SMART LIGHTING

Software Design Specification

Introduction to Software Engineering

TEAM 5

Team Leader	황정윤
Team Member	정성욱
Team Member	이채은
Team Member	유상범
Team Member	Aysun Ogut
Team Member	Selin Samra

Contents

LIST OF FIGURES

LIST OF TABLES

1. Preface

This section contains the readership, scope, objective and document structure of the Software Design Specification of the Smart Lightning project.

1.1. Readership

This document consists of 10 parts and each part has sub-parts. What is explained in each title will be explained in section 1.4. The main reader of this document is Team 5, in addition, professors and TAs of the Introduction to Software Engineering course and students can be considered as readers of this document.

1.2. Scope

This document covers the design features of the Smart Lighting project. In general, it aims to show the specification of the user's access to devices via the app and the design of the technical association of these devices.

1.3. Objective

The purpose of creating this Software Design Specification is to explain the design in the Smart Lighting project with explanations and visuals. It presents the stages in the implementation of the project by addressing all the necessary aspects. It describes the design specifications of the project from different angles, such as frontend, backend, database. Diagrams and function explanations are also included in this document.

1.4. Document Structure

- **Preface:** This section contains the readership, scope, objective and the structure of the Software Design Specification document.
- **Introduction**
- **Overall System Architecture**
- **Frontend System Architecture**
- **Backend System Architecture:** This section contains the backend structure of the project with class diagram and sequence diagram.

- **Protocol Design**
- **Database Design:** This section contains the ER diagram of the project. It describes the entities in the project, their attributes and the relationships between them.
- **Testing Plan:** This section contains the plan of the testing of the project.
- **Development Plan:** This section contains which tools will be used to develop the application with constraints, assumptions and dependencies for developing the system.
- **Supporting Information:** This section contains the document history.

2. Introduction

Smart Lighting project is an automatic lighting system using various sensors and GPS, and the purpose of this project is to improve the quality of the user's residential environment and help them live a more convenient and prosperous life. The system operates in a form that adjusts lighting and the appliances based on IoT in a certain situation or a way that users set in advance. The user includes the general public, including the hearing impaired. By replacing the auditory elements of the residential environment with the color of visual lighting and the intensity of light, it is possible to provide a more user-friendly service.
This design specification specifies the design structures used directly or indirectly to implement the smart lighting system. The described design structure is designed according to the requirements made in the previous Software Requirement Specification document.

2.1. Objectives

This chapter describes the various diagrams and tools applied to this project in the design phase, and scope of the project.

2.2 Applied Diagrams

2.2.1. UML

UML is an abbreviation for the Unified Modeling Language, a standardized general-purpose modeling language used in software engineering. UML provides a rich visual representation model based on standardized techniques and is able to apply regardless of the type or size of the system. Similarly, it is easy to understand and allows developers to communicate with a

uniform understanding of the program. In this project, UML will structure the operation and relationship of the system, and also help readers to understand them more easily.

2.2.2. Use Case Diagram

Use Case Diagram is an expression of the interactions between a system and its environment, showing the service or function of the system and external elements related thereto from the user's point of view. It indicates which features are available to users within the system represented by circle, ellipse, and stick figures. This diagram allows customers and developers to coordinate opinions on requirements.

2.2.3. Class Diagram

Class Diagram is a static diagram that expresses the components of an object class and the associations between classes. This diagram can represent the structure of some or all of the system, and it is also possible to visualize abstract concepts that are not easily visible from the user's point of view.

2.2.4. Sequence Diagram

Sequence Diagram is a diagram that defines objects for problem solving and represents a sequence of interactive messages between objects over time. In other words, it is a time-based expression of how an actor or each system component works with a message in the operation of a specific function of a program. So, it is easy to grasp the scenario of the system.

2.2.5. Context Diagram

Context Diagram expresses the scope of a system in an organization. By showing system boundaries, external objects interacting with the system, and major information flows between the object and the system, the entire system can be viewed. It does not describe the details, but it is usually expressed at the highest level.

2.2.6 Entity Relationship Diagram

Entity Relationship Diagram is called the ER diagram, which analyzes systems from a database perspective. As a result of the process of Entity-Relationship Modeling(ERM), it represents data as entity, attribute, and relationship, and expresses the structure of the data and the accompanying constraints.

2.3. Applied Tools

2.3.1. Microsoft PowerPoint

Microsoft PowerPoint is a program originally used to create presentations, but it is also suitable for creating simple figures or organized charts. In the process of preparing this specification document, it was selected because it was convenient to use when a figure to be referred to is required rather than a UML diagram.

2.3.2. GitMind

GitMind is a free-of-charge mind map production tool written in JavaScript. This program is used to draw UML diagrams. Since various templates are provided to facilitate drawing diagrams or flowcharts, it is possible to write what we want to express more conveniently.

2.4. Project Scope

Smart Lighting system is a project that stems from a need for improvement for all users, including impaired persons, to lead a more prosperous and convenient life, and to help physical disabilities not lead to life disabilities. Since it directly affects the user's residential environment, the data server to manage the user's personal information and settings is needed. And the system and hardware must be linked, requiring appropriate hardware performance. Through this system, users will be able to live in a more convenient and automated environment.

3. Overall System Architecture

3.1. Objective

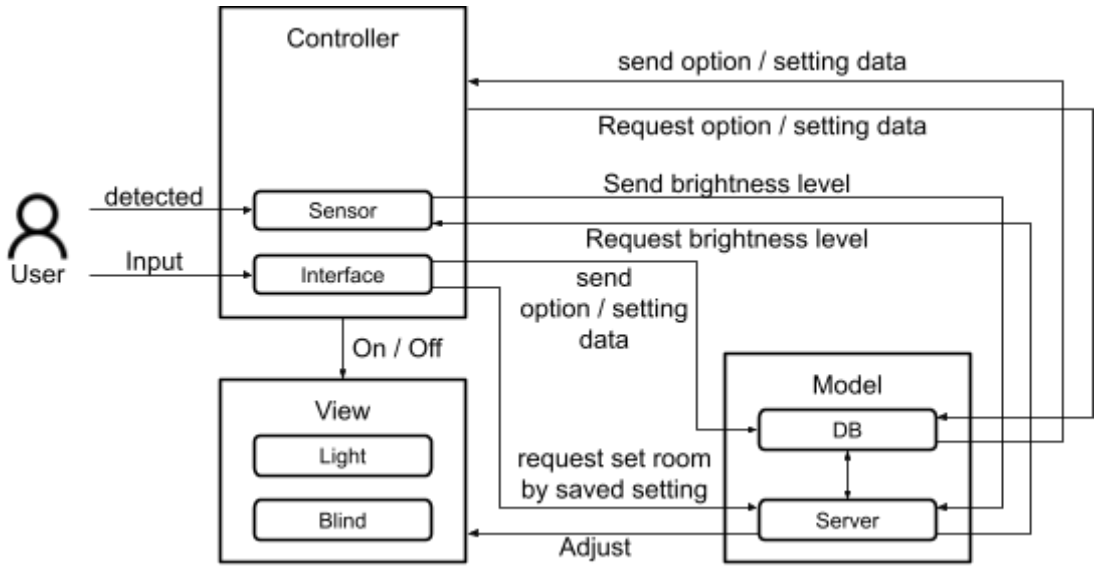
This chapter describes the design configurations from the front-end to the back-end of the system.

3.2. System Configuration

Our system consists of five main parts: Main control system, Light control system, Sensor, Interface, and Database. Main control system deals with all of the signals coming from the sensors, interface(smartphone). It also reads data from the database and writes data to the database. And it sends signals to the light control system to turn on/off the light and change the brightness & color of the lightbulbs. Light control system receives signals from the main control system and based on the signals, it turns on/off the light and changes the brightness & color of the lightbulbs. Sensor consists of several sensors: sleep detection sensor, voice recognition sensor, indoor brightness measurement sensor, sensor that detects user entering or leaving the house, sensor that detects user entering a room, door bell, emergency alarm sensor. When these sensors detect a situation that they intend to detect, they send signals to the main control system. Interface is the smartphone device that the user controls the lighting settings with. By tapping buttons on the screen, the user can send signals to the main control system in order to give a change to the lighting configuration. And it also receives current lighting configuration information and displays on the screen. Database is where all of the data are stored. It consists of setting data, data of configuration of the lightings, etc.

