Smart Chandelier

Color every moment

- Software Design Description (SDD) -

TEAM 1

	Gyeongsu Kim
	Daeun Kim
	Hyuckjoong Yoon
	Sana Kang

Table Of Contents

1. System Overview

The “Smart Chandelier” is a smart IoT system that serves a special mood to the users through the light. The product consists of Smart Chandelier lighting modules, supplementary sensor modules, optional hardware, and a central hub.

The lighting module provides a powerful lighting service that covers the entire indoor area. The Smart Chandelier replaces the main lighting in the living room, so it provides various functions including simple lighting, colorful effects, and lighting supervision. The target users include general households, gym owners, cafe owners, and etc.

This product is expected to be used as a way to replace or upgrade the main lighting in customers' home automation as a high-quality light. It’s the reason that this product pursued convenience while incorporating necessary functions for actual use as much as possible. It provides adequate lighting without the user’s care, and can also be customized if needed.

A. Requirements

Functional Requirements		
R.ID	**Requirement Description**	**Dependencies/Assumptions**
R.F.A-1	The web app should allow users to create user accounts.	
R.F.A-2	The web app should allow users to register an user account to a hub.	R.F.A-1
R.F.A-3	The web app should allow users to login to a registered hub.	R.F.A-1
R.F.A-4	The web app should allow users to delete an user account from a hub.	R.F.A-1, R.F.A-2
R.F.A-5	The web app should allow users to modify user account information.	R.F.A-1
R.F.A-6	The web app should allow users to change the account passwords.	R.F.A-1, R.F.A-5
R.F.A-7	The web app should allow users to find the account PIN when they forget it.	R.F.A-1
R.F.A-8	The web app should allow users to reset the account PIN.	R.F.A-1, R.F.A-5
R.F.B-1	Modules should be able to be registered to a hub only through the Web app.	
R.F.B-2	Modules should be able to be deleted from a hub only through the Web app.	R.F.B-1
R.F.B-3	Module settings should be modified only through the Web app.	R.F.B-1
R.F.B-4	Module history should be able to be shown to users.	

R.F.B-5	Module history should be able to be deleted.	
R.F.B-6	Registered module list should be given to users.	R.F.B-1, 2
R.F.B-7	The web app should allow the user to set the timer for the module.	R.F.B-1, 3
R.F.C-1	Events should be able to be added to a lighting module.	
R.F.C-2	Events should be able to be modified.	R.F.C-1
R.F.C-3	Events should be able to be deleted from a lighting module.	R.F.C-1
R.F.D-1	Module power should be able to be controlled remotely.	Raspberry Pi
R.F.D-2	Module mode should be able to be changed remotely (auto / manual mode).	
R.F.D-3	Module state should be able to be shown to users.	
R.F.E-1	Lighting modules should allow users to set a lighting pattern manually.	
R.F.F-1	Lighting modules should allow users to set a lighting pattern automatically.	
R.F.F-2	The web app should allow people to write feedback.	
R.F.G-1	The web app should allow people to upload a lighting pattern to the community.	R.F.E-1
R.F.G-2	The web app should allow people to download a lighting pattern to the community.	

Non-Functional Requirements	
R.ID	**Requirement Description**
R.N.P-1	The average TTFB speed is no more than 150ms on the desktop.
R.N.P-2	The average TTFB speed is no more than 200ms on mobile.
R.N.P-3	The average Speed Index speed is no more than 2 seconds on desktop with a total 5000 simultaneous users.
R.N.P-4	The average Speed Index speed is no more than 2.5 seconds on mobile with a total 5000 simultaneous users.
R.N.P-5	The average size of a website is no more than 0.8 Mb for desktop
R.N.P-6	The average size of a website is no more than 0.6 Mb for Mobile
R.N.P-7	The average response time is no more than 300ms on the desktop.
R.N.P-8	The average response time is no more than 400ms on the mobile app.

R.N.P-9	The playing time of each lighting pattern file should be less than 8 hours per unit.
R.N.P-10	The user can upload three lighting pattern files at a time.
R.N.R-1	The system should be able to manage one million users without degradation in performance.
R.N.R-2	Applicants have complete access to the community 99 percent of the time.
R.N.R-3	A download for the same item might be requested by up to 500 applicants.
R.N.R-4	The system will retain mobile backups of all database updates for each record transaction to ensure data integrity.
R.N.A-1	Users can post 80 lightning patterns on the website throughout the week at any time during the day.
R.N.A-2	One user can download 30 lighting patterns in an hour.
R.N.A-3	When one user downloads a lighting pattern on a specific device, another user cannot download the pattern on the same device.
R.N.A-4	Users can choose their profile image among 15 default images.
R.N.S-1	Users can add an uploader to their Favorites list. The maximum number of Favorites listed is 60.
R.N.S-2	The user gets notified when an uploader on their favorite list uploads a new pattern.
R.N.E-1	If the user community service becomes unavailable, it may be inaccessible for up to three hours while it is being maintained.
R.N.E-2	If a temporary error occurs, a notice instructing users to reload the site as well as an apology are displayed.
R.N.H-1	Noise levels between 35 dB pl and 90 dB pl are measured in 1 dB pl increments.
R.N.H-2	Temperatures ranging from -40 to 80 degrees of celsius are measured in 1 degree increments, with an error range of 0.5 degrees. The frequency of data measurement is one hertz (Hz).
R.N.H-3	Humidity ranging from 0 to 100 percent is measured, with an error range of 2~5 percent. The frequency of data measurement is one hertz (Hz).
R.N.H-4	The operating time is limited to less than 2 seconds. The range of detectable distance is from 0 to 12 meters. Additionally, it can detect between 0.2 m/s and 1.3 m/s of speed when the object moves more than 0.5m. Moreover, it should be installed at a height of 1.8 m to 3 m.
R.N.H-5	Lighting Modules can fit into covers of various designs. The cover types are significantly different in shape, color, and pattern.
R.N.D-1	The user should read the manual documentation before beginning the service, which should be 5 pages for the mobile app and 3 pages for the web service.
R.N.D-2	The manual documentation appears only when users launch the app or log in the website for the first time. Users can read the documentation whenever they want in the “FAQ” category.

R.N.D-3	The manual explains how to create, apply, and submit a pattern, as well as how to apply particular criteria to motions.
R.N.D-4	The manual's font is at least 15pt, and numerous illustrations are included to make it simpler to read for people of all ages.
R.N.D-5	The user should read the terms and conditions documentation while signing up, which should be 3 pages for the mobile app and 5 pages for the web service.
R.N.D-6	The manual's font is at least 9pt.
R.N.E-1	The software system is accessible via a mobile app, a mobile webpage, and a PC webpage.
R.N.E-2	Users may use as many Smart Chandeliers as they want. However, at least 8 square meters of space per Smart chandelier must be guaranteed. Users can purchase extra sensors separately and install them anywhere they want.
R.N.S-1	Only site administrators can view the applicant’s personal information.
R.N.S-2	Users may protect their history of patterns played and preferences with a password.

B. Tasks

1) User Account Management

Task ID	Task Description	Related Req(s).
T.1	Implement “Create a user account” functionality.	R.F.A-1, R.N.S-1, R.N.R-1
T.1.1	The system should show the terms and conditions pop-up.	R.N.D-5,6
T.1.1.1	Implement “Consent to the terms and conditions” functionality.	
T.1.2	Implement “Authenticate and Active a user account” functionality.	R.F.A-1, 5
T.1.2.1	Implement “Authenticate by SNS account” functionality.	
T.1.2.2	Implement “Authenticate by your phone number” functionality.	
T.1.2.3	Implement “Authenticate by your email” functionality.	
T.2	Implement “Set the account information” functionality.	
T.2.1	Implement “Set user ID and community nickname” functionality.	
T.2.2	Implement “Login with touchID/FaceID/PIN’ functionality.	
T.2.3	Implement “Register your profile image” functionality.	
T.2.3.1	The system should offer 15 kinds of default profile images.	R.N.A-4
T.3	Implement “Finish the user registration” functionality.	R.F.A-2, R.N.S-1
T.3.1	Create a user account information object and send it to the database.	

T.3.2	The system shows the manual documentation right after the registration process.	R.N.D-1,2,3,4
T.3.2.1	Implement “Close the manual and go back to the main page” functionality.	
T.4	Implement “Login” functionality.	R.F.A-3
T.4.1	The system should create the makeLoginKey object when the login attempt occurs.	
T.4.2	The system should check whether the specific user with the input makeLoginKey exists in the database.	
T.4.3	The system should verify the user input of Faceid/ Touchi/ Pin is valid.	
T.4.3.1	The system should reload the homepage when the verification process returns True.	
T.4.3.2	The system should pop up the error message when the verification process returns False.	R.N.E-2
T.4.3.3	The system should destroy the loginKey object after the login process is done.	R.F.A-1
T.5	Implement “Forgot the PIN” functionality.	R.F.A-7, 8
T.5.1	Implement “Find the PIN” functionality.	
T.5.2	Implement “Reset the PIN” functionality.	
T.6	Implement “Change the user information” functionality.	R.F.A-5,6, R.N.S-1
T.6.1	Implement “Change user nickname” functionality.	
T.6.2	Implement “Change user phone number” functionality.	
T.6.3	Implement “Change user PIN” functionality.	
T.7	Implement “Delete account” functionality	R.F.A-4, R.N.S-1
T.7.1	Implement “retrieve the user list” functionality.	
T.7.2	The system should leave the post previously posted by the user as it is and change the user information to “Unknown.”	
T.7.3	The system should delete the user from the registered hub.	
T.7.4	The system should delete the user from the database.	

