected Subsets - Expected levels of functionality at checkpoints during development

6. Fundamental Assumptions - Some specifics about input, output, or behavior upon which other requirements are founded will be listed here

7. Expected Changes - Future features and directions the project is expected to take

8. Appendices - Details aiding the understanding of this document

8.1 Definitions and acronyms - Any technical terms or abbreviations will be spelled out here for ease of use of the document

8.1 Definitions - Definitions of technical or unusual terminology

8.1.2 Acronyms and Abbreviations - Any abbreviated terms will be expanded here

8.2 References - any external references necessary or helpful to understanding this document will be listed here

# 3.    Detailed Requirements

## 3.1 System Inputs and Outputs for Customers

### 3.1.1 Inputs for Web

The input of the application comes from the user.

Login interface comes at the beginning. There are two text boxes to be entered, account number and password.

In the navigation bar, there are "home page", "lights", "Sensors", "rooms", "current user identity" and "user personal information". Click on "lights" and there will be two drop-down menus of "building name" and "room number", "enter" and "return to the previous page" buttons on the left side of the interface. After clicking "Enter", there are all the lights in the room on the right side of the interface, as well as the switch of the lights, the check of the lights (full selection, reverse selection), the status of the light sensor and the prompt information box of the room.

For administrators, there is also a managements button in the navigation bar. If this button is pressed, there will be several buttons appearing, such as *see what alarms are on* button, *see in which room the panic button was pressed* button, *add buildings* button,  *remove buildings* button, *add a room to a building* button, *remove rooms from a building* button, *give teacher permission* button, *revoke teacher permission* button, *change user’ s authority* button, *set up how long will the light stay on without anyone in the room* button, *define the default state of the lights* button, *add sensors* button, *remove sensors* button, *add actuators* button, *remove actuators* button and *turn off alarms* button.

Input at login interface:

\* Account: must be made up of numbers. It can only be one of the teaching number, teacher's work number and administrator's ID number.

\* Password: 6-20 characters.

\* Login: Click on this button to enter the next interface with the correct account number and password.

Under "sensors", click on the Add button and enter the following:

\* Sensor types: Only one of three types can be selected from the drop-down menu.

Under "rooms", click the Add button and enter:

\* Room number: Input cannot conflict with an existing room number. And it is less than 5 legal numbers or letters.

Input basic information:

\* Nickname: less than 20 characters

\* ID number: less than 10 digits

\* School: less than 200 characters

\* Professional: less than 20 characters

\* Class: less than 20 characters

"Modify password" input:

\* Old passwords: 6-20 characters

\* "New password": 6-20 characters.

### 3.1.2 Outputs for Web

Display graphical user interface. Each current interface contains all text boxes or interactive buttons created for users to enter.

Output to the user:

Login interface:

\* If the password or account is incorrect, a pop-up window will prompt "incorrect password or account".

Turn on the lights:

\* If the user is a student and the room is occupied, when the "turn on" button is pressed, a pop-up window will prompt "the room is occupied, the students can not turn off the lights at will". If the room is unoccupied, when the "turn off" button is pressed, a window will pop up to indicate that "the room is unoccupied", and students can not turn on the light at will. If the switch is checked, similar.

### 3.1.3 Inputs for APP

The input of the application comes from the user.

Login interface comes at the beginning. There are two text boxes to be entered, account number and password.

After logging in, the app will display building interface. In the bottom navigation bar, there are "buildings" "rooms", "user profile" and "current user identity". Click on these buttons will change to different interface. In "buildings", there is a list of buildings, including the building name. Click specific building will jump to the “rooms” interface where a list of rooms in this building is, including the building where these rooms are and the room number. After clicking a specific room, there are all the lights’ names/numbers in the room on the left side of the interface, while the switches of the lights display on the right side of the interface (switch shows the status of lights). At the bottom of this list, there are all sensors and their status.

Input at login interface:

* Username: 8-20 characters and cannot contains space.
* Password: 6-20 characters.
* Login button: Click on this button to verify username and password and jump to the next interface with the correct account number and password.