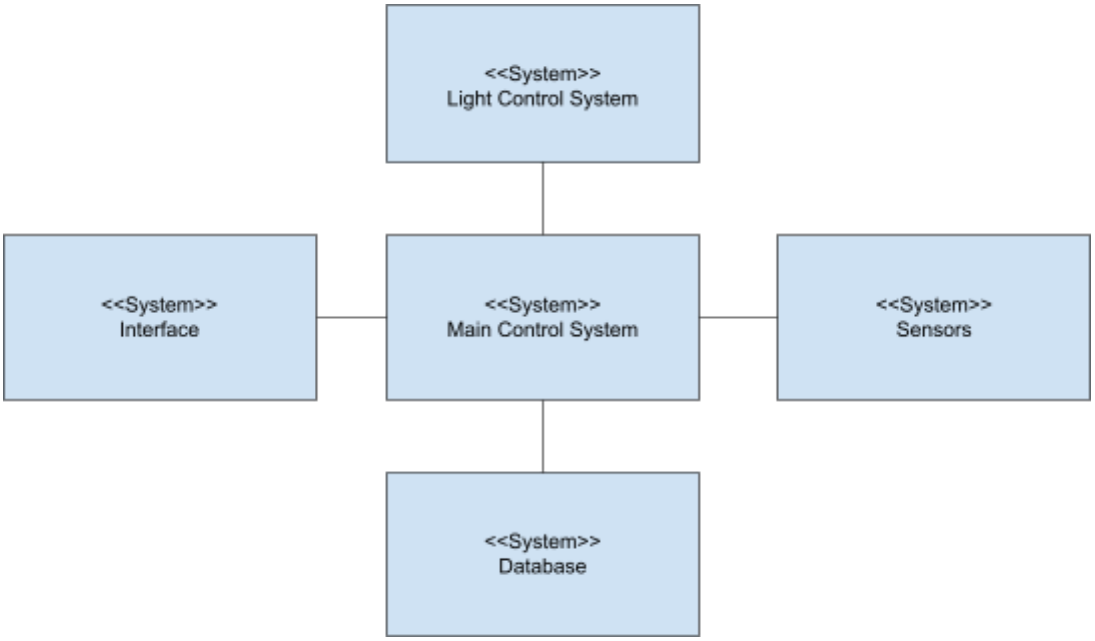
3.2.1. MVC Model

[Figure 1] MVC Model Diagram



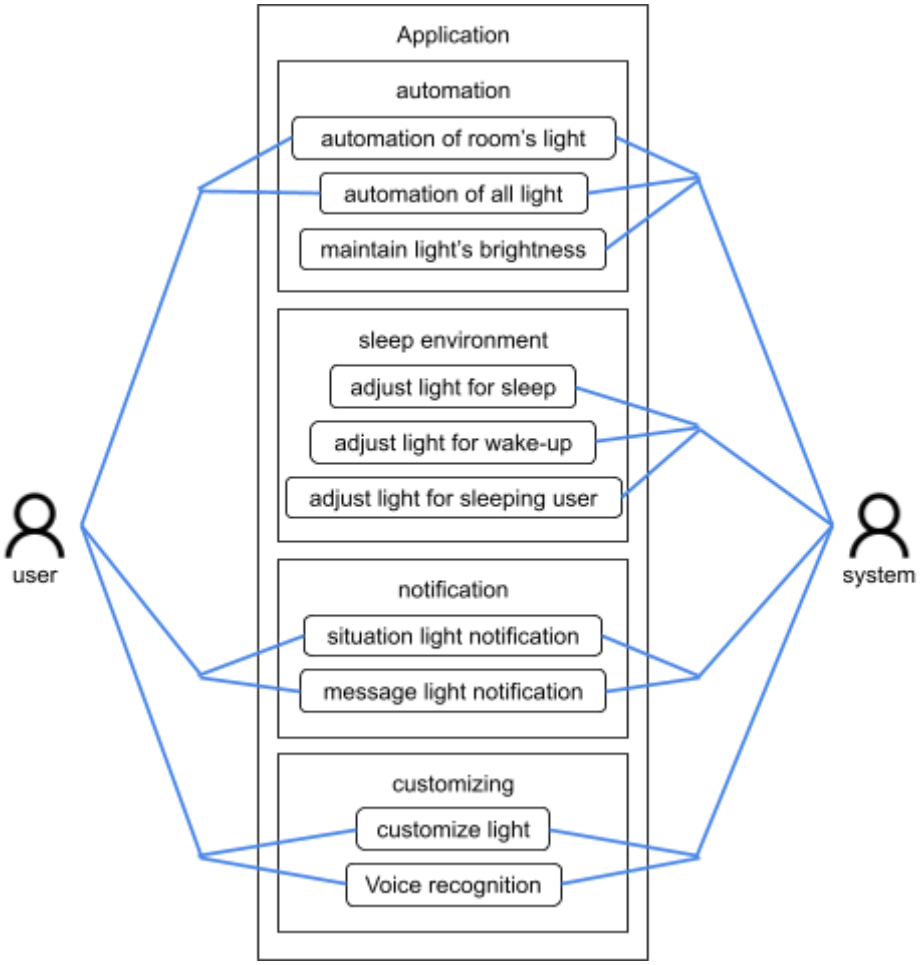
3.2.2. Context Diagram

[Figure 2] Context Diagram



3.2.3. Use Case Diagram

[Figure 3] Use case Diagram



4. Frontend System Architecture

4.1. Objectives

This chapter describes components(attributes & methods) of the frontend system and the relationship between components.

4.2. Subcomponents

4.2.1. Main Interface

The main interface class has information of all smart-house system’s rooms and handles a user’s request to the current room.

4.2.1.1. Attributes

These are the attributes that the main interface class has

- **room_list** : list of rooms that smart-house system have
- **cur_room** : room object that is displayed on the screen of user interface now
- **device_list** : list of device_id of cur_room
- **config_list** : list of config_id cur_room

These are the attributes that the room object has

- **room_name** : name of room.
- **room_id** : id of room. room_id of each room is unique.
- **device_list** : list of device object
- **config_list** : list of configuration object

These are the attributes that the device object has.

- **device_name** : name of device
- **device_id** : id of device. device_id of each device is unique.
- **manufacture** : manufacture of device
- **device_type** : type of device. light source or blinder
- **isActive** : power state(on / off) of device
- **brightness** : brightness of device. percentage value compared to max
- **color** : color of device's light. RGB value. If it does not support color function, the value is null.
- **effect** : current effect state of device's light. If it does not support effect function, the value is null.
- **monthlyUsage** : the amount of the usage of the light for one month
- **dailyUsage** : the amount of the usage of the light for one day
- **isNotif** : state(on / off) of notification option

These are the attributes that the configuration object has.

- **config_name** : name of configuration
- **config_id** : id of configuration. config_id of each configuration is unique.
- **device_list** : predefined list of device object

4.2.1.2. Methods

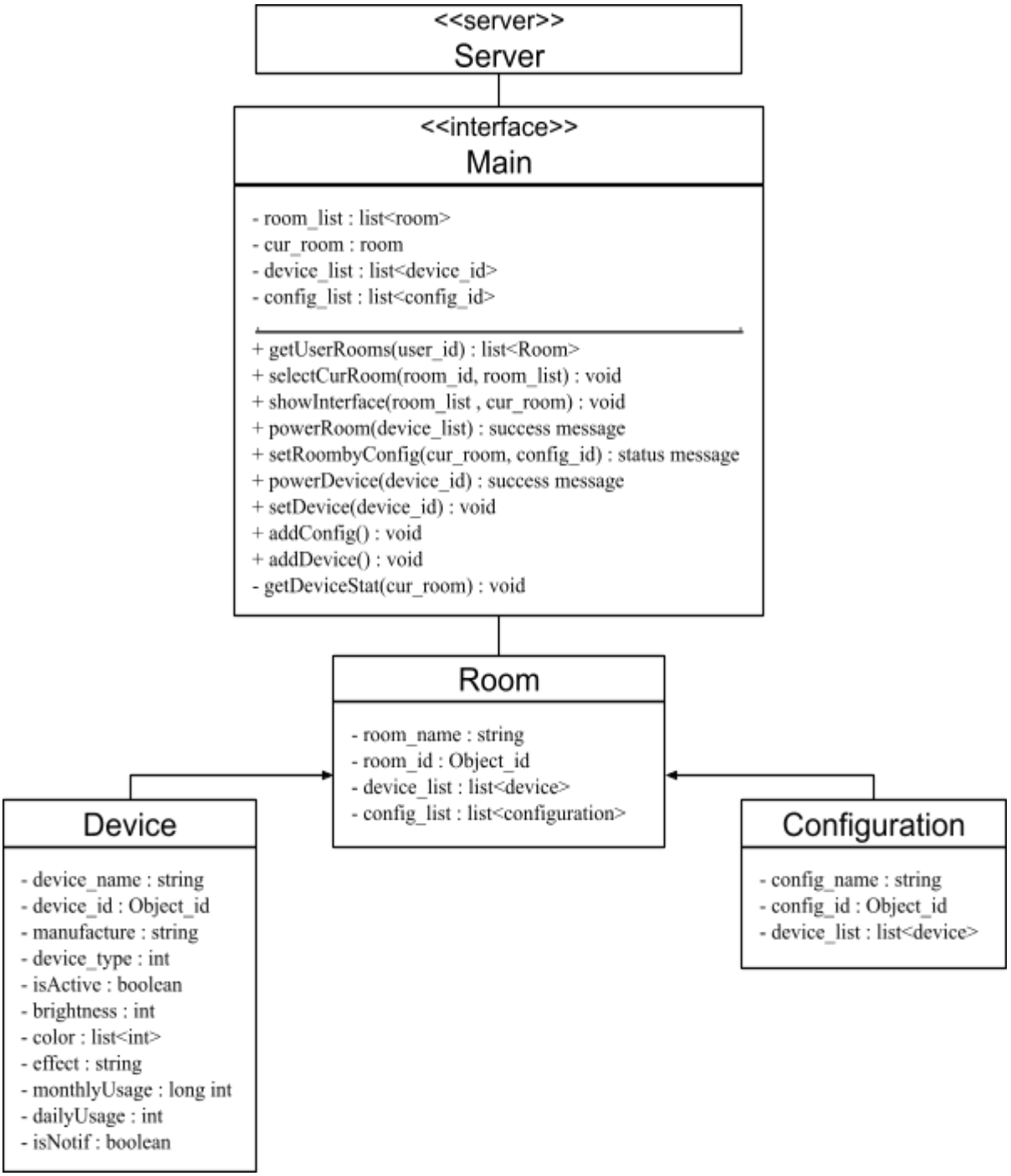
These are the methods that the main interface class has

- getUserRooms()
- selectCurRoom()

- showInterface()
- powerRoom()
- setRoombyConfig()
- powerDevice()
- setDevice()
- addConfig()
- addDevice()
- getDeviceStat()

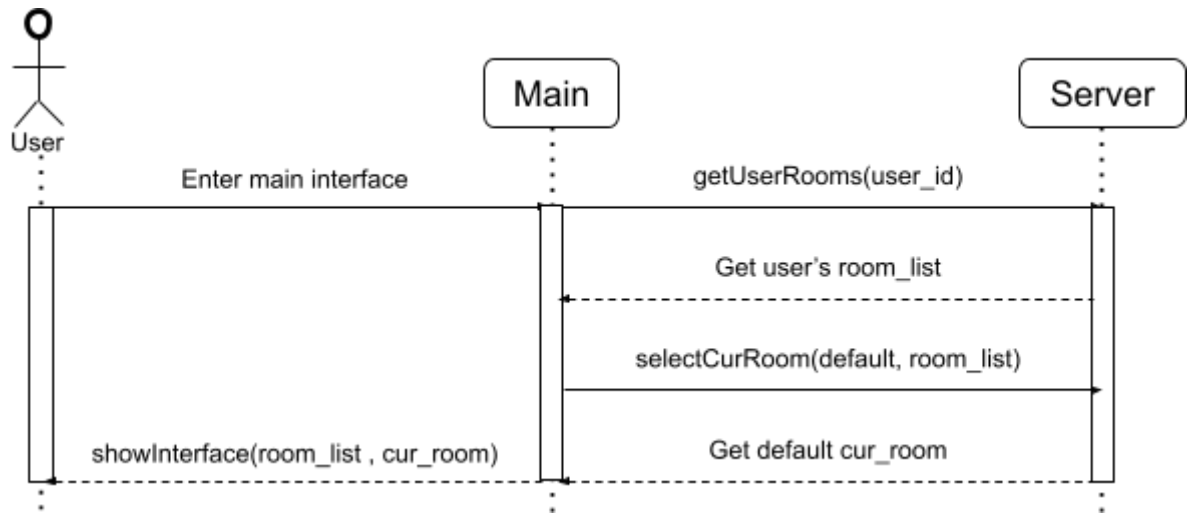
4.2.1.3. Class Diagram

[Figure 4] Class Diagram - Main



4.2.1.4. Sequence Diagram

[Figure 5] Sequence Diagram - Main



4.2.2. Setting Light Interface

The main interface class has information of all smart-house system's rooms and handles a user's request to the current room.