2) Module Management

Task ID	Task Description	Related Req(s).
T.1	Implement “Find registered modules” functionality.	R.F.B-6
T.1.1	The Web app should access the database to check whether the	

	requested module exists.	
T.2	Implement “Turn on/off the module” functionality.	R.F.D-1
T.2.1	The SmartThings Cloud should set power to the hub.	
T.2.2	The hub should set power to the requested module.	
T.3	Implement “Set the timer” functionality.	R.F.B-7
T.3.1	The SmartThings cloud should require the hub to set the timer.	
T.3.2	The hub should set the timer of the module.	
T.4	Implement “Register the module” functionality.	R.F.B-1, R.N.E-2
T.4.1	Implement “Register lighting module” functionality.	
T.4.2	Implement “Register sensor module” functionality.	
T.4.3	Implement “Set the name of the module” functionality.	
T.4.4	Implement “Double-check the installation location of the module.”	
T.4.4.1	The system should open the “Virtual Room” or make a new one depending on the installation location.	
T.4.4.2	The system should warn the user if the same kind of module already exists in the virtual room.	
T.5	The system should create a module information object and send it to the database.	R.F.B-1
T.6	Implement “Change the state of module” functionality.	R.F.B-3, D-3
T.6.1	Implement “Run the module” functionality.	
T.6.2	Implement “Deactivate the module” functionality.	
T.6.2.1	The system should bring the data from the weather API if the temperature/humidity sensor is deactivated.	-
T.6.3	Implement “Change the module mode” functionality.	R.F.D-2
T.6.4	Implement “Delete the module” functionality.	R.F.B-2
T.6.4.1	Implement “Retrieve connected modules list” functionality.	
T.6.4.2	The system should delete the module object from the database.	
T.7	Implement “Change the module information” functionality.	R.F.D-3
T.7.1	Implement “Change the module name” functionality.	
T.8	Implement “Read module history” functionality.	R.F.B-4
T.8.1	The system allows users to see the history up to a year ago.	
T.8.2	The system allows users to see the history divided by user	

	accounts.	
T.8.3	Implement “See light patterns made by the user” functionality.	
T.8.4	Implement “See the mostly used patterns Top 5” functionality.	
T.8.5	Implement “Load the sensor data” functionality.	
T.9	Implement “Delete the module history” functionality.	R.F.B-5, R.N.S-2
T.9.1	The system should lock the module history of each account.	
T.9.2	Implement “Calibrate the sensor” functionality.	R.N.H-1, 2, 3, 4, 5
T.10.1	Implement “Check the threshold value of sensor modules” functionality.	
T.10.2	Implement “Change the threshold” functionality.	

3) Event Management

Task ID	Task Description	Related Req(s).
T.1	Implement “Add a new event” functionality.	R.F.C-1
T.1.1	The system should check the database whether the connected hub exists.	
T.1.2	Implement “Set conditions to trigger the event” functionality.	
T.1.3	Inherit “Bring history” functionality to set event’s lighting pattern .	
T.1.4	Inherit “Set the lighting pattern” functionality to set event’s lighting pattern .	
T.1.5	Implement “Set the event name” functionality.	
T.1.6	The system should create an event object and send it to the database and the cloud system.	
T.2	Implement “Edit the event” functionality.	R.F.C-2
T.2.1	Inherit all detailed functionalities of “Add Event”’ functionality.	
T.2.2	Implement “Activate the event” functionality.	
T.3	Implement “See the event list” functionality.	
T.4	Implement “Delete event” functionality.	R.F.C-3

4) Manual Mode

Task ID	Task Description	Related Req(s).
T.1	Implement “Create a manual mode” functionality.	R.F.E-1

T.1.1	Implement “Set the lighting pattern” functionality.	
T.1.2	Implement “Set the module name” functionality.	
T.1.3	Implement “Set the light color” functionality.	
T.1.3.1	The user should specify the exact color from the color plate.	
T.1.3.2	Implement “See the color choice recommendation”’ functionality.	
T.1.4	Implement “Set advanced control” functionality.	
T.1.4.1	Implement “Set the gradation effect” functionality.	
T.1.4.2	Implement “Set the flash effect” functionality.	
T.1.4.3	Implement “Set the frequency of the light variation” functionality.	
T.1.5	The system should save conditions and actions of the mode in the database and the lighting module.	
T.2	Inherit “Set the timer” functionality to set the duration time of the lighting pattern.	
T.3	The system should offer the user community.	R.F.G-1, R.N.E-1, R.N.R-2,3 R.N.A-1,2,3
T.3.1	Implement “Download the pattern” functionality.	R.F.G-2
T.3.1.1	The system should save the downloaded pattern to the database and lighting module.	
T.3.2	Implement “Load the downloaded pattern” functionality.	
T.3.3	Implement “Upload the pattern” functionality.	R.N.P-9, 10
T.3.3.1	The system should create the lighting pattern object and send it to the community server. The object contains maker info, and pattern info.	
T.3.4	Implement “Rate the lighting pattern in the community” functionality	R.F.F-2
T.3.4.1	Implement “Retrieve the pattern list of the order of rating” functionality	
T.3.5	Implement “Add the pattern to Favorite list” functionality.	R.N.S-1, 2
T.3.5.1	Implement “Retrieve the Favorite list” functionality.	

5) Auto Mode

Task ID	Task Description	Related Req(s).
T.1	Implement “Run the auto mode” functionality.	R.F.F-1
T.1.1	Implement “Sensor activation” functionality.	

T.1.1.1	Inherit “See the event list”’ to detect the change of an activated event.	
T.1.1.2	Inherit “Check the threshold value of sensor modules” to check if the current sensor data exceeds a threshold.	
T.1.2	Implement “Calculate the most fitted light pattern” functionality.	
T.1.2.1	Inherit “Get the current sensor data” functionality.	
T.1.2.2	Inherit “Check the events and their states of a certain lighting module” to find the activated event.	
T.2	Implement “Give a feedback” functionality.	R.F.F-2
T.2.1	The system should save the feedback information to the database.	