### 3.1.4 Outputs for APP

Android app uses UI interface to interact with user.

Login interface:

If the username and password is not matched, a pop-up window will prompt "Username and password don’t match".

## 3.2 Detailed Output Behavior for Customers

### 3.2.1 For Web

Login interface comes at the beginning. There are two text boxes to be entered, account number and password.

In the navigation bar, there are "home page", "lights", "Sensors", "rooms", "current user identity" and "user personal information". Click on "lights" and there will be two drop-down menus of "building name" and "room number", "enter" and "return to the previous page" buttons on the left side of the interface. After clicking "Enter", there are all the lights in the room on the right side of the interface, as well as the switch of the lights, the check of the lights (full selection, reverse selection), the status of the light sensor and the prompt information box of the room. From the administrator's perspective, there is a red remove button next to each light, and a green new one light button in the right place. The lower right corner of the interface has remove ticks.

Click on "sensors" and there will be two drop-down menus of "building name" and "room number", "enter" and "return to the previous page" buttons on the left side of the interface. Click "Confirm" and all the sensors and their status will appear on the right side of the interface.

Click on "rooms" and there will be a drop-down menu of "teaching building name", "confirmation" and "return to the previous page" buttons on the left side of the interface. Click on the "Confirm" button and all the room numbers in this building will appear on the right side of the interface.

Click on "User Personal Information" and the buttons "Basic Information" and "Modify Password" appear on the left side of the interface. After clicking on the "basic information", there will be "nickname", "ID number", "school", "major" and "class" on the right side of the interface, as well as a "confirm modification" button. Click "Modify Password" and the text box of "New Password" and "Old Password" will appear on the right side of the interface, and the button "Confirm Modification" will appear.

For administrators, there is also a managements button in the navigation bar. If this button is pressed, there will be several buttons appearing, such as *see what alarms are on* button, *see in which room the panic button was pressed* button, *add buildings* button,  *remove buildings* button, *add a room to a building* button, *remove rooms from a building* button, *give teacher permission* button, *revoke teacher permission* button, *change user’ s authority* button, *set up how long will the light stay on without anyone in the room* button, *define the default state of the lights* button, *add sensors* button, *remove sensors* button, *add actuators* button, *remove actuators* button and *turn off alarms* button.

Click on *see what alarms are on* button, and UI will display the list of alarms which are on in the main page. Click on *see in which room the panic button was pressed* button, and UI will display the room’s information in which the panic button was pressed. Click on *add buildings* button or *remove buildings* button. If *remove buildings* button is pressed, UI will show up a building drop-down menu and a *confirm* button. If *add buildings* button is pressed, UI will show up a *confirm* button and a text box which lets administrator input the new buildings’ information. Click on *add a room to a building* button or *remove rooms from a building* button. UI will update the page with a building drop-down menu and let the administrator choose one building. If *add a room* button is pressed, UI will show up a *confirm* button and a text box which lets administrator input room number. If *remove rooms* button is pressed, the UI will show up a *confirm* button and a list of check boxes which lets administrator choose rooms to remove. Click on *give teacher permission* button or *revoke teacher permission* button. UI will show up a *confirm* button and a text box which lets administrator input the teacher’s information. Click on *change user’s authority* button. UI will show up a text box which let administrator input user’s information, a drop-down menu which lets the administrator choose new authority and finally a *confirm* button. Click on *set up how long will the light stay on without anyone in the room* button. UI will show up a *confirm* button and a text box which lets the administrator input the time. Click on *define the default state of the lights* button. UI will show up a *confirm* button and a state drop-down menu which lets administrator choose the state. Click on *add sensors* button or *remove sensors* button. Administrator chooses a building and a room from two drop-down menus. If *add sensors* button is pressed, UI will show up a *confirm* button and a text box which lets administrator input sensors’ information. If *remove sensors* button is pressed, UI will show up a *confirm* button and a list of check boxes which lets the administrator choose sensors. Click on *add actuators* button or *remove actuators* button. Administrator chooses a building and a room from two drop-down menus. If *add actuators* button is pressed, UI will show up a *confirm* button and a text box which lets the administrator input actuators’ information. If *remove actuators* button is pressed, UI will show up a *confirm* button and a list of check boxes which lets administrators choose actuators. Click on *turn off alarms button*. UI will show up a *confirm* button and a drop-down menu which lets administrator choose a building.