4.2.2.1. Attributes

These are the attributes that the setting light interface class has

- **device_name** : name of device that user interact with now
- **device_id** : id of device that user interacts with now. It is unique.
- **brightness** : brightness of device. percentage value compared to max
- **preset_color** : predefined RGB value.
- **current_effect** : current effect state of device's light.
- **available_effect** : list of available effects of device's light. If it does not support effect function, the value is empty list.
- **isNotif** : state(on / off) of notification option

These are the attributes that the device object has same as 1.1.2.1

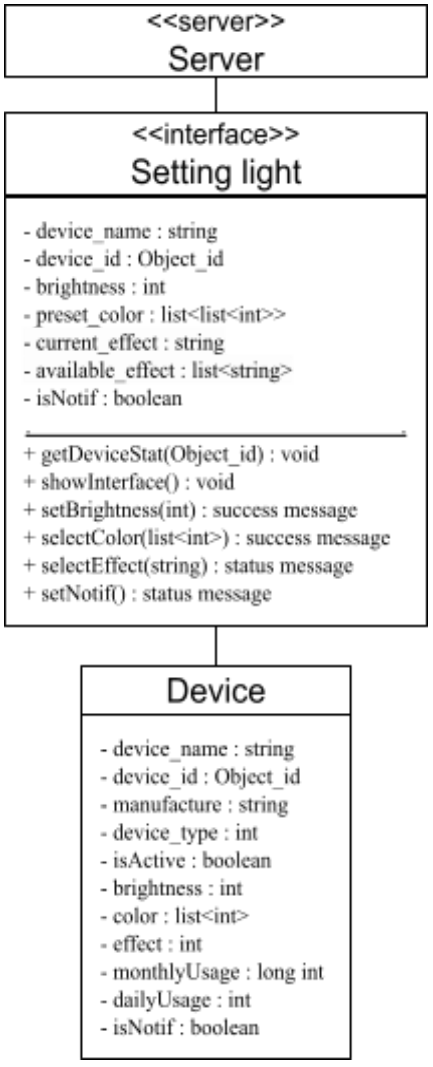
4.2.2.2. Methods

These are the methods that the setting light class has

- getDeviceStat()
- showInterface()
- setBrightness()
- selectColor()
- selectEffect()
- setNotif()

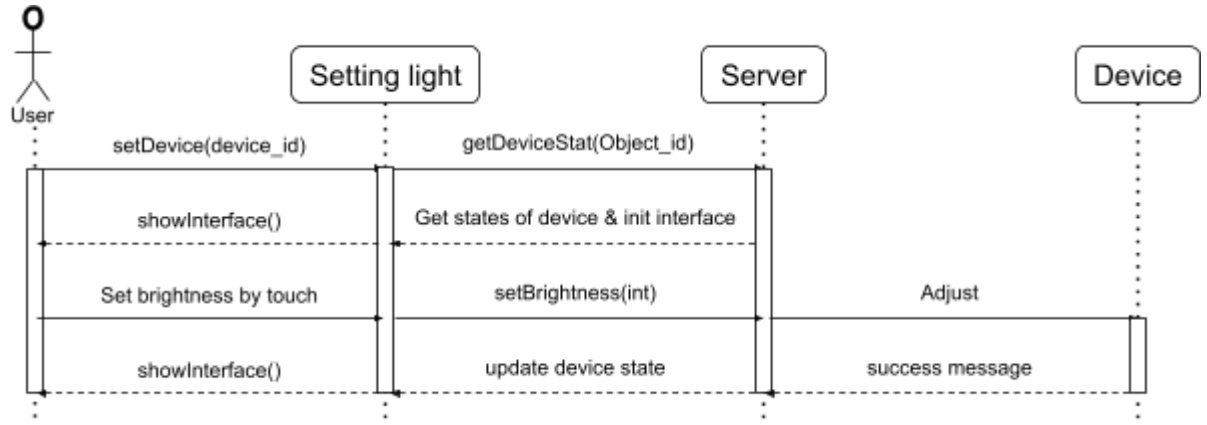
4.2.2.3. Class Diagram

[Figure 6] Class Diagram - Setting light



4.2.2.4. Sequence Diagram

[Figure 7] Sequence Diagram - Setting light



4.2.3. Configuration Interface

The configuration interface class make customized configurations that include icon image of new configuration and settings of devices

4.2.3.1. Attributes

These are the attributes that the configuration interface class has

- **config_name** : name of configuration. initial value is null
- **config_id** : id of configuration.it is unique.
- **preset_config_icons** : predefined list of icons
- **device_list** : list of device object

These are the attributes that the device object has same as 1.1.2.1

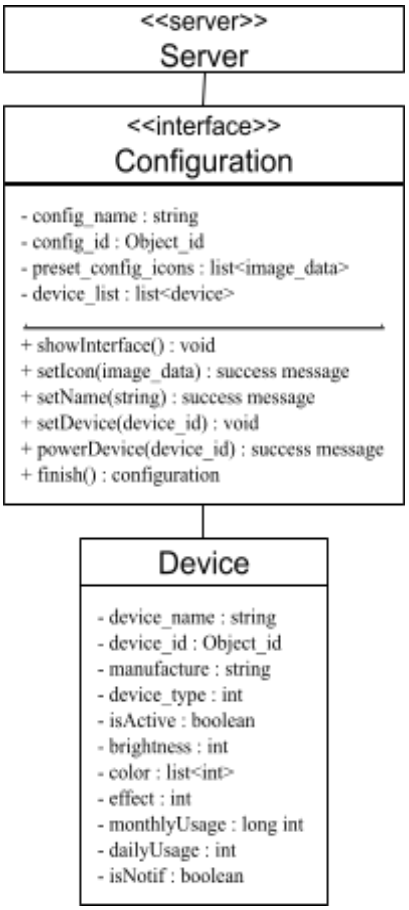
4.2.3.2. Methods

These are the methods that the configuration class has

- showInterface()
- setIcon()
- setName()
- setDevice()
- powerDevice()
- finish()

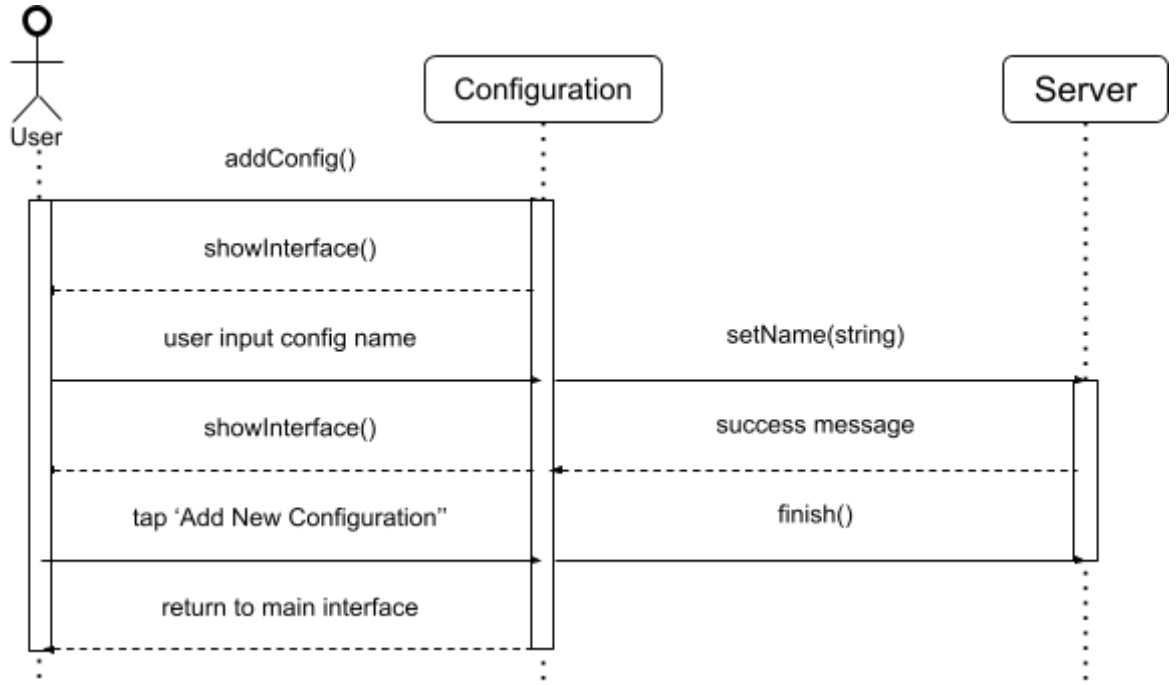
4.2.3.3. Class Diagram

[Figure 8] Class Diagram - Configuration



4.2.3.4. Sequence Diagram

[Figure 9] Sequence Diagram - Configuration



4.2.4. Electricity Usage Interface

The electricity usage interface class show user how much electricity is consumed for light system and how much it costs in daily/monthly.