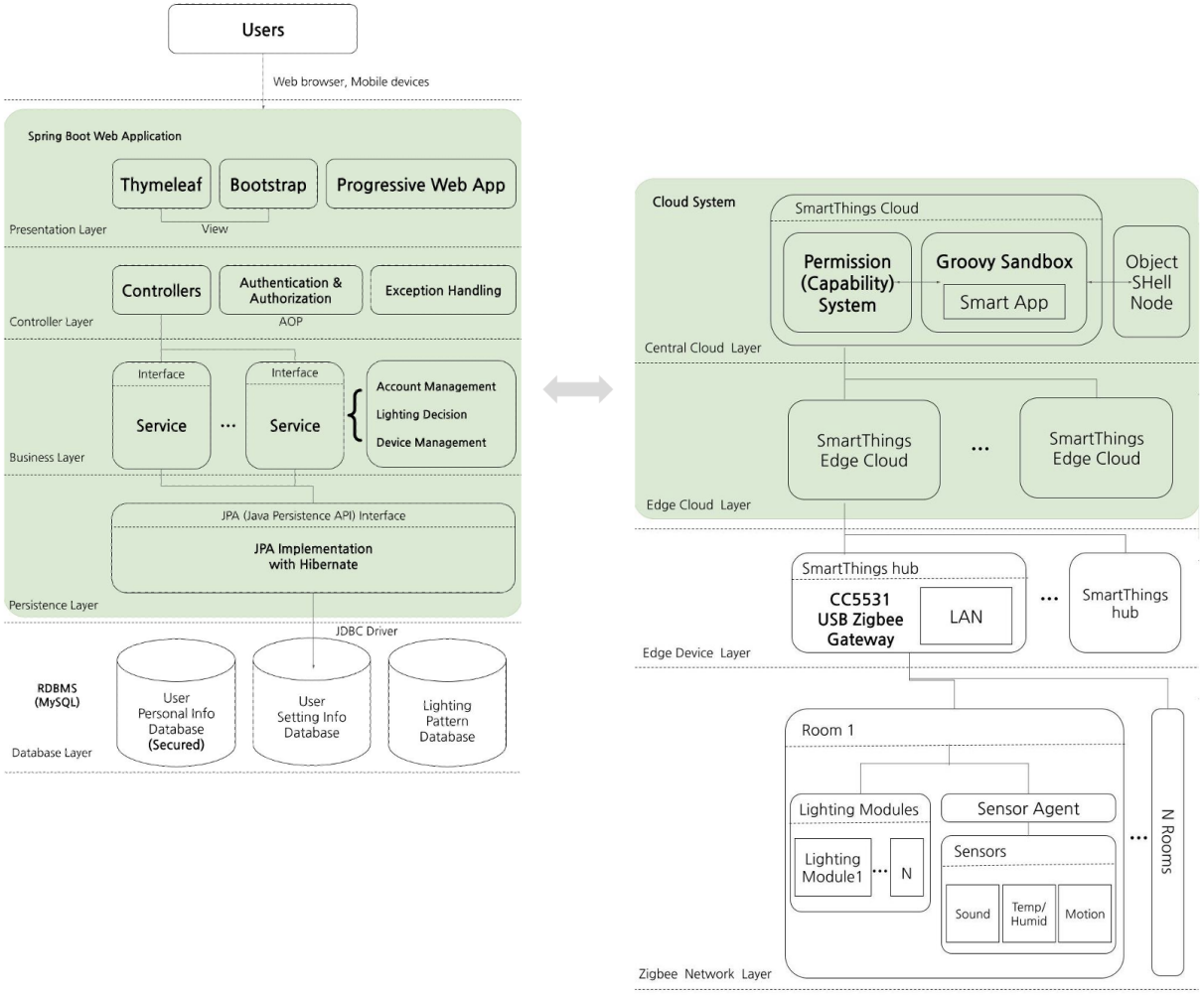
6) Extra Function

Task ID	Task Description	Related Req(s).
T.1	The software system is accessible via a mobile app, a mobile webpage, and a PC webpage.	R.N.E-1
T.2	The system should backup the database.	R.N.R-4
T.3	All the transaction speed should be no more than the designated time that figured in the non-FR document.	R.N.P-1,2,3,4
T.4	All the website size should be no more than the designated size that figured in the non-FR document.	R.N.P-5,6
T.5	All the response time should be no more than the designated time that figured in the non-FR document.	R.N.P-7,8

2. System Design

A. System Architecture

3-A System Architecture.pdf



Here is our overall system architecture shown above. Users mainly deal with the web application, which is designed according to the Spring Boot framework. The left green box represents a brief inner architecture of the web application.

There is one more important job of the web app, that it should interact with the SmartThings Cloud system through its REST API. The right green box shows the existing system's inner architecture briefly, and the others below show how our modules interact with the SmartThings hub and cloud.

B. Class Diagram

1) Descriptions and Classifications

a) Lighting Module Class description

Overall description : The aim of the lighting module system is to change the lighting pattern based on the given tasks from above. Following the given tasks, this system calculates the metadata object, 'LightingModuleDetails'. Based on that, the lighting module changes its lighting pattern.

This is implemented based on the 'master-slave' architectural model. Slave is generated per user. To change the lighting pattern, slave's 'LightingModuleDetails' should be copied to the master's 'LightingModuleDetails'.

There are two reasons for using that model. First, it enables synchronization among concurrent multiple user's transactions. Second, the user could have multiple choices. For example, the user could use the latest lighting pattern that others have used until recently, or find and use the color he or she used most recently. Please follow the table below for the details.

Class Name	Description
LightingModule	Master class of lighting module system. It receives the tasks from above, and distributes them for the appropriate slaves. After slaves do the tasks, it receives the 'LightingModuleDetails' from the slaves. Based on that, it changes the lighting pattern. Also it has synchronization method 'lock', to deal with the concurrent multiple user's transaction.
LightingModuleSlave	Slave class of lighting module system. It is generated per user, and it deals with all the tasks corresponding to that user. Based on the tasks, it calculates the metadata object, 'LightingModuleDetails', and sends it to the master.
LightingModuleDetails	Metadata object of lighting module system. It contains all the necessary information about the lighting pattern, such as mode, time duration, RGB color value, etc.

b) Hub Class description

Overall Description: Hub receives information from SmartThings Cloud and delivers it to sub-modules. Relation between Hub class and Module class is implemented based on the layer architectural model. Hub belongs to the edge device layer, and Module belongs to the zigbee network layer of system architecture. As you can see in system architecture, the edge device layer connects the SmartThings Cloud system layer and Zigbee network layer.

Class Name	Description
Hub	Hub object should be connected to the SmartThings Cloud system so that it can receive information about hub-connected users, modules.
Module	Class Module is the parent class of class LightingModule and SensorAgentModule. Class Module has attributes and methods that can be generally used in the LightingModule and SensorAgentModule such as modulecode, modulename, or shutdown().

c) Sensors Class description

Overall Description: Sensor Agent consists of three sub sensor agents: Temperature And Humidity Agent, Sound Pressure Sensor Agent, and Temperature And Humidity Sensor. The main role of the sub-sensor agent is to manage one or several sensors of each type, to gather the information from sensors, and to pass it to the sensor agent at

once. However, the main role of the sensor agent is to gather all information from three sensor agents, to verify whether those data trigger pre-defined conditions, and to load the data to the upper-level. This sensor class with sensor agents enables the system to effectively synchronize data of all different kinds and to minimize unnecessary steps to check whether the conditions of a specific mode are triggered or not.

Class Name	Description
SensorAgent	The SensorAgent has the virtual room number and datetime as attributes. It manages three Sub-SensorAgents so that it can activate and deactivate each of them. It receives collected data from Sub-SensorAgent and analyzes whether conditions are triggered or not. Moreover, it sends this data to the upper-side which is the “Module” class.
Sub-SensorAgent	The Sub-SensorAgent consists of Temperature And Humidity Agent, Sound Pressure Sensor Agent, and Temperature And Humidity Sensor, respectively. When it is activated, it manages one or more sensors of each type; however, when it is deactivated, it manages no sensor at all. It is responsible to register, to activate, and to deactivate each sensor. Moreover, it also takes a role in collecting data from sensors.
Sensor	The sensor consists of Temperature And Humidity Sensor, Sound Pressure Sensor, and Motion Sensor. The sensor collects each data depending on its own frequency when it detects the temperature/ sound/ motion of the surrounding environment exceeds its threshold value.

d) Web App Class description

Overall Description: The classes can be classified into five: Controller, Service, Repository, Entity, Dto. Controllers do a role of ‘Controller’ of MVC design pattern. There are four controller classes, each deals with their own domain. There are five service classes, which process main business logics. Four service classes are for each domain, and the AuthenticationService class is used for Sprint Boot framework’s authentication and authorization designed as AOP. The repository classes and entity classes are used for JPA. The entity classes corresponds to a DB table, and the repository is used to deal with the data. Finally, the DTO classes are used to data transfer, especially between the user and the web app. They are passed from the user to the web server with the appropriate input in a proper attribute.

Class Name	Description
HomeController	A controller class of main screen: user account management and it shows the connected hub list.
HubController	A controller class of hub control screen: hub management and it shows the connected module list.
ModuleController	A controller class of module controll screen: module management and it shows the module’s detail state.
CommunityController	A controller calss of community service: searching, reading, and

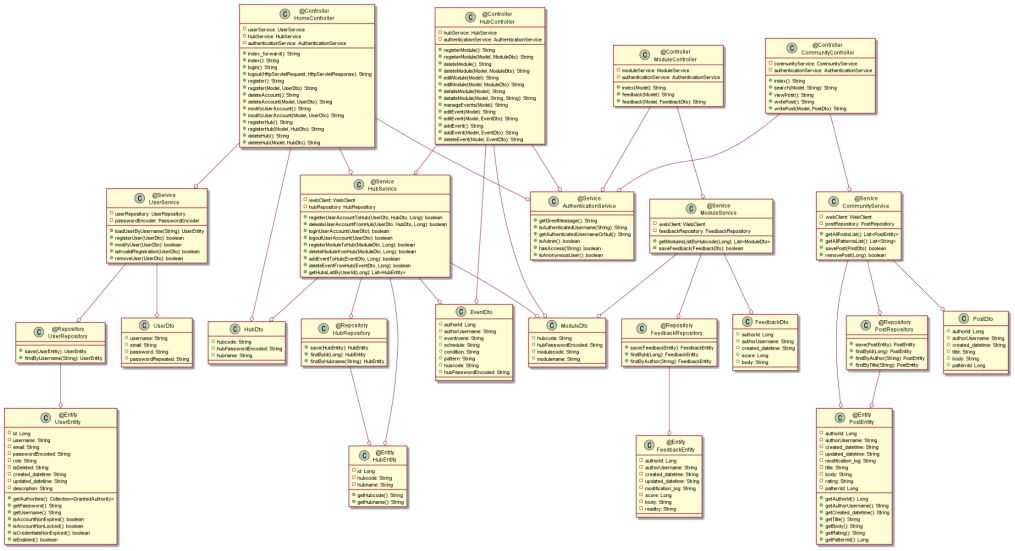
	writing a post and uploading/downloading patterns.
UserService	A service class that processes a logic for user account management: registration, modification, deletion, etc.
HubService	A service class that processes a logic for hub management: registration of a hub, event management, etc
AuthenticationService	A service class that processes a logic for user authentication and authorization.
ModuleService	A service class that processes a logic for module management: listing up module state, feedback, etc
CommunityService	A service class that processes a logic for community service: reading & writing a post, uploading & downloading a pattern, etc.
UserRepository	A repository class for UserEntity.
HubRepository	A repository class for HubEntity.
FeedbackRepository	A repository class for FeedbackEntity.
PostRepository	A repository class for PostEntity.
UserEntity	An entity class for user data on the database.
HubEntity	An entity class for hub data on the database.
FeedbackEntity	An entity class for feedback data on the database.
PostEntity	An entity class for post data on the database.
UserDto	A data transfer object class for user information.
HubDto	A data transfer object class for hub information.
EventDto	A data transfer object class for event information.
ModuleDto	A data transfer object class for module information.
FeedbackDto	A data transfer object class for feedback information.
PostDto	A data transfer object class for post information.

