**Software Requirements Specification (SRS)**

Revision History:

|  |  |  |
| --- | --- | --- |
| Date | Author | Description |
| 2019.3.25 | Li Yuanjin | Editing Software Requirements Specification |
| 2019.3.26 | Zhang Yifan | Editing the Detailed Requirements |
| 2019.3.26 | Wang Zhongyu | Editing the Quality Requirements |
| 2019.3.31 | Zhang Yifan | Modify the use cases |
| 2019.3.31 | Li Yuanjin | Modify the use cases |
| 2019.4.1 | Li Yuanjin | Modify the Output |
| 2019.4.1 | Zhang Yifan | Modify the Input |
| 2019.4.1 | Zhang Yifan | Add the Definitions |
| 2019.4.1 | Li Yuanjin | Modify the use cases |
|  |  |  |

**1.  Introduction**

**1.1    Intended Audience and Purpose**

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

Primary Customer - This page will detail all of the application 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.

User - A prospective user will be able to use this document to identify the main functionality included in the application. Furthermore, the application will have a set of system requirements before the application can be run. Details regarding these requirements can be found here.

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. System Capabilities - 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  
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  
    8.1.1 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

**2.** **System Capabilities**

This product is a control system for intelligent lighting systems which developed for university classrooms. The system can realize the control of the switch state of the lights in the classroom by analysis according to the information of the light sensor and the infrared sensor sent to the server and the received network request.

**2.1    System Context**

The system needs to be used on the PC side, so the system environment applied is as follows:

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)
* RAM: 128 MB
* Disk space: 128 MB
* Processor: Minimum Pentium 2 266 MHz processor

Mac OS X:

* Intel-based Mac running Mac OS X 10.8.3+, 10.9+
* Administrator privileges for installation

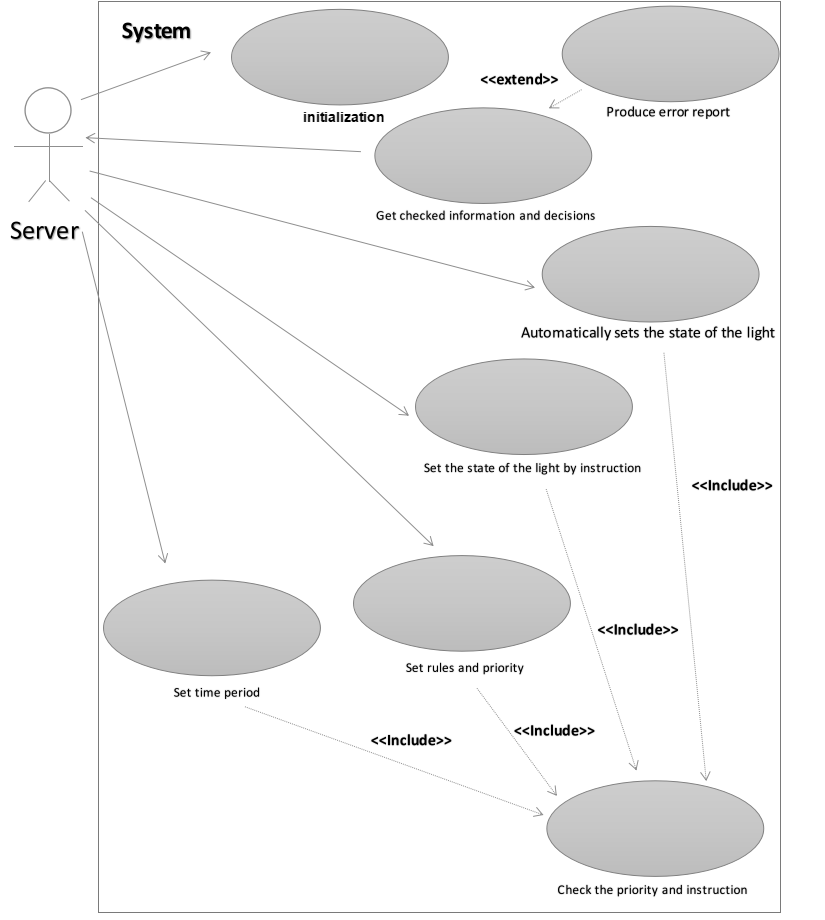
Linux:

* Oracle Linux 5.5+1
* Oracle Linux 6.x (32-bit), 6.x (64-bit)2
* Oracle Linux 7.x (64-bit)2 (8u20 and above)
* 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**

The control system can be divided into two main modes: automatic control mode and human command mode. In the automatic control mode, the system can automatically determine whether it is necessary to change the state of the lamp and execute it by acquiring sensor information. In the human command mode, the system performs different operations according to the authority of the command. When the authority level of the instruction is high, the instruction can be executed, however, when the authority level of the instruction is low, in order to prevent the misoperation behavior, the situation is judged by acquiring the sensor information to intelligently control the light in the room.