4.2.4.1. Attributes

These are the attributes that the electricity usage interface class has

- **daily_usage**: daily usage of the whole energy of smart-house's light system.
- **monthly_usage_list** : list of monthly usage of the whole energy of smart-house's light system.
- **daily_cost**: daily cost of the whole energy of smart-house's light system.
- **monthly_cost** : monthly cost of the whole energy of smart-house's light system.

4.2.4.2. Methods

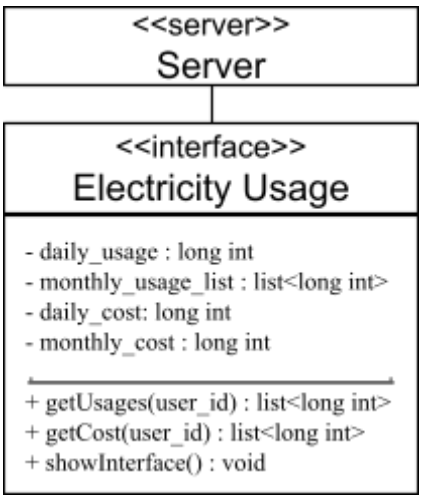
These are the methods that the electricity usage interface class has

- getUsages()

- getCost()
- showInterface()

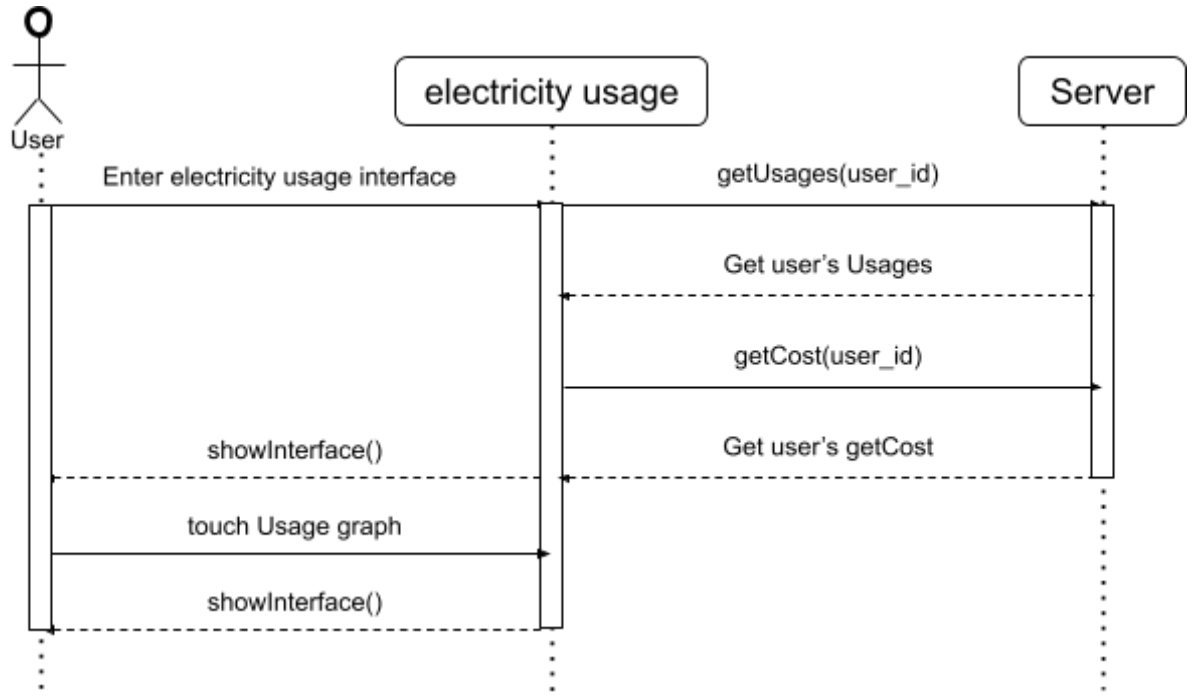
4.2.4.3. Class Diagram

[Figure 10] Class Diagram - Electricity Usage



4.2.4.4. Sequence Diagram

[Figure 11] Sequence Diagram - Electricity Usage



4.2.5. Setting Interface

The setting interface class shows user available functionality of the light system and handles a user's request to change functionality options and option values.

4.2.5.1. Attributes

These are the attributes that the setting interface class has

- **options** : options object that is a list of (on / off)state of toggle options.

These are the attributes that the options object has

- **isVisitorAlarm** : (on / off)state of function that alerts light alarm the user that there is a visitor by the lighting system.
- **isAuto** : (on / off)state of function that turn on/off automatically the light when the user enters/leaves a room.
- **isNotif** : (on / off)state of function that alerts light alarm the user by the lighting system when an app alarm goes off or the phone rings.
- **isSleepLight** : (on / off)state of function that turn the light on/off automatically when it is time to wake up/sleep.
- **isMaintainLight** : (on / off)state of function that adjust automatically the light to a certain brightness.
- **isSleepDetect** : (on / off)state of function that detect the user sleeping and automatically turn the light off gradually.
- **sleep_cycle** : time to wake up/sleep, if 'sleep light' option is on.
- **brightness** : brightness level to maintain if 'maintain fixed brightness' option is on.

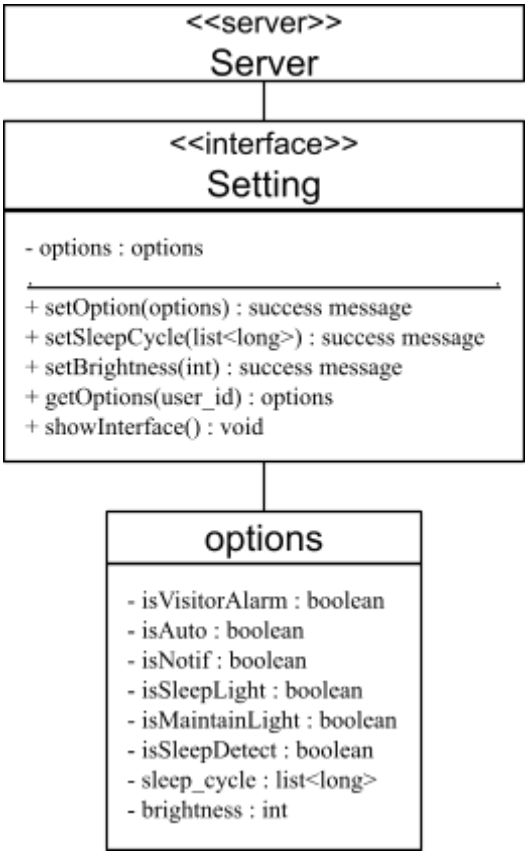
4.2.5.2. Methods

These are the methods that the setting interface class has

- setOption()
- setSleepCycle()
- setBrightness()
- getOptions()
- showInterface()

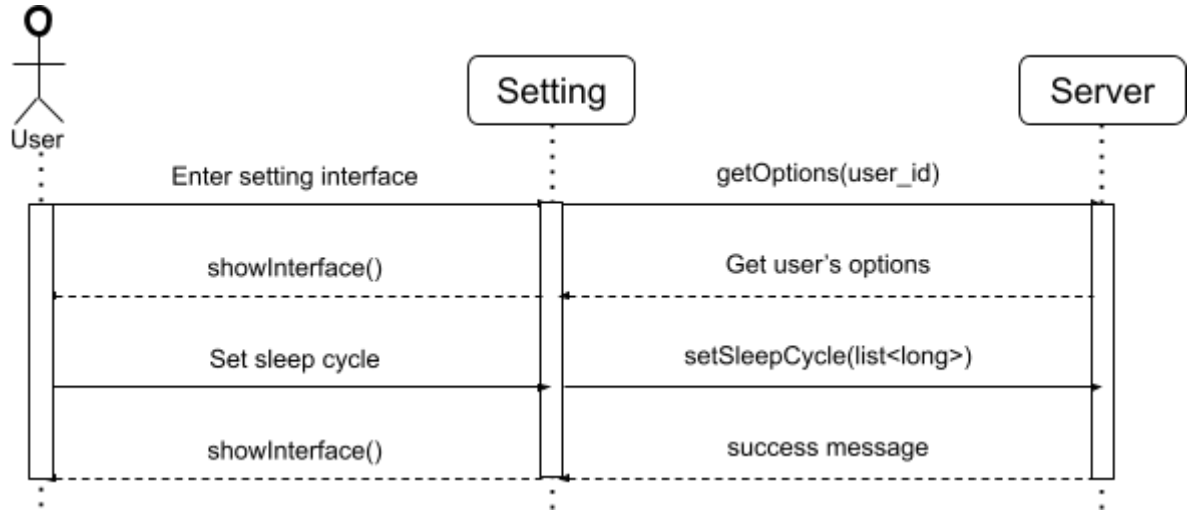
4.2.5.3. Class Diagram

[Figure 12] Class Diagram - Setting



4.2.5.4. Sequence Diagram

[Figure 13] Sequence Diagram - Setting



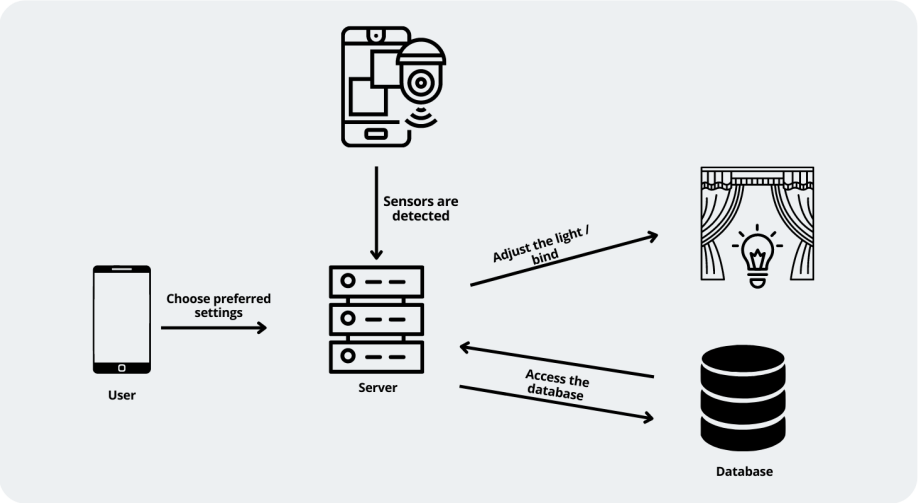
5. Backend System Architecture

5.1. Objectives

This chapter describes the structure of the backend system including the database in our project.