2) Class Diagram

We divided our system into two separate parts: **Web app** and **hub system**. As the web app and hub system communicate with each other only through the SmartThings Cloud system, there is no direct interaction between them. And especially for the Web app part, as we designed it according to an existing well made framework, the classes of the Spring Boot library are not on the diagram (e.g. WebClient). And there are only key annotations written over the class name if used.

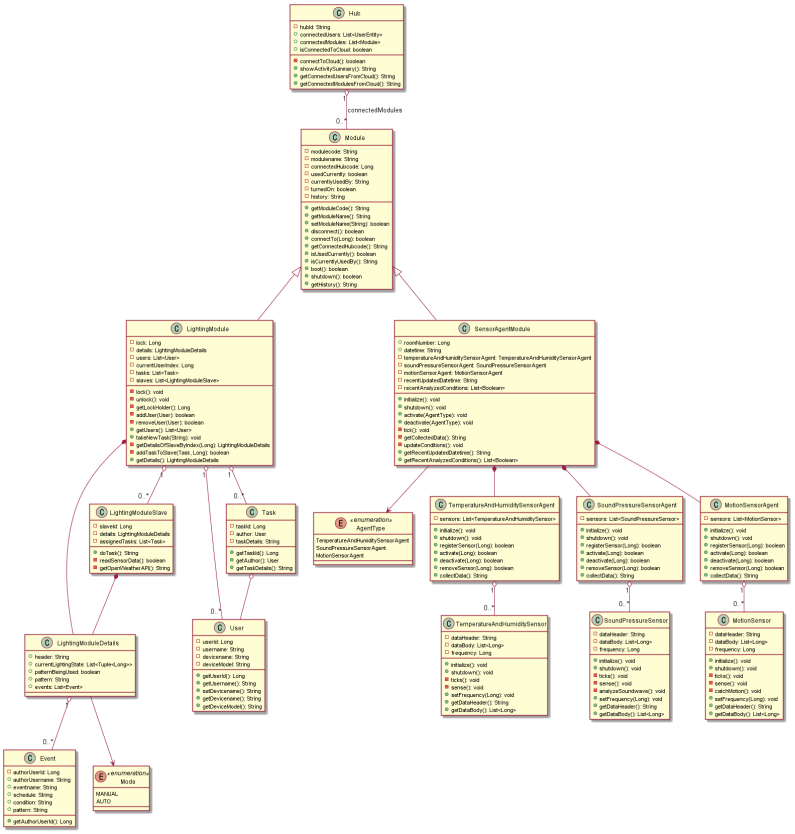
a) Web app class diagram

classdiagram_cs350_(1)webapp.png



b) Hub system class diagram

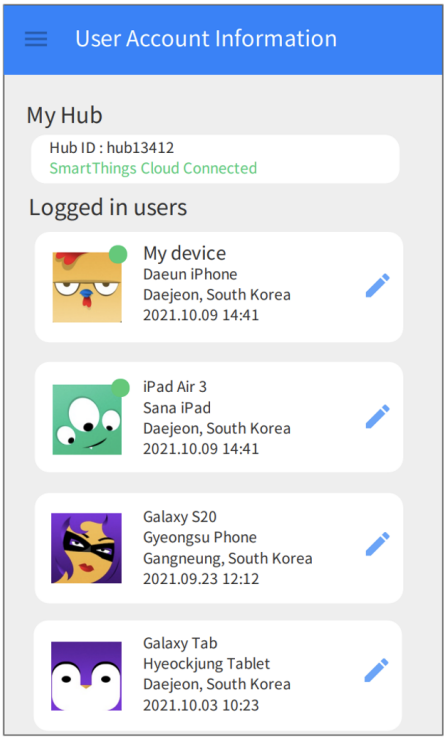
classdiagram_cs350_(2)hub.png

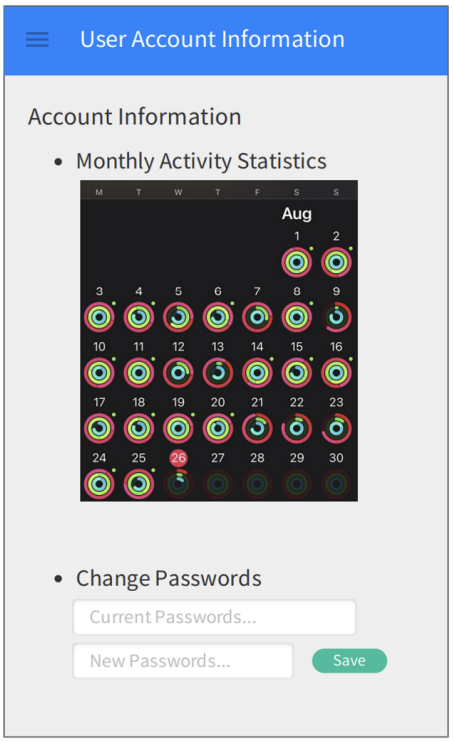


C. User Interface Design

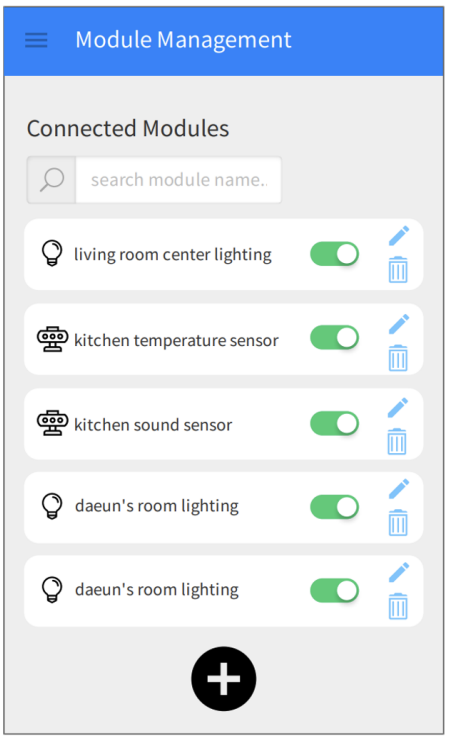
User Interface Final.pdf

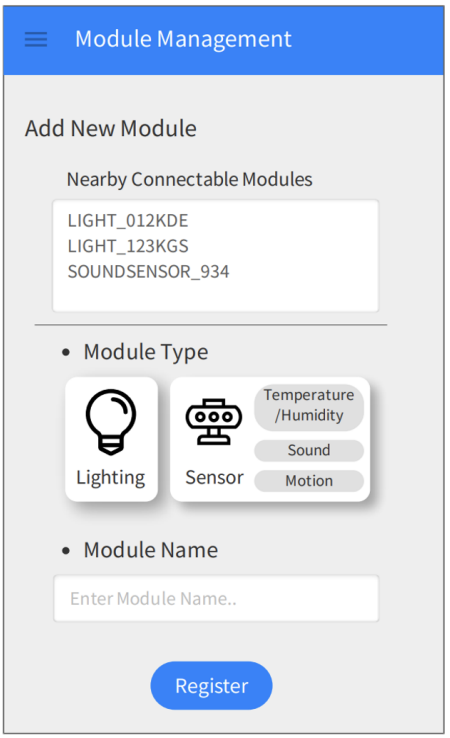
1) User Account



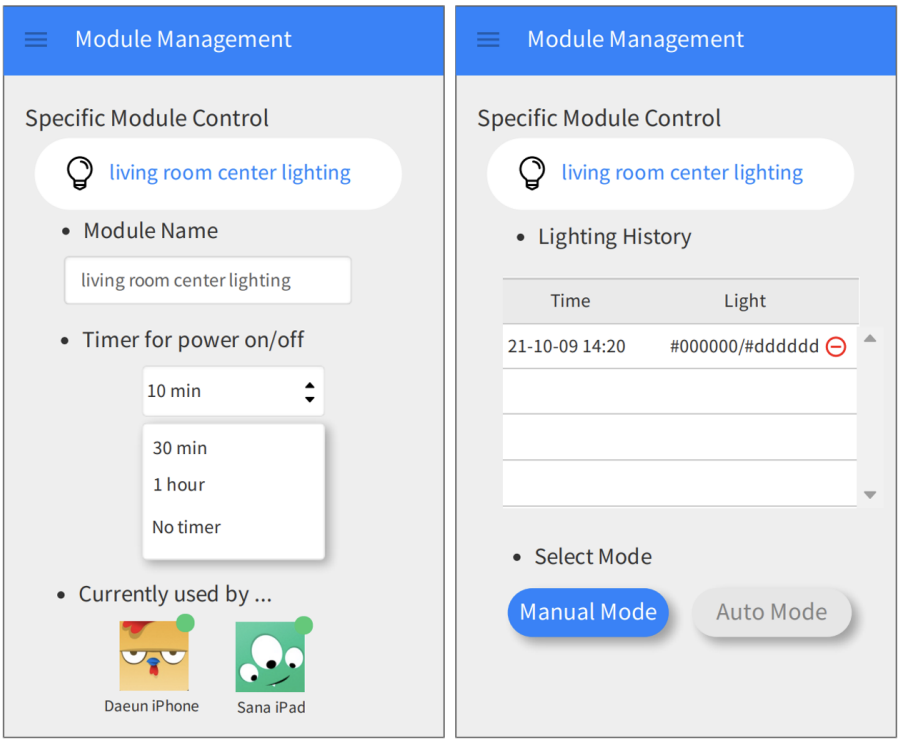


2) Module Management

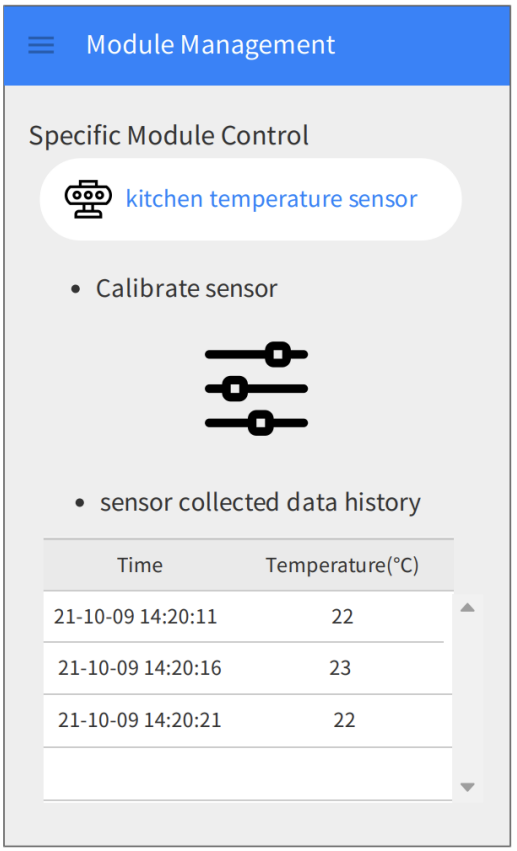




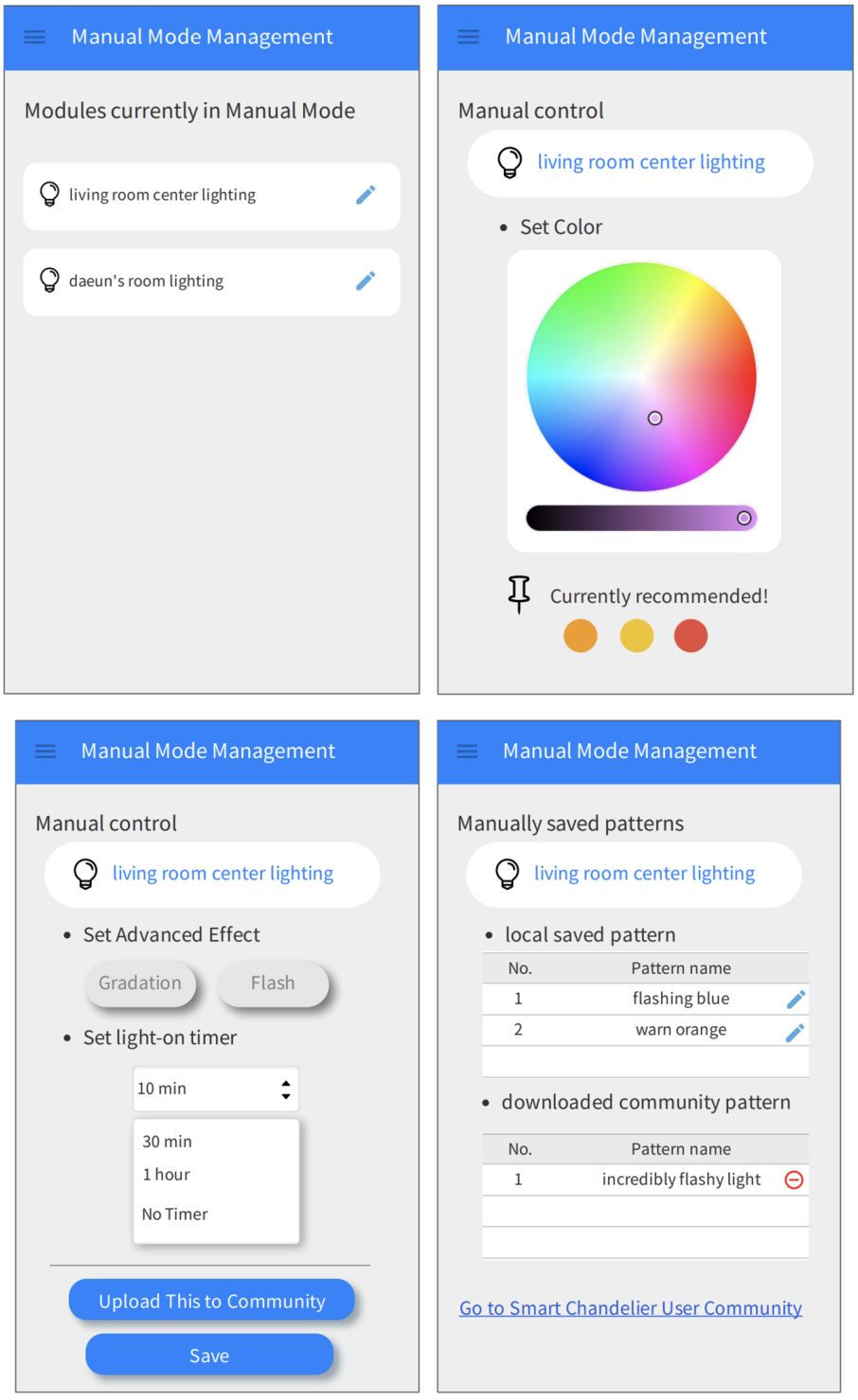
3) Lighting module control



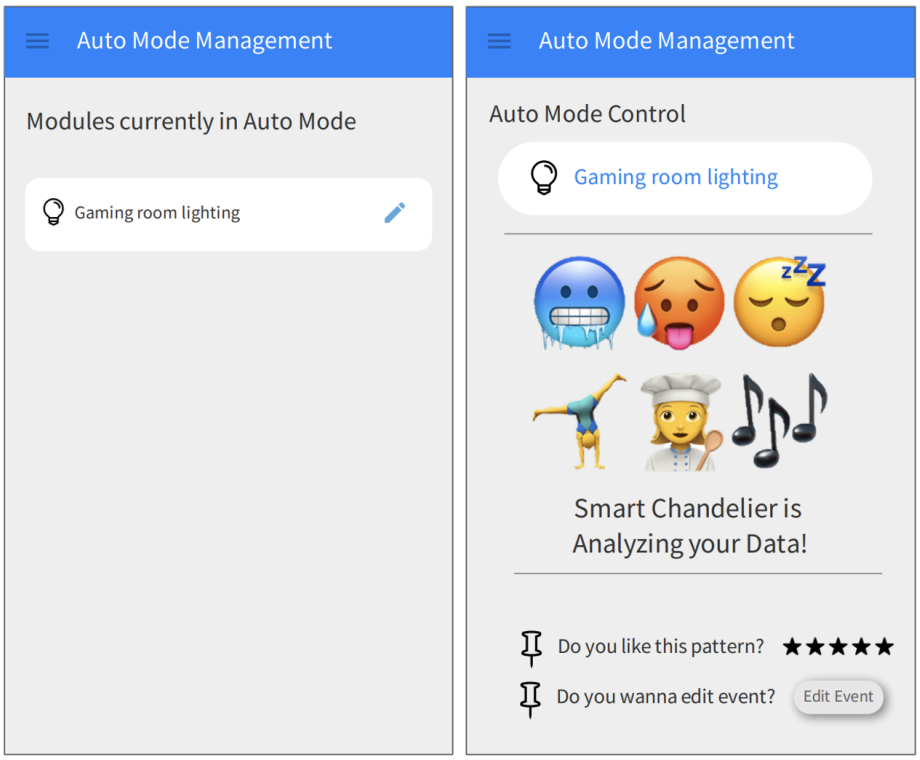
4) Sensor module control



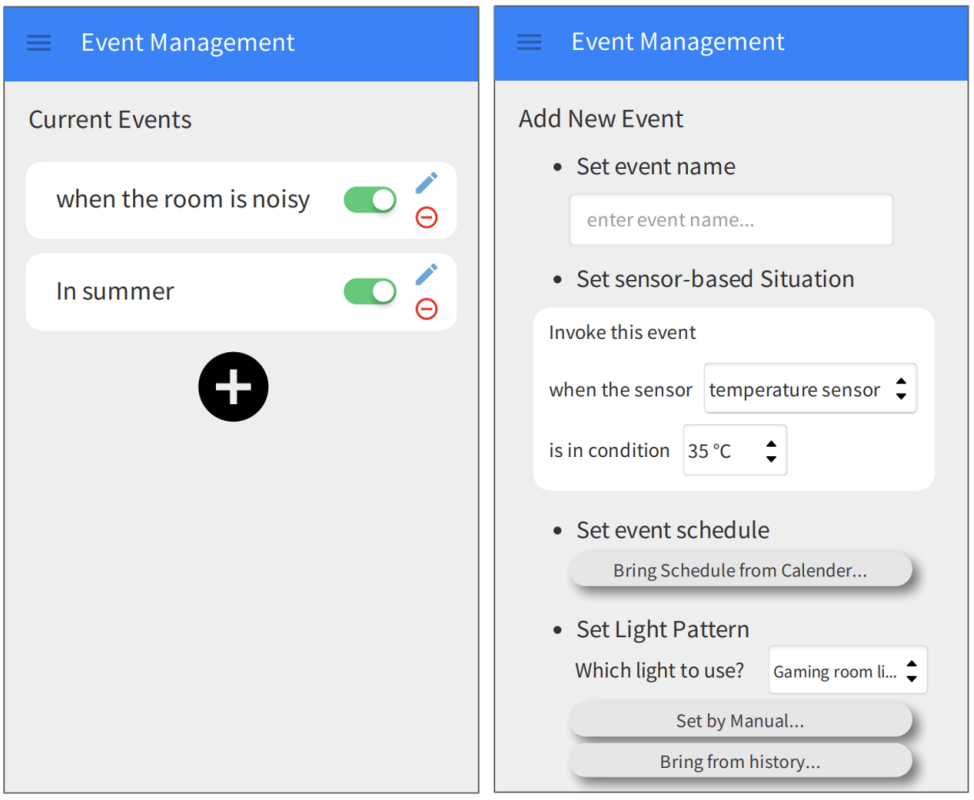
5) Manual mode



6) Auto mode

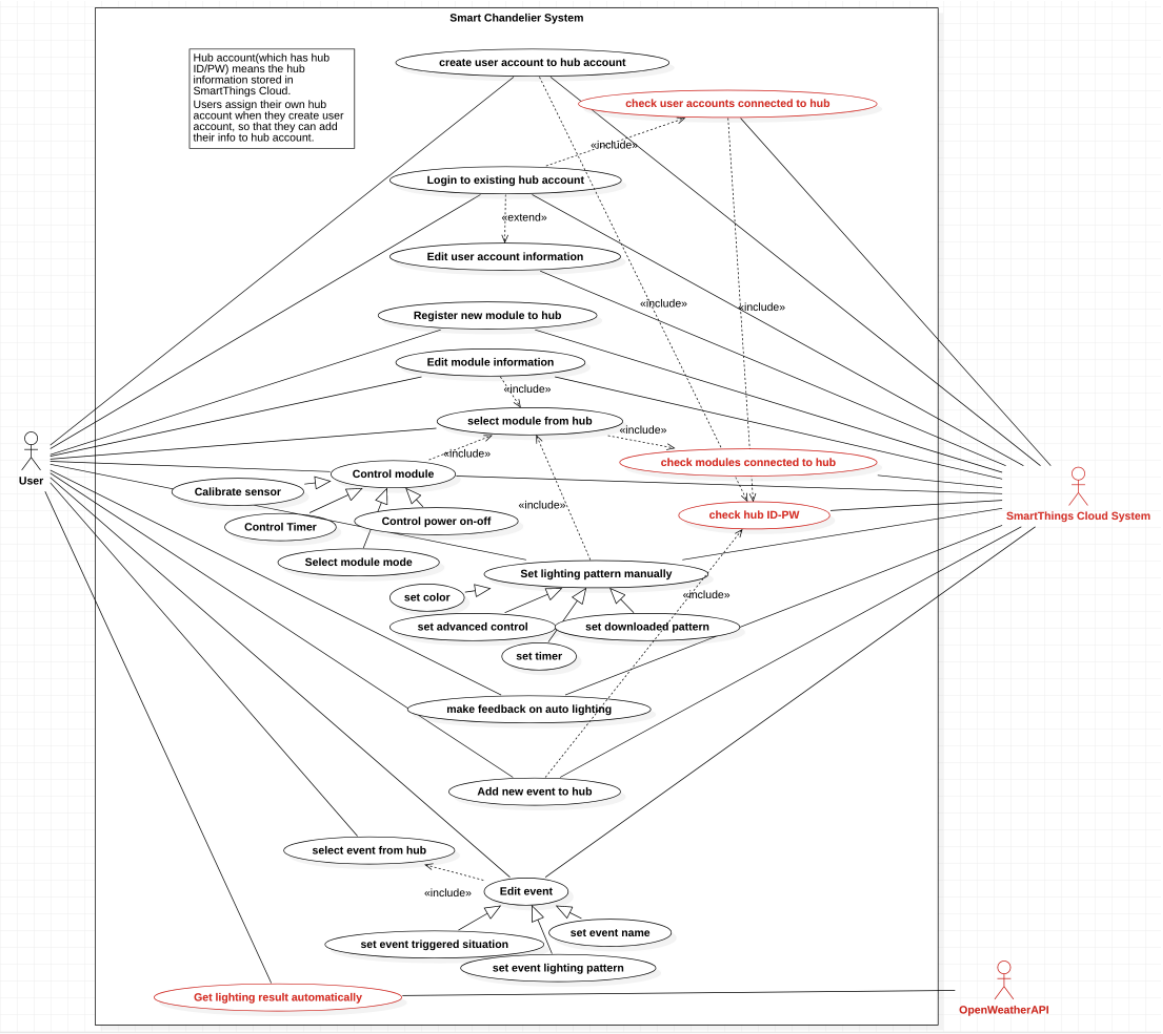


7) Add a event



D. Use Case Diagram & Description

Use case diagram final.png



1) Account Management

Use case name	Login to existing hub account	
Related Requirements	R.F.A-3	
Goal in Context	The user logs in to the existing hub account.	
Preconditions	Login is limited to hub-connected accounts, which hub is known to SmartThings Cloud System.	