### 3.2.2 For APP

The input of the application comes from the user.

Login interface comes at the beginning. There are two text boxes to be entered, account number and password.

After logging in, the app will display building interface. In the bottom navigation bar, there are "buildings" "rooms", "user profile" and "current user roles". Click on these buttons will change to different interface. In "buildings", there is a list of buildings, including the building name. Click specific building will jump to the “rooms” interface where a list of rooms in this building is, including the building where these rooms are and the room number. After clicking a specific room, there are all the lights’ names/numbers in the room on the left side of the interface, while the switches of the lights display on the right side of the interface (switch shows the status of lights). At the bottom of this list, there are all sensors and their status.

If click “rooms” directly, app will jump to last rooms edited/explored by user previously.

“user profile” interface will display username, nickname, name, a “change password” button and a “log out” button.

“current user roles” is a textbox and should display user’s role. This textbox is disabled.

## 3.3 System Inputs and Outputs for Developer

### 3.3.1 Inputs

The inputs send to the server when client queries hardware’s data should be in the form of json which content is:

uid: The user’s unique identification.

sid: User’s secure ID.

hid: The hardware’s unique identification.

The inputs send to the server when client want to operate a hardware should be in the form of json which content is:

uid: The user’s unique identification.

sid: User’s secure ID.

hid: The hardware’s unique identification.

cmd: The command client sent.

The inputs send to server when hardware want to report their data should be in the form of json which content is:

data: The data which sensor want to report.

The inputs send to server when intelligence controller generated command should be in the form of json which content is:

data: The command that intelligence controller generated.

ROOM{

\*Room\_id: the id of the room

\*Light state{

\*State: it can be a boolean type, whose value is true or false. True means that it is on now, while false means the opposite.

...

}

\*Sensor state{

\*kind: it is a string type, has three values, {motion, light, button}

\*online: it is a boolean type.

\*value: It is a numerical type.

}

};

Instruction{

\*User\_priority: it is a numerical type and means user’s priority

\*Instruction\_type: the instruction has four kinds, { auto, instruction, time, rules}.

\*Extra\_information: set time period or make rules.

};

Extra\_information{

\*Data\_about\_time: .....

\*Data\_about\_rule: ......

\*Data\_about\_priority: ......

} ;

The input to the database comes from the server. The input to the database comes from the server. There are 5 tables in the database, namely client table, light table, sensor table, room table and actuator table. The input requirements for each attribute of each table are as follows.

|  |  |  |
| --- | --- | --- |
| Name | Type | Explanation |
| UID | int[1] | UID is the user's account number, which is an integer less than max\_int. |
| name | char[20] | name is a string of up to 20 lengths representing the user name |
| password | char[50] | The password is to save the password of each user. It should be encrypted. |
| label | int[1] | label saves the attribute identification of each user, indicating that he is a student, teacher, or administrator account. |
| LID | int[1] | LID is the light's number in a room, which is an integer less than max\_int. |
| roomID | int[1] | roomID should be generated when adding rooms. They cannot be modified and they are different. |
| State | int[1] | State is an integer that holds the state of the lamp on, off, or damaged |
| Settime | string | SetTime represents the installation time of the bulb, which should be a string limited to yyyy-mm-dd format |
| Life | int[1] | Life is an integer representing the life of a light bulb in hours |
| SID | int[1] | SID is the number of sensor, which is an integer less than max\_int. |
| Type | int[1] | Type is an integer describing the type of sensor |
| Lightnum | int[1] | Lightnum is an integer describing the number of bulbs in a room |
| sensornum | int[1] | sensornum is an integer describing the number of sensors in a room |
| AID | int[1] | AID is the number of actuator, which is an integer less than max\_int. |

### 3.3.2 Outputs

The outputs send to intelligence controller from server when something need to do with hardware should be in the form of json which content is:

sensors: The list of sensors with their up-to-date data.

device: The device and its up-to-date data.

cmd: The command (Leave blank if there is no command existed.)

authority: The level of operator.