5.2. Overall Architecture

[Figure 14] Overall Architecture of system



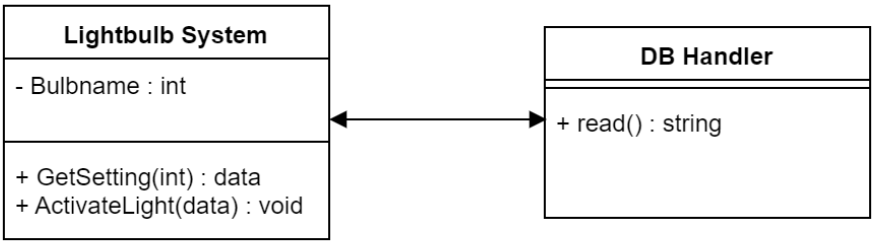
5.3. Subcomponents

5.3.1. Lightbulb System

Lightbulb system receives the setting information of the bulb that needs to be operated from the database, and activates the light bulb accordingly. The lightbulb system requests the data with the name of the light bulb, and the database returns the information on the light corresponding to the received value. Then, the system activates the light bulb according to the setting.

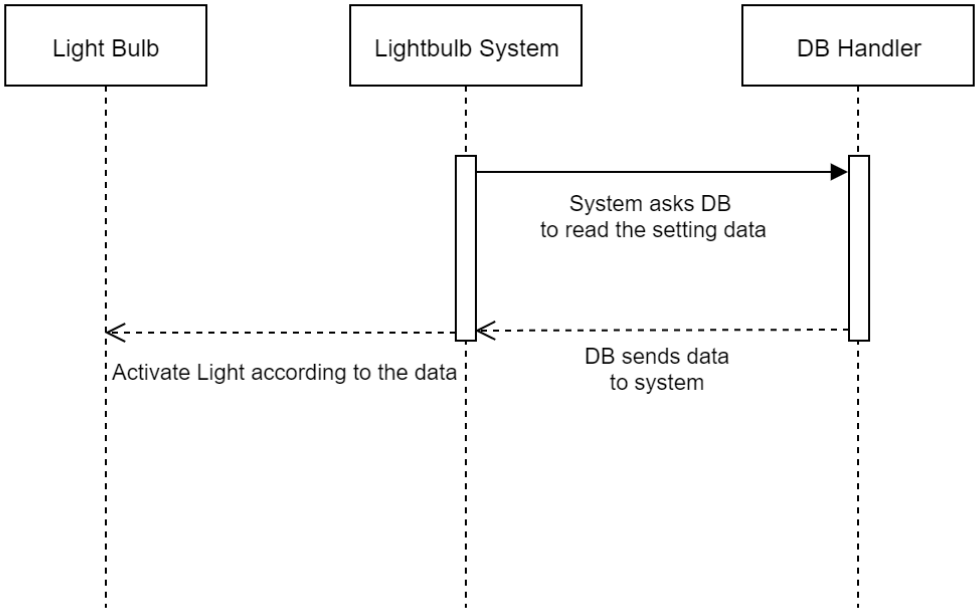
5.3.1.1. Class Diagram

[Figure 15] Class Diagram - Lightbulb System



5.3.1.2. Sequence Diagram

[Figure 16] Sequence Diagram - Lightbulb System

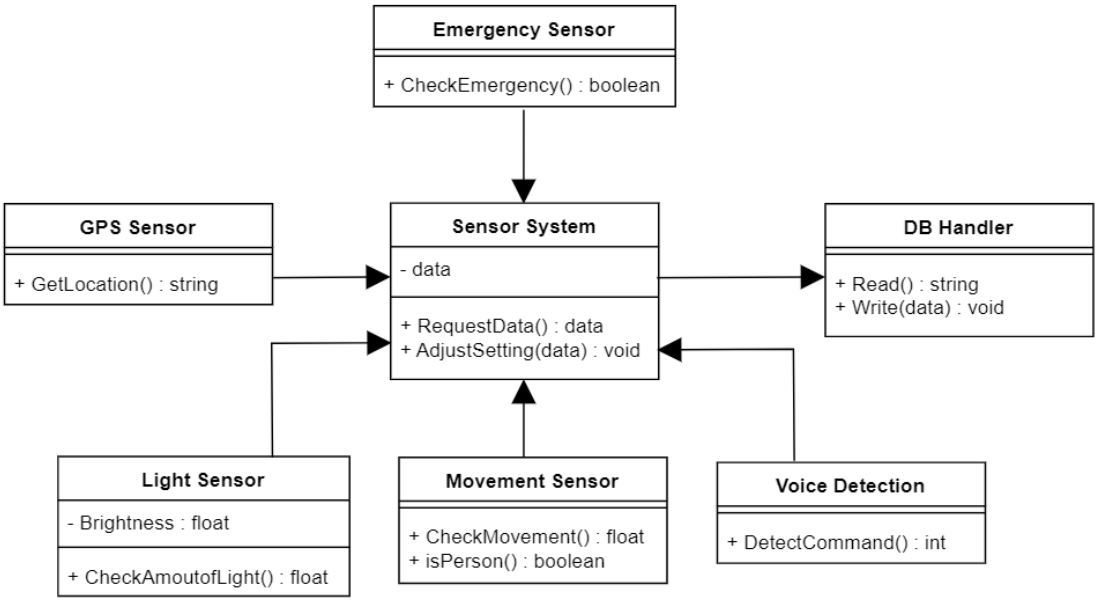


5.3.2. Sensor System

The sensor system detects the sensor and its type, then according to its type it writes the new data and gets the information from the database to see if any adjustments will be made or not.

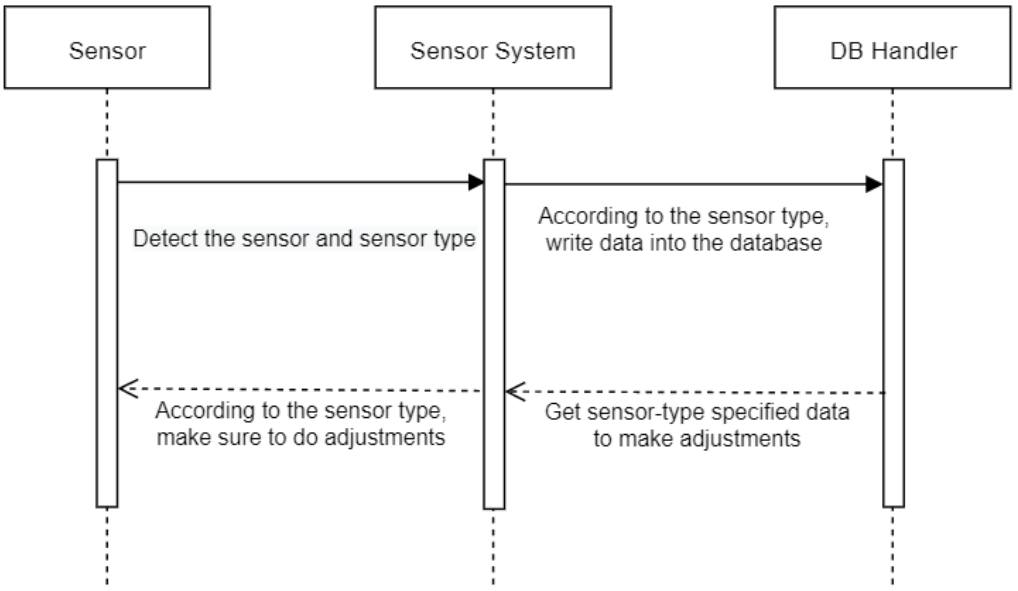
5.3.2.1. Class Diagram

[Figure 17] Class Diagram - Sensor System



5.3.2.2. Sequence Diagram

[Figure 18] Sequence Diagram - Sensor System

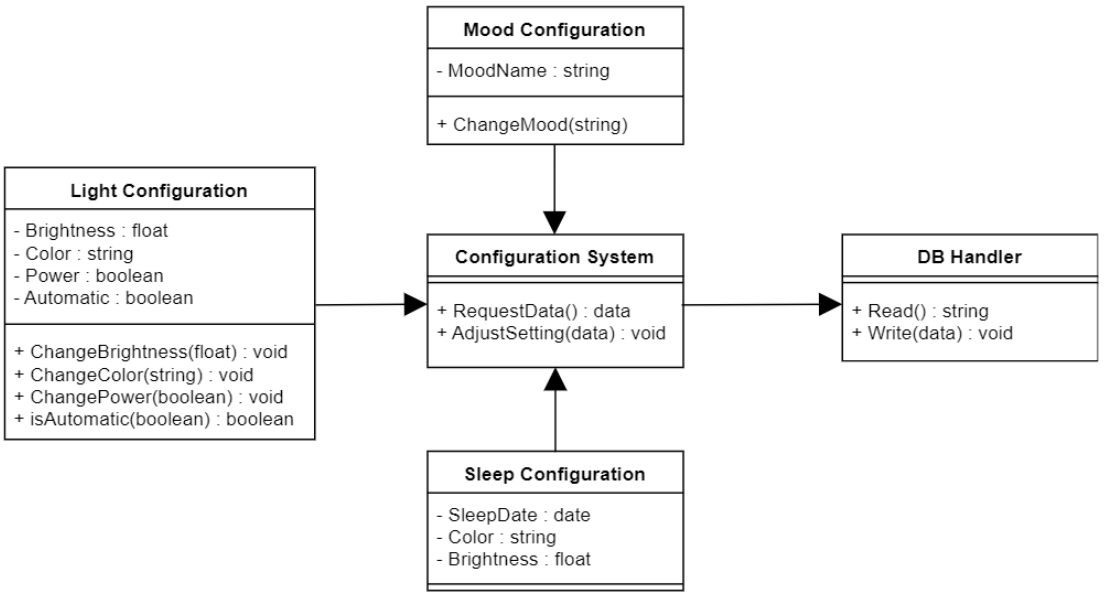


5.3.3. Configuration System

Configuration system deals with all configurations/changes that are made by the user. It receives the data, writes it to the database and reads it when it is needed.