Successful End Condition	The user account is logged in to the hub account.	
Failed End Condition	The application for login to the hub account is rejected.	
Primary Actors	User	
Secondary Actors	SmartThings Cloud System	
Trigger	The user asks the Smart Chandelier Web server to log in to the existing hub account.	
Main Flow	Step	Action

	1	The user asks the Smart Chandelier Web server to log in to the existing hub account. (Login can be done in 3 ways. FaceID, touchID, PIN.)
	2	SmartThings Cloud System checks if the hub account is stored in Smart Chandelier DB.
	3	SmartThings Cloud System verifies if the user account is connected to the hub account.
	4	The user account is logged in to the hub account.
Extensions	Step	Action
	2.1	SmartThings Cloud System does not find the hub account in Smart Chandelier DB.
	2.2	The application for login to the hub account is rejected.
	3.1	SmartThings Cloud System does not verify the user account is connected to the hub.
	3.2	The application for login to the hub account is rejected.
	4.1	After successful login, users can edit user account information.

2) Module Information Management

Use case name	Register new module to hub	
Related Requirements	R.F.B-1	
Goal in Context	The user registers a new module to hub.	
Preconditions	The hub and the module which wants to connect to each other should be connected by the Zigbee network.	
Successful End Condition	The module is registered to the hub.	
Failed End Condition	The application for registering a new module to hub is rejected.	
Primary Actors	User	
Secondary Actors	SmartThings Cloud System	
Trigger	The user asks the Smart Chandelier Web Server to register a new module to the hub.	