**2.3 Use Cases**



**2.3.1 Initialize the system**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Initialize the system | | |
| Version | 1.0 | Created | *2019-4-1* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | Initialize the system | | |
| Goals | Make the system start to work | | |
| Summary | Server give a signal to make the system initialized. | | |
| Actors | Server | | |
| Trigger | Customer start the system | | |
| Precondition | None | | |
| Basic Flow | *Actor* | | System |
| 1 | Server give a package of the data to initialize the system | |  |
| 2 |  | | Initialization and give a reply |
| Frequency | Once. | | |  |
| Type | Primary | | |
| Postconditions | The project assignment is created | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |

**2.3.2 Automatic mode**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Automatic mode | | |
| Version | 2.0 | Created | *2019-4-1* |
| Author | Li Yuanjin | | |
| Source | Requirement | | |
| Purpose | Power saving intelligently | | |
| Goals | Control the status of the light | | |
| Summary | Automatically sets the state of the light. | | |
| Actors | Server | | |
| Trigger | None | | |
| Precondition | Automatic mode | | |
| Basic Flow | *Actor* | | System |
| 1 | Server give a package of the data | |  |
| 2 |  | | Judge the situation, check the priority and instruction and give the command |
| Frequency | 1 time in a minute | | |  |
| Type | Primary | | |
| Postconditions | The project assignment is created | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |

**2.3.3 Command-light mode**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Use Case | | Command-light mode | | | | |
| Version | | 2.0 | | Created | *2019-3-31* | |
| Author | | Zhang Yifan | | | | |
| 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: Send instruction to change the state  of the light | | |  | |
| 2 | |  | | | Check the priority and instruction and make a decision back to the server | |
| Frequency | | 2s | | | | |
| Type | | Primary | | | | |
| Postconditions | | The project assignment is created | | | | |
| Chart | |  | | | | |
| Alternate Flow | | *Actor* | | | System | |

**2.3.4 Time setting mode**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Time setting mode | | |
| Version | 2.0 | Created | *2019-3-31* |
| Author | Zhang Yifan | | |
| 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. | | |
| Goals | Set the time period | | |
| Summary | An administrator issues a command to change the time periods 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. | | |
| Actors | Server | | |
| Trigger | A command to change the time periods | | |
| Precondition | The command came from an administrator. | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends data to Intelligent Control | |  |
| 2 |  | | By checking the priority and instruction system make a decision and send it to Server |
| Frequency | 2s | | |
| Type | Primary | | |
| Postconditions | The project assignment is created | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |

**2.3.5 Rules setting mode**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Rules setting mode | | |
| Version | 2.0 | Created | *2019-3-31* |
| Author | Zhang Yifan | | |
| Source | Requirement | | |
| Purpose | (The administrator) Set the rules of our system, including priority and orders | | |
| 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. | | |
| Actors | Server | | |
| Trigger | A command to set the rules. | | |
| Precondition | The command came from an administrator. | | |
| Basic Flow | *Actor* | | System |
| 1 | Server sends data to Intelligent Control System | |  |
| 2 |  | | By checking the priority and instruction system make a decision and send it to Server |
| Frequency | 2s | | |
| Type | Primary | | |
| Postconditions | The project assignment is created | | |
| Chart |  | | |
| Alternate Flow | *Actor* | | System |

**3.    Detailed Requirements**

**3.1 System Inputs and Outputs**

This system will automatically ask for data about the sensors and lights from the Server in cycles.

And when a instruction arrives at the Server, it will sends data to our system, including the state of the light, the the relevant sensors’ data, the user’s information and the instruction.  
The sensors now have three kinds, which are the motion sensor, light senor and button sensor.

**3.1.1 Inputs**

(1)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.

}

};

(2)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.

};

(3)Extra\_information{

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

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

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

} ;

**3.1.2 Outputs**

This system will send its output 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.

}

**4   Quality Requirements (Non-functional Requirements)**

**Behavioral (run-time) qualities:**

* Performance

the elapsed time between the IC receive the command from server and the IC

system’s response must be <= 100 ms

* Security

No body can change the rule except administrator

* Reliability

The IC can ignore those commands illegal or too often such as

1. Students want to turn on the light when there is nobody in the room.
2. Students want to turn on the light when the light is enough.
3. IC received more than one command during 1 second.
4. IC received the student’s command after teacher’s command one minute later.(the IC must set how long does each command can effect or shield other commands)

* Availability

If receives an illegal command, IC should return the error information to the server such as :

1. Condition error: students want to turn off the light but somebody is in the room.(there is confliction between students’ commands and sensors’ information)
2. Authority error: students want to turn off the light during the specific time set by administrator.

**Developmental (design-time) qualities:**

* Modifiability
* Maintainability
* Portability
* Reusability
* Understandability

**5. Expected Subsets**

The core of this intelligent control system is to be able to analyze the priority of turning on/off the light according to the authority and sensor status.

**6.    Fundamental Assumptions**

The application can run on any system.

Developing Software: not sure.

Developing Environment: Raspberry Pi

**7.    Expected Changes**

7.1.We expect that the system can be expended to serve for multiple rooms and multiple lights. We can group rooms and light to fit our needs in different situation.

**8.    Appendices**

**8.1    Definitions and acronyms**

**8.1.1    Definitions**

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
| \*Sensor state | The structure contains attributes that all sensors have. |
| \*online | Boolean type, true means the sensor is online, false means offline |
| \*value | Its meaning changes over different sensor. For motion sensor, if there are someone in the room, the value will be larger than zero. And for the “light sensor”, the brighter the room, the lager the value will be. Finally, for the “button sensor”, if the user wants to change the light’s state, the value will be larger than zero |
| \*User\_priority | The smaller the number is, the low-level permissions the user has. Now settings are that students have level-1, buttons have level-2 and teachers & Administrators both have level-3. |
| \*Data\_about\_time | Some information about the instruction that plans to set the time period |
| \*Data\_about\_rule | Some information about the instruction that plans to set the rules. |
| \*Data\_about\_priority | Some information about the instruction that plans to set the priority |

**8.1.2    Acronyms and abbreviations**

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

**8.2    References**