The outputs send to client when server report hardware’s information should be in the form of json which content is:

hid: The hardware’s unique identification.

online: Whether the hardware is online.

nickname: The nickname of hardware.

last: The timestamp of last update.

data: The hardware’s data.

The outputs send to hardware when server send command should be in the form of json which content is:

data: The command.

The outputs send to the Server.

\*Result: There outputs required, there are {value, room, hint}.

{

\*value: it is a string type whose value is in set:{“open”, “close”, “null”, “exception”} . “open” means turn on the light, “close” means turn off the light, “null” means do nothing and “exception” means don’t change the light and send some error information to the Server.

\*room: it is a numerical type that means the result for which room.

\*hint: it is a string type, the content is for explaining the result when intelligent control system reject the command.

}

The output of the database is provided to the server. The following table specifies the specific form of the output that will be provided to the server.

|  |  |  |
| --- | --- | --- |
| Name | Type | Explanation |
| UID | Int[1] | UID is the user's account number, which is an integer less than max\_int. |
| name | Char[20] | name is a string of up to 20 lengths representing the user name |
| password | Char[50] | The password is to save the password of each user. It should be encrypted. |
| label | Int[1] | label saves the attribute identification of each user, indicating that he is a student, teacher, or administrator account. |
| LID | Int[1] | LID is the light's number in a room, which is an integer less than max\_int. |
| roomID | Int[1] | roomID should be generated when adding rooms. They cannot be modified and they are different. |
| State | Int[1] | State is an integer that holds the state of the lamp on, off, or damaged |
| Settime | string | SetTime represents the installation time of the bulb, which should be a string limited to yyyy-mm-dd format |
| Life | Int[1] | Life is an integer representing the life of a light bulb in hours |
| SID | Int[1] | SID is the number of sensor, which is an integer less than max\_int. |
| Type | Int[1] | Type is an integer describing the type of sensor |
| Lightnum | Int[1] | Lightnum is an integer describing the number of bulbs in a room |
| sensornum | Int[1] | sensornum is an integer describing the number of sensors in a room |
| AID | Int[1] | AID is the number of actuator, which is an integer less than max\_int. |
| Flag | Bool[1] | Flag is a flag indicating whether the operation on the database is successful |

**3.5 Detailed Output Behavior for Developer**

The database provides various access interfaces to the server. This section details the capabilities of these interfaces and their possible output formats.

* Function1: query the corresponding account information according to the user UID

Query the client-database with UID as the primary key.

1. If the user of UID does not exist in the database, return null.

2. If the user exists, return the output value.

* Function2: query the light information according to LID and roomID

Query the light-database with LID and roomID as the primary key.

1. If the light of SID does not exist in the database, return null.

2. If the light exists, return the output value.

* Function3: query light information in a room through roomID

Query the information of all the bulbs in the database whose room number equals the query value

1. If no light bulb has the same room number as the query value, return empty.

2. In other cases, list all light bulb information with room number equal to query value.

* Function4: query the sensor information according to the sensor SID

Query the sensor-database with SID as the primary key.

1. If the sensor of SID does not exist in the database, return null.

2. If the sensor exists, return the output value.

* Function5: query sensor information in a room through roomID

Query the information of all the bulbs in the database whose room number equals the query value

1. If no sensor with room number equal to the query value is found in the database, return empty

2. In other cases, list all sensors information with room number equal to query value.

* Function6: query room information by roomID

Query the room-database with roomID as the primary key.

1. If the user of rommID does not exist in the database, return null.

2. If the user exists, return the output value.

* Function7: list all the rooms

input: no iuput

output: roomID(int[1]), lightnum(int[1]), sensornum(int[1])

Detailed output：

Traverse the room database and output all information.