Main Flow	Step	Action
	1	The user asks the Smart Chandelier Web Server to register a new module to the hub. - The user enters the module information such as my own module name.
	2	SmartThings Cloud System asks the hub to connect the module(by zigbee network).

	3	The module is registered to the hub.
Extensions	Step	Action
	2.1	Connection between hub and module failed.
	2.2	The application for registering a new module to hub is rejected.

3) Module Control

Use case name	Control module	
Related Requirements	R.F.B-3	
Goal in Context	The user controls the module.	
Preconditions	The selected module should be connected to the hub so that SmartThings Cloud System is aware about it.	
Successful End Condition	The module control is applied to the module.	
Failed End Condition	The application for the controlling module is rejected.	
Primary Actors	User	
Secondary Actors	SmartThings Cloud System	
Trigger	The user asks the Smart Chandelier Web Server to control the selected module.	
Main Flow	Step	Action
	1	The user asks the Smart Chandelier Web Server to control the selected module.
	2	SmartThings Cloud System checks if the selected module is connected to the hub.
	3	According to the control type, SmartThings Cloud system sends requests to the hub to apply the control to the module. - control types are ‘power on-off’, ‘set timer’, ‘calibrate sensor’, ‘select module mode’...
	4	The module control is applied to the module.
Extensions	Step	Action
	2.1	Smart Chandelier System fails to find the module in hub-connected modules.
	2.2	The application for the controlling module is rejected.