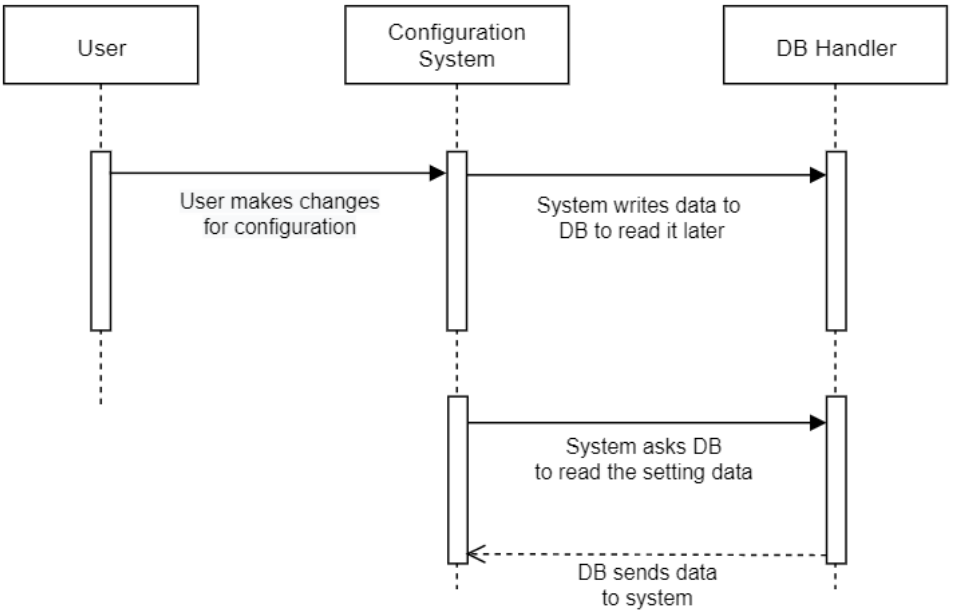
5.3.3.1. Class Diagram

[Figure 19] Class Diagram - Configuration System



5.3.3.2. Sequence Diagram

[Figure 20] Sequence Diagram - Configuration System

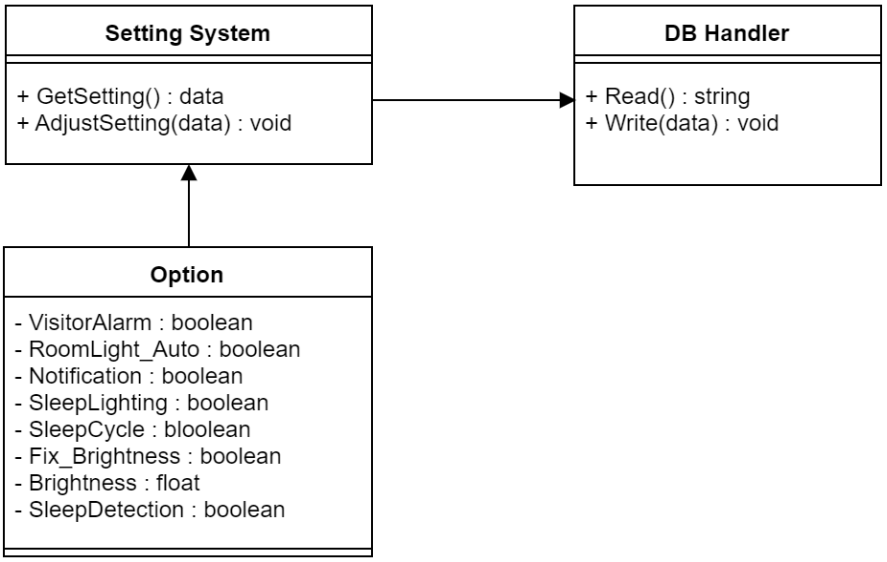


5.3.4. Setting System

Setting system receives whether to execute or not each option set by the user and stores it in the database. When the user sets each option, the system gets the information of options and sends it to the database for saving and updating the setting.

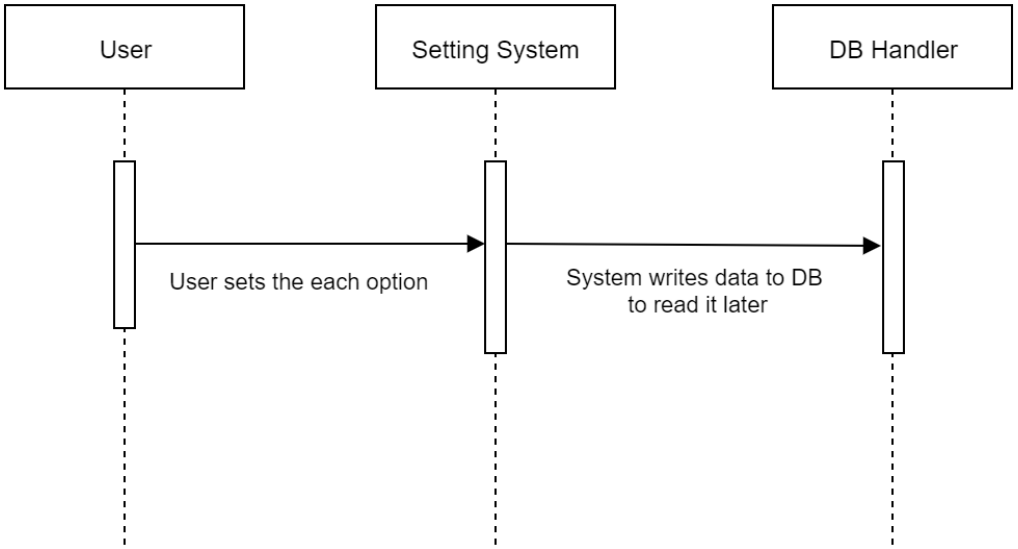
5.3.4.1. Class Diagram

[Figure 21] Class Diagram - Setting System



5.3.4.2. Sequence Diagram

[Figure 22] Sequence Diagram - Setting System



6. Protocol Design

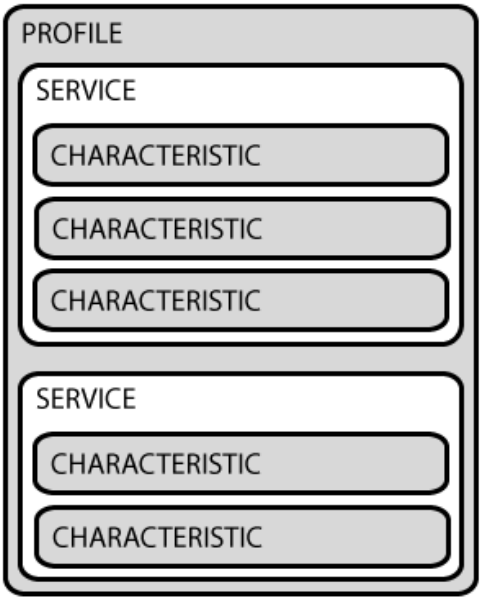
6.1. Objective

This chapter describes the protocols that are used in the communication between the interface and the server. This document will focus mainly on the actual data that is being transmitted.]

6.2. BLE

Our smart lighting system used BLE(Bluetooth Low Energy) to send data between the interface and the server. BLE is a near-range wireless communication technology. It divides the frequency band into several channels, then uses frequency hopping to avoid interference between signals. Once BLE devices are connected to each other and ready for data communication, they should now be aware of what each other can do and exchange data according to a specific situation. GATT(Generic Attribute Profile) abstracts data formats and procedures so that data can be sent and received in a consistent manner no matter what device it is connected to. GATT server keeps the services and data that it provides in the following hierarchy.

[Figure 23] Hierarchy of GATT



A profile is a conceptual distinction that indicates what a BLE device serving as a GATT server does. A service is a collection of data that is related to a particular function. A characteristic holds and manages the real data. The characteristic has three parts: declaration,

value, and descriptor. This document focuses on the characteristics values, which contains the actual data.

6.3. ATT

ATT can be said to be a specification for storing the concepts of service and characteristics defined in GATT in the form of data. ATT consists of four categories: Handle, Type, Permissions, Value.

- Handle: The value that the BLE device arbitrarily specifies to distinguish between several ATT entries. 16bit ID.
- Type(UUID): ID that not only identifies the service/characteristic but also informs each type. 2,4,8, or 16bit ID.
- Permission: Access rights to the corresponding ATT entry.
- Value: The actual data. It is limited to 512 bits.

6.3.1. Change Lighting Setting

[Table 1] ATT - Change Lighting Setting

Characteristic	Change Lighting Setting
Handle	0x8000
Type(UUID)	LIGHT(0x2803)
Value	Room# / Lightbulb # / Power / Brightness / Color / Effect / Alarm
Permissions	Read

6.3.2. Create New Configuration

This service contains several characteristics that contains information about the above lighting settings of each lightbulb.