1. If the database is empty, return null.

2. Output all information of the room database.

* Function8: query the sensor information based on the actuator AID

Query the actuator with AID as the primary key.

1. If the actuator of AID does not exist in the database, return null.

2. If the driver exists, return the output value.

* Function9: add/remove/modify a light

First use the roomID as the primary key to query the room-database, and then use the roomID and the LID as the primary key to query the light-database.

1.If the room dose not exist, the flag is false.

2.If the LID in the room has exist, the flag is false.

3. Else the flag is true

* Function10: add/delete/modify a room

Query the room-database with roomID as the primary key.

1.If the roomID has already exist ，the flag is false.

2 Else the flag is true

* Fuction11: add/remove/modify a sensor

First use the roomID as the primary key to query the room-database, and then use the room number and the SID as the primary key to query the light-database.

1.If the room does not exist, the flag is false.

2. If the SID in the room has exist, the flag is false.

3. Else the flag is true.

* Fuction12: add/delete/modify an actuator

First use the roomID as the primary key to query the room-database, and then use the room number and the AID as the primary key to query the light-database.

1. If the room does not exist, the flag is false.

2. If the AID in the room has exist, the flag is false.

3. Else the flag is true.

* Fuction13: add/delete/modify a user

input: SID(int[1]), roomID(int[1])

output: flag(bool[1])

Detailed output：

1. If the UID has already exist, the flag is false.

2. In other condition, the flag is true.

# 4.   Quality Requirements (Non-functional Requirements)

The system must show good behavior in many fields like Performance, Security, Availability, Reliability, Modifiability, Maintainability, Understandability.  
Interface aesthetics:

Simple, comfortable and elegant.

Performance:

The system can respond the users’ operation in less than 500ms

The hardware can respond the command in less than 1000ms

Security:

The system must have different authority. The administrator’s jurisdiction must not be used by any other users.

Availability:

The user’s operation must be judged strictly by control part. Every situation must have a solution even if the user has a wrong operation.

Reliability:

The system must be anti-interference. When some signal comes in a wrong way, the system should recognize it and give the respond.

Modifiability:

The system can be changed. When users need some new functions, we can add up them into the system.

Maintainability:

The system has to easily to be fixed. If some parts get wrong, it can easily to find some other things to take place.

Understandability:

The system must be easy for users. The UI and specification have to be good for users.

# 5. Expected Subsets

L0:

- Basic GUI.

- Users can log in. Ability to send data to back-end storage and call data from back-end storage.

L1:

- Better GUI

- Ability to add/remove actuators (lights). Administrators have this permission.

- Ability to add/delete new rooms. Administrators have this permission.

- Ability to add/remove sensors.

L2:

- Complete GUI for Intelligent Lighting Control

- Ability to see the status of the light. All three users have this permission.

- Check if a room is occupied. All three users have this permission.

- Ability to check the status of the light sensor. All three users have this permission.

- Ability to turn on/off the light. All three users have this right.

# 6.   Fundamental Assumptions

Hardware: Raspberry pi 3B+, Camera, Light sensor, Light.

Software: Linux operating system，Python 3.6

# 7.    Expected Changes

* Add light history analysis function.
* Add monitor function.
* Adjust the brightness of the light
* Personal Web Pages for Skin Change
* Provide personalized web customization
* Provide hotline for maintenance personnel.
* Provide multilingual support.
* Retrievable password and change password at any time
* Support binding mobile phone number and login by phone number.

# 8.    Appendices

## 8.1    Definitions and acronyms

### 8.1.1    Definitions

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
| Raspberry Pi | A portable single-board computer |
| Untouchable | If this component is touched, nothing will happen. |
| Time-out value | how long will the light stay ON when lights are left ON and there is none in this room |
|  |  |

### 8.1.2    Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or**  **Abbreviation** | **Definitions** |
| GUI | Graphical User Interface |
| IC | Intelligent controller |
|  |  |

## 8.2 key technology

### 8.2.1 socket instead of web

**By using socket instead of web, the communication between sever and other parts could be easier. Server don’t need to roll polling anymore, and that could be save a lot of time and bandwidth.**