4) Manual Mode Management

Use case name	Set lighting pattern manually	
Related Requirements	R.F.E-1	
Goal in Context	The user manually sets the lighting pattern of the selected lighting module.	
Preconditions	Selected module is lighting module. The selected module should be connected to the hub so that SmartThings Cloud System is aware about it.	
Successful End Condition	Manual lighting pattern is applied to the selected lighting module.	
Failed End Condition	The application for setting lighting pattern manually is rejected.	
Primary Actors	User	
Secondary Actors	SmartThings Cloud System	
Trigger	The user asks the Smart Chandelier Web Server to manually set the lighting pattern of the selected lighting module.	
Main Flow	Step	Action
	1	The user asks the Smart Chandelier Web Server to manually set the lighting pattern of the selected lighting module.
	2	SmartThings Cloud System checks if the selected module is connected to the hub.
	3	According to the entered lighting pattern, SmartThings Cloud System sends requests to the hub to apply the pattern to the lighting module. - There are 4 kinds of lighting patterns. Colors, timers, advanced control(flash, gradation), downloaded pattern.
	4	Manual lighting pattern is applied to the selected lighting module.
Extensions	Step	Action
	2.1	Smart Chandelier System fails to find the module in hub-connected modules.
	2.2	The application for setting lighting pattern manually is rejected.

5) Auto Mode Management

Use case name	Get lighting result automatically
Related Requirements	R.F.F-1
Goal in Context	User gets an automatically calculated lighting pattern.

Preconditions	The lighting module is in auto mode. The lighting module should be connected to the hub.	
Successful End Condition	User automatically gets the lighting result of the auto mode lighting module.	
Failed End Condition	-	
Primary Actors	User	
Secondary Actors	OpenWeatherAPI	
Trigger	The user sets the lighting module mode to auto mode through Smart Chandelier Web Server.	
Main Flow	Step	Action
	1	The user sets the lighting module mode to auto mode through Smart Chandelier Web server.
	2	The lighting module gets sensor data from the sensor module, and gets activated events information from the hub continuously.
	3	OpenWeatherAPI gives external data to lighting module continuously.
	4	Lighting module calculates lighting patterns appropriate to gathered data.
	5	User automatically gets the lighting result of the auto mode lighting module.
Extensions	Step	Action
	4.1	If change is detected in gathered data, the lighting pattern changes and applied to lighting.

6) Event Management

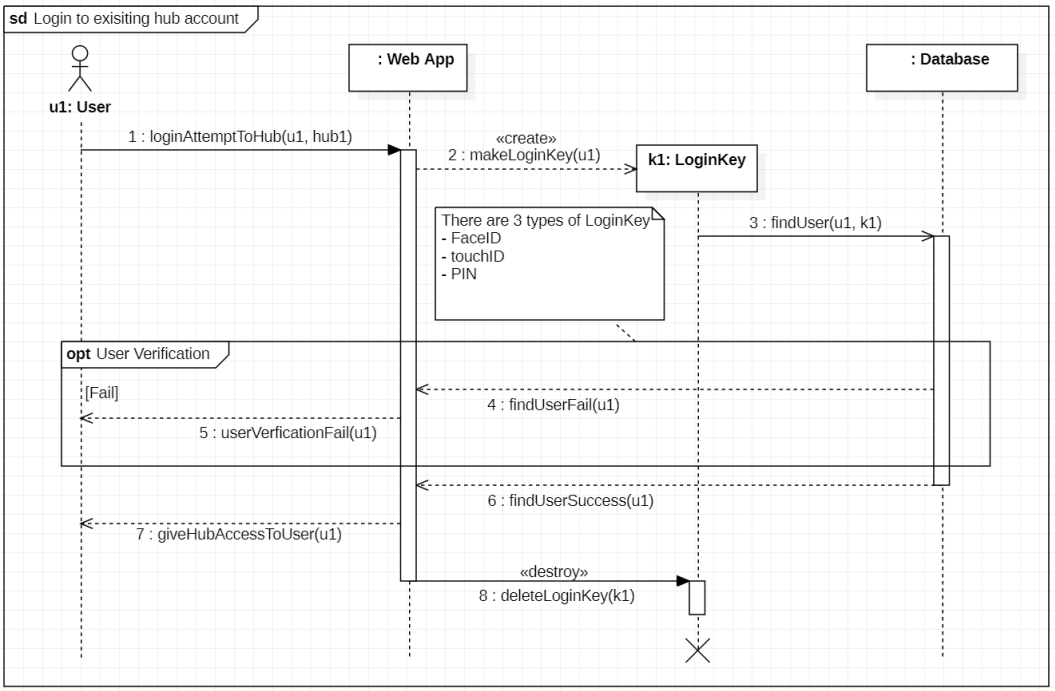
Use case name	Add new event to hub
Related Requirements	R.F.C-1
Goal in Context	The user adds a new event to hub.
Preconditions	The hub account is known as SmartThings Cloud System.
Successful End Condition	New event is added to the hub.
Failed End Condition	The application for adding a new event to hub is rejected.
Primary Actors	User
Secondary Actors	SmartThings Cloud System
Trigger	The user asks the Smart Chandelier System to add a new event to hub.

Main Flow	Step	Action
	1	The user asks the Smart Chandelier System to add a new event to the hub. - The user enters the event information(event name, triggered situation, result lighting pattern)
	2	SmartThings Cloud System checks if the hub account is stored in Smart Chandelier DB.
	3	Event is stored in the Smart Chandelier system DB.
	4	New event is added to the hub.
Extensions	Step	Action
	2.1	Hub account is not stored in the Smart Chandelier DB.
	2.2	The application for adding a new event to hub is rejected.