[Table 2] ATT - Create New Configuration

Characteristic	Create New Configuration
Handle	0x8001
Type(UUID)	CONFIG(0x2A37)

Value	ConfigName / IconID / Room# / Lightbulb # / Power / Brightness / Color / Effect / Alarm
Permissions	Read

6.3.3. Toggle Settings

[Table 3] ATT - Toggle Settings

Characteristic	Toggle Settings
Handle	0x8001
Type(UUID)	SETTING(0x2A38)
Value	visitorAlarm(1 bit) / roomLightAutomation(1 bit) / notificationLighting(1bit) / sleepDetection(1 bit) / / sleepCycle(float 4 byte*2=8 byte) / fixedBrightness(1 bit) / Brightness(float 4 byte)
Permissions	Read

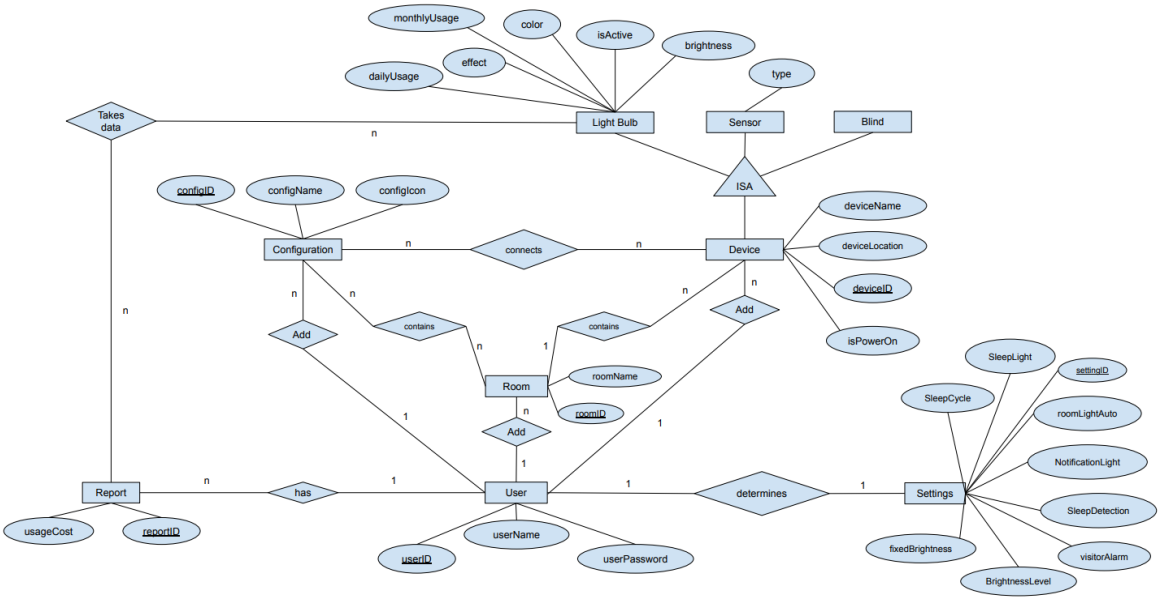
7. Database Design

7.1. Objectives

In this section, the database of the system is represented. Entities, attributes and relations are shown on the E-R diagram and then each entity is explained one by one. The cardinality of the relations between the entities are also indicated as 1 and n (1-1: one-to-one, 1-n: one-to-many, n-1: many-to-one).

7.2. ER Diagram

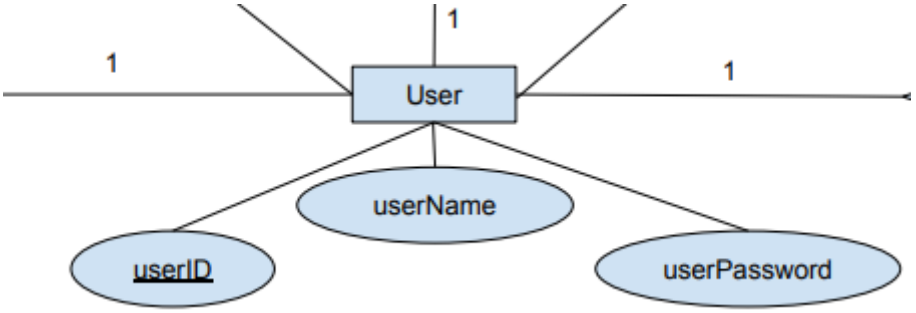
[Figure 24] ER Diagram - Overall



7.3. Entities

7.3.1. User

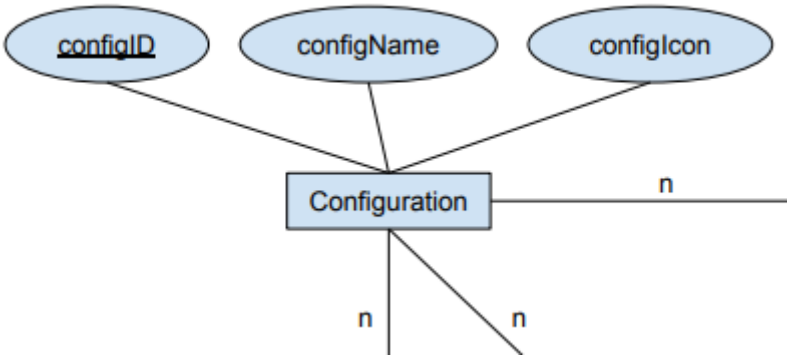
[Figure 25] ER Diagram - User



User entity represents the user of this system. Generally, the user of this system is the resident of the smart house. It consists of userID, userName, and userPassword. userID is a primary key. userName contains the name of the user. userPassword contains the password of the user which is needed when the user logs in.

7.3.2. Configuration

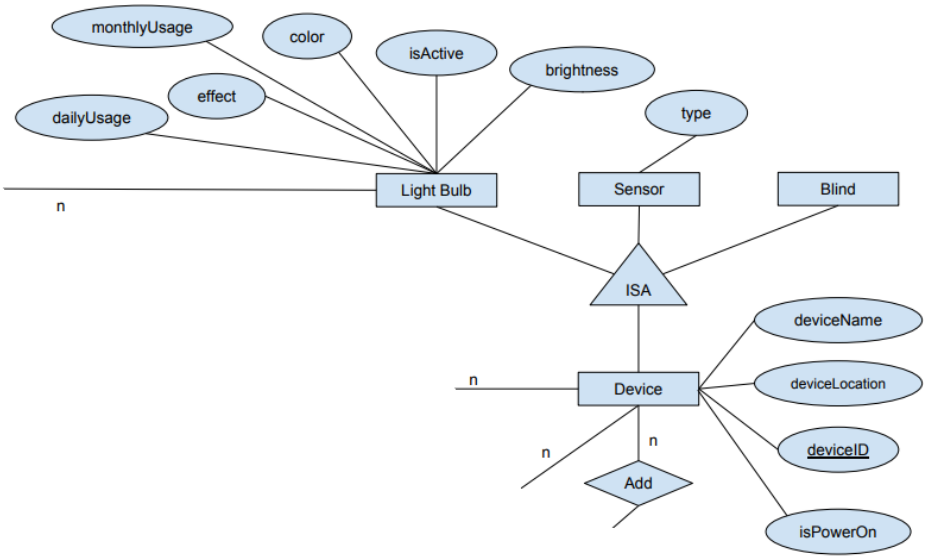
[Figure 26] ER Diagram - Configuration



Configuration entity represents the configuration of the light. Users can make and save a configuration of the lights and apply it to one of the lights. It consists of configID, configName, and configIcon. configID is a primary key. configName is the name of configurations. configIcon is the image that is shown in the UI.

7.3.3. Device

[Figure 27] ER Diagram - Device

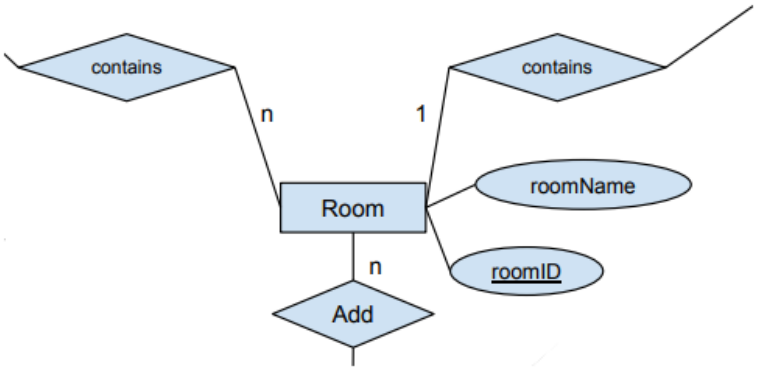


Device entity represents things that are connected to the system. Device consists of deviceName, deviceLocation, deviceID, isPowerOn. deviceID is a primary key. deviceName is the name of each device. deviceLocation is the location of each device. isPowerOn is

whether each device is on or off. Also there are 3 types of device, Light Bulb, Sensor, and Blind. Blind is blind which is automatically moved. Sensor consists of type. type is the kind of the sensor. Light Bulb consists of dailyUsage, monthlyUsage, effect, color, isActive, brightness. dailyUsage is the amount of the usage of the light for one day. monthlyUsage is the amount of the usage of the light for one month. effect is the effect of the light. color is the color of the light. isActive is whether the light is activated for notification. brightness is the amount of the light.

7.3.4. Room

[Figure 28] ER Diagram - Room

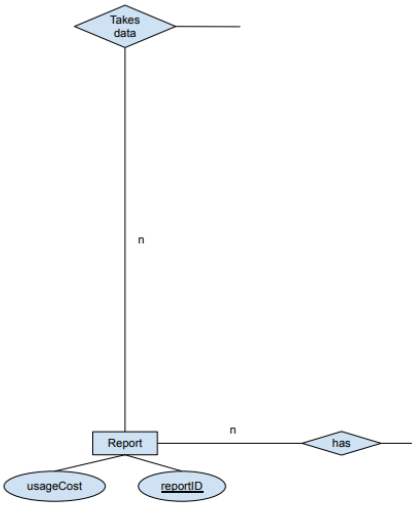


Room entity represents room in the house and system added by the user. The attributes of this entity are the roomID that defines the room and the name that will appear in the app is roomName. It is in a one-to-many relationship with the User. That is, the user can add more than one room, but these rooms can only belong to a single user. This means that the access of the rooms cannot be managed by other users. Also, Room contains relationships with entities 'Configuration' and 'Devices'. Room can contain more than one configuration and any configuration can be in more than one room. On the other hand, a room can contain more than one device, but that specific device can belong to only one room.

7.3.5. Report

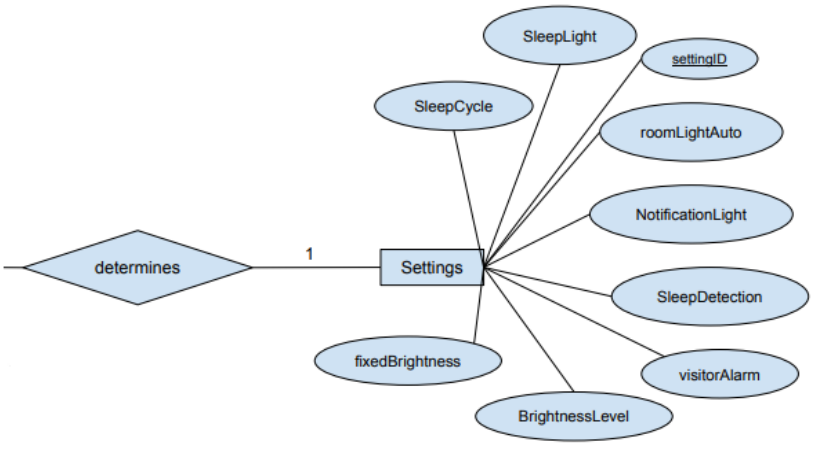
[Figure 29] ER Diagram - Report

The Report entity keeps the data to show the amount of electricity consumed and the amount resulting from this use to present to the user. It is in a 'has' relationship with the user, and a user can receive multiple reports while a customized report belongs to only one user. It is also in a 'takes data' relationship with Light Bulb, so it calculates usageCost using light bulb's attribute dailyUsage and monthlyUsage. Information from more than one light bulb can be used within the report, and any light bulb can also be included as an element of more than one report.



7.3.6. Settings

[Figure 30] ER Diagram - Setting



Settings entity holds the properties that the user has selected from the settings section. SleepLight, roomLightAuto, NotificationLight, SleepDetection, visitorAlarm, and fixedBrightness are boolean values and the user manages them as desired. BrightnessLevel and SleepCycle are double values and the user is expected to enter a numeric input value. It is

in a 'determines' relationship with the user and the user can manage a single set of settings, while the specific settings ID can only be accessed and edited by a single user.

8. Testing Plan

8.1. Objectives

Testing is an important stage of software development. This section basically focuses on three different tests: development testing, release testing and user testing. The reason why testing is important in software development is to test periodically whether the work done is correct or not. Thus, errors can be detected before the application is released, and can be prevented without making it more expensive and dangerous.

8.2. Testing Policy

8.2.1. Development Testing

Development testing is testing in a broad sense with the aim of integrating the development phase and testing phase of the program. In general, it aims to detect errors in the program and start to prevent them early. Therefore, it is an important step for the progress of the implementation. It is used to check whether the design specifications are implemented correctly. It provides efficiency in software development. In this test, topics such as performance, reliability and security are examined.

8.2.1.1. Performance

Normally, we can turn on and off a light system in our house directly without any need, and this is a situation that takes place within seconds. One of the challenges of a 'Smart' system is making sure it works quickly and easily as regular ones. Because the user will not want to wait for a slow system return. Therefore, performance is one of the most important factors in this project and devices must be well connected to the app. In addition, one of the goals of the Smart Lighting project is energy conservation. Although this applies to savings, it is desirable to minimize the extra use of computing resources when using this system. Considering that many devices will be combined in one system, good performance is one of the requirements

of the project. Therefore, tests are prepared that measure the speed of the system and the strength of its capacity.

8.2.1.2. Reliability

Reliability is also one of the most important building blocks of this project. First of all, it can be said that any error in the system will reduce the quality of life. For example, the user is sitting at home and has an important job, and the lights turn off due to a system error, a bug, or the lights turn on and off while sleeping for the same reason. No user wants this and such errors reduce the reliability of the system. In addition, if the devices of a large-scale system connected to a large structure such as a house are thought to be connected to an app, the security of the house and the user will be in danger due to a faulty bug or in case of malicious software or malicious access by others. Therefore, it is important that the program runs without errors so that the user can trust this system and allow access to the house by the system. The system should be checked systematically and regularly with unit tests.

8.2.1.3. Security

Software security is an important step in understanding and preventing vulnerabilities. In the Smart Lighting project, the security of the home is, in a way, tied to app access. Therefore, ensuring software security ensures that the house is also safe. If there is a vulnerability that can be managed by other users, it will be a very serious problem. The project needs to be within a multi-faceted security framework. The secure operation of the project also increases reliability. Since information belonging to each user and their home is also kept, it is important that this information is securely encrypted.

8.2.2. Release Testing

Release testing is making a part of the program available to people outside the developer team. With this method, an example is given to the customer and feedback can be received whether he likes it or not. For this testing, it is not necessary to complete the program completely, on the contrary, the purpose here is to show the user after making minor improvements, to receive feedback and to continue improvements and corrections in line with these feedbacks. It is important that the deficiencies are corrected in each release and that the

user feels the development. With this testing, the program is planned to be carried out in small but frequent stages and in a controlled manner.

8.2.3. User Testing

Usability and the system being user-friendly in general are important for the continuity of the application and for understanding the usefulness of the program. User testing has an important place in measuring how the system will react in a real situation and shows the results of real-life situations. First of all, we aim to present the system in an offline structure connected to only one user, without networking, and observing the user usage. We aim to release the beta version so that all users' data can be collected and managed through an admin cloud system. Thus, it is measured how the system will respond to user inputs and usage in various cases.

8.2.4. Testing Case

Testing cases will be created within the scope of the metrics specified in 8.2.1. These are performance, reliability and security. With test cases, the program's reactions to corner cases are understandable and it is aimed to prevent possible errors that are difficult to predict. With various test cases, it is observed whether the program fulfills the design specifications and different functions correctly.

9. Development Plan

9.1. Objectives

This chapter explains the technologies and environment used for the development of the application and the system.

9.2. Frontend Environment

9.2.1. Figma (UI / UX Design)

[Figure 31] Figma logo

With Figma, designers can build dynamic prototypes, test them for usability, and sync up all of the progress. It also makes it easier for groups to work together by allowing a collaborative environment where designers can work at the same time. It would be a nice start to see the most basic version of our application design.

9.2.2. Adobe XD (UI / UX Design)

[Figure 32] Adobe XD logo

Adobe XD is one of the UI/UX design tools for websites, apps and more. It helps to create prototypes that you can interact with. Since it can be used both on Mac and Windows systems, most designers can access it and work together on the project. It would be nice to see how the application can see at the end of the designing process more realistically while interacting.

9.2.3. Android Studio (Application)

[Figure 33] Android Studio logo

Android Studio is the official IDE for Android application development that is based on IntelliJ IDEA. It has built-in support for Google Cloud Platform which makes it easy to integrate Google Cloud Messaging and App Engine. Also, since it is possible to find sample codes, it could make the team's starting process way easier.

9.2.4. Xcode (Application)

[Figure 34] xcode logo

Xcode is Apple's IDE for macOS that can be used for developing software for iOS, iPadOS and more. It also supports source code for several programming languages, and developers can see their teammates' comments inside their code with a collaboration with GitHub. It would be nice to use Xcode for better iOS development of the application.

9.3. Backend Environment

9.3.1. GitHub (Open Source)

[Figure 35] GitHub logo

GitHub is a code hosting platform for version control and collaboration. Developers all around the world can work together on the projects, and can access many open source codes.

9.3.2. MySQL (Database)

[Figure 36] MySQL logo

MySQL is a DBMS is a client/server system that consists of a multithreaded SQL server that supports different back ends and several client programs.Moreover, because it is relational, it separates tables rather than putting all data in one big storeroom.

9.3.3. FireBase (Server)

[Figure 37] Firebase logo

Firebase is a Google-backed application development software that enables developers to develop iOS, Android and web apps. Firebase Cloud can be used as a server and datastore as well. It can be a good choice for large data sets because NoSQL scales data horizontally, and faster.

9.4. Constraints

This application and the system will be designed and implemented based on the information mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but comply with the following:

- Motion detecting should be for humans only, not animals
- Use the technology that has already been widely proven
- Avoid using software that requires a separate license
  (Exclude, if this is the only technology or software that the system must require.)
- Decide in a more user-friendly way
- Consider overall system performance
- Consider the system and maintenance cost
- Use optimized source code to prevent waste of system resources
- Consider future maintenance and use sufficient comments in the source code
- Develop using the latest version of Android Studio as 2021.1.1.23
- Test operation of the product using Android OS 10.0 and iOS 13

9.5. Assumptions and Dependencies

This document was written on the assumption that the product is designed for smartphone devices that have iOS or Android OS. Therefore, all content is based on the 10.0 version for

Android OS, and 13.0 version for iOS and, it may not apply to other operating systems or versions.

10. Supporting Information

10.1. Software Design Specification

This software design specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016).

10.2. Document History

[Table 4] Document History

Data	Version	Description	Writer
2022.05.10	0.1	Style and Overview	황정윤
2022.05.11	1.0	Addition of 7	Aysun Ogut, 정성욱
2022.05.11	1.1	Addition of 5.1-5.3.2	Selin Samra, 이채은
2022.05.11	1.2	Addition of 1	Aysun Ogut
2022.05.12	1.3	Addition of 5.3.3-5.3.4	Selin Samra, 이채은
2022.05.12	1.4	Addition of 4	유상범
2022.05.12	1.5	Addition of 3	황정윤
2022.05.12	1.6	Addition of 8	Aysun Ogut
2022.05.12	1.7	Addition of 9	Selin Samra
2022.05.13	1.8	Addition of 6	황정윤
2022.05.13	1.9	Addition of 2	이채은
2022.05.14	1.10	Revision of Structure	유상범, 정성욱