E. Sequence Diagram

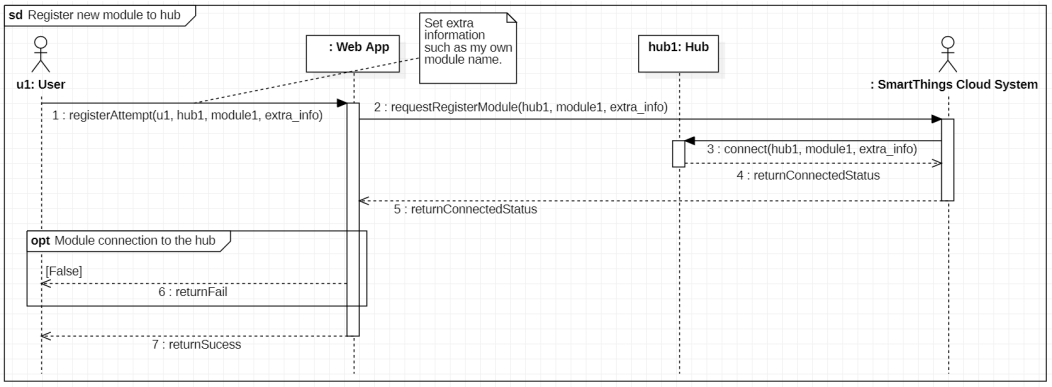
1) Login with the existing account

add_new_event_to_hub.PNG



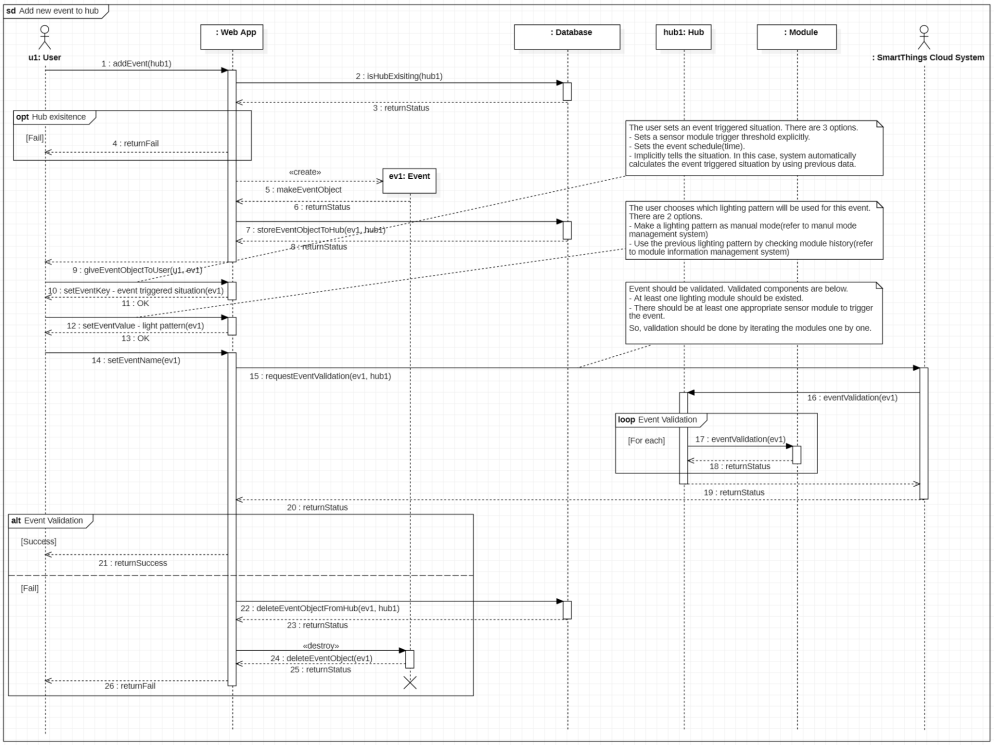
2) Register a new module

register_new_module_to_hub.PNG



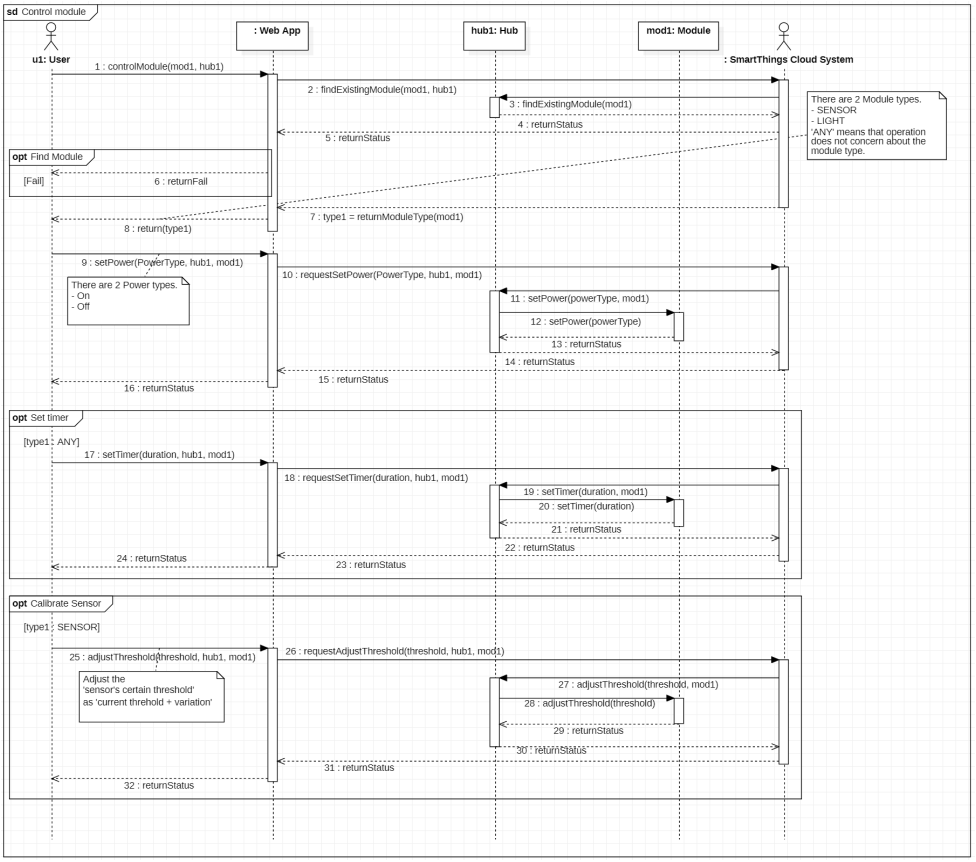
3) Add a new event

add_new_event_to_hub.PNG



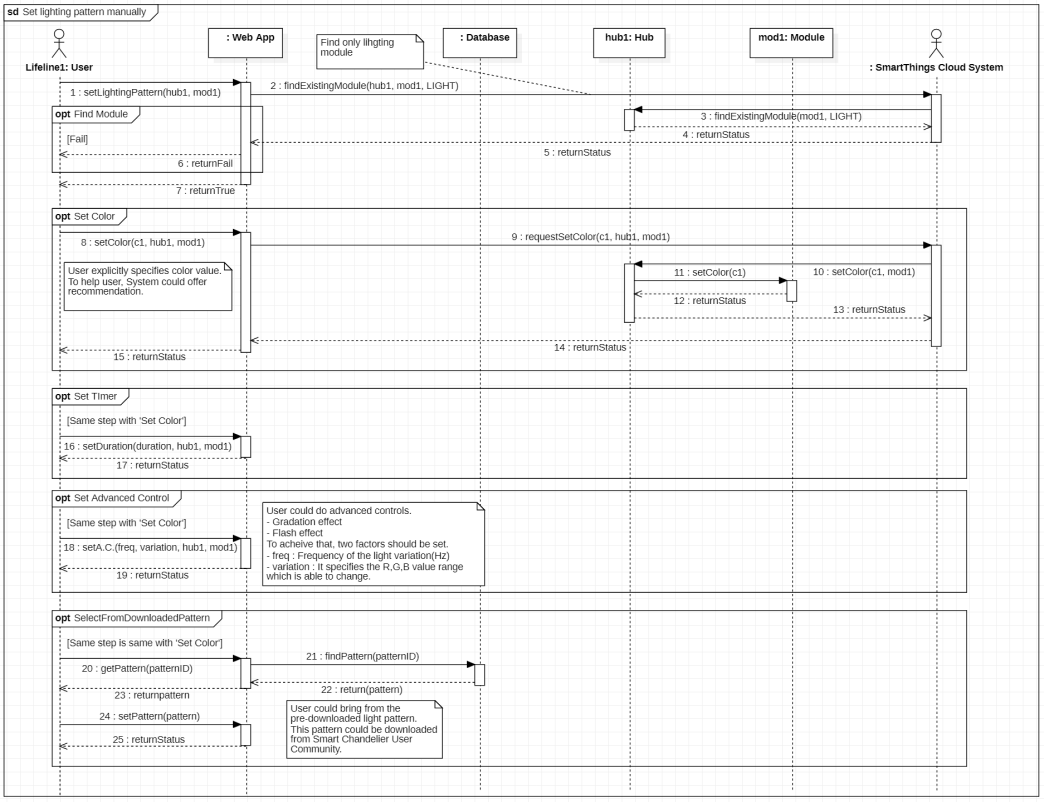
4) Control the module

control_module.PNG



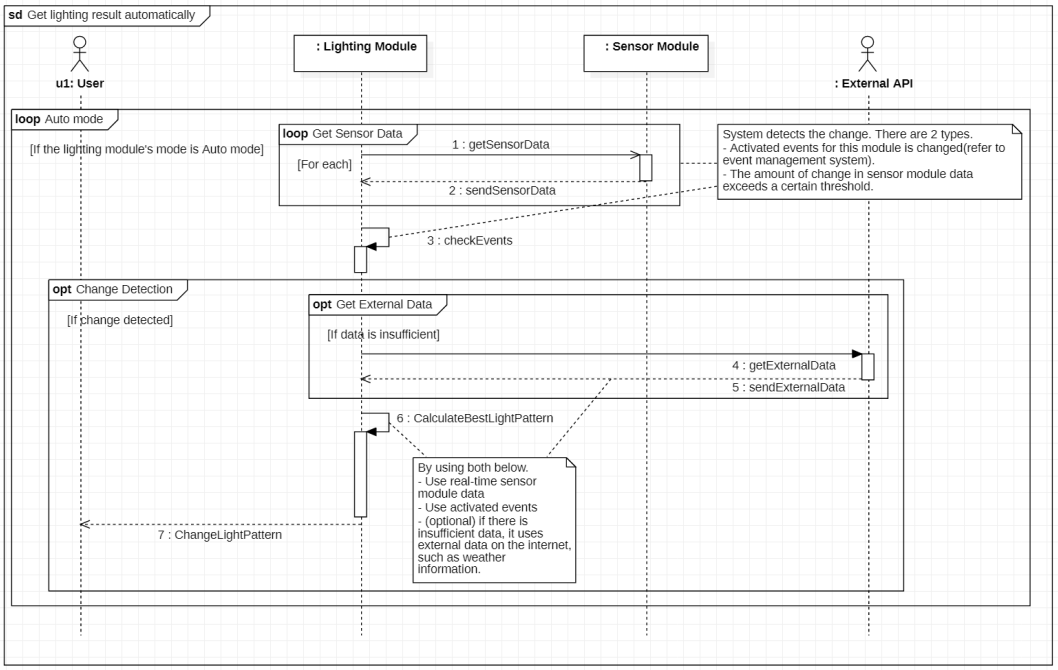
5) Manual mode

set_lighting_pattern_manually.PNG



6) Auto mode

get_lighting_result_automatically.PNG



3. Acknowledgements

Gyeongsu Kim	2-A System Architecture, 2-B Class Diagram (Lighting Module), 2-E Sequence Diagram
Daeun Kim	2-B Class Diagram (Hub), 2-C User Interface Design, 2-D Use Case Diagram & Description
Hyuckjoong Yoon	1-A Requirements (FR), 2-A System Architecture, 2-B Class Diagram (Web, Arrangement & Supplementation)
Sana Kang	1-A Requirements (Non-FR), 1-B Tasks, 2-A System Architecture, 2-B Class Diagram (Sensors), SDD Documentation Arrangement