



Sensor-Based Indoor Air Purification System

Software Requirement Specification

2022.05.01.

Introduction to Software Engineering 42

TEAM 11

| | |
|-------------|---------------|
| Team Leader | Cha Jeong Min |
| Team Member | Hangyu Kim |
| Team Member | Joonsun Back |
| Team Member | Seyeon Park |
| Team Member | Sungjoon Lee |
| Team Member | Jingyu Lee |
| Team Member | Maike Helbig |

CONTENTS

| | |
|--|---------------|
| 1. Introduction | - 6 - |
| 1.1. Purpose..... | - 6 - |
| 1.1.1. Purpose of Documents | - 6 - |
| 1.1.2. Purpose of Systems..... | - 6 - |
| 1.2. Scope..... | - 6 - |
| 1.3. Definitions, Acronyms, and Abbreviation..... | - 7 - |
| 1.4. References | - 8 - |
| 1.5. Overview | - 8 - |
| 2. Overall Description | - 9 - |
| 2.1. Product perspective..... | - 9 - |
| 2.1.1. System Interfaces | - 9 - |
| 2.1.2. User Interface | - 9 - |
| 2.1.3. Hardware Interface..... | - 10 - |
| 2.1.4. Software Interface..... | - 10 - |
| 2.1.5. Communications Interface | - 10 - |
| 2.1.6. Memory Constraints | - 11 - |
| 2.1.7. Operations | - 11 - |
| 2.1.7.1. Operations of the Admin | - 11 - |
| 2.1.7.2. Operations of the User | - 11 - |
| 2.2. Product functions | - 12 - |
| 2.2.1. Measurement and Transmission Module..... | - 12 - |
| 2.2.2. Data Analysis Module..... | - 12 - |
| 2.2.3. Numerical adjustment module..... | - 13 - |
| 2.2.4. Validity Module..... | - 13 - |
| 2.3. User Characteristics | - 14 - |
| 2.3.1. Admin Characteristics..... | - 14 - |
| 2.3.2. Standard User Characteristics..... | - 14 - |
| 2.4. Constraints..... | - 14 - |
| 2.5. Assumptions and Dependencies | - 15 - |
| 3. Specific Requirements | - 16 - |

| | |
|---|---------------|
| 3.1. External Interface Requirements | - 16 - |
| 3.1.1. User Interfaces..... | - 16 - |
| 3.1.2. Hardware Interfaces | - 20 - |
| 3.1.3. Software Interfaces | - 21 - |
| 3.1.4. Communication Interfaces | - 23 - |
| 3.2. Functional Requirements | - 25 - |
| 3.2.1. Use Case Example | - 25 - |
| 3.2.2. Use Case Diagram | - 29 - |
| 3.2.3. Data Dictionary | - 29 - |
| 3.2.4. Data Flow Diagram..... | - 35 - |
| 3.3. Nonfunctional Requirements | - 35 - |
| 3.3.1. Product Requirements..... | - 35 - |
| 3.3.1.1. Usability Requirements | - 35 - |
| 3.3.1.2. Dependability Requirements..... | - 36 - |
| 3.3.1.3. Efficiency Requirements..... | - 36 - |
| 3.3.1.4. Probability Requirements | - 37 - |
| 3.3.1.5. Security Requirements | - 37 - |
| 3.3.2. Organizational Requirements | - 38 - |
| 3.3.2.1. Development Requirements | - 38 - |
| 3.3.2.2. Environmental Requirements..... | - 38 - |
| 3.3.2.3. Operational Requirements | - 39 - |
| 3.3.3. External Requirements..... | - 39 - |
| 3.3.3.1. Legislative Requirements | - 39 - |
| 3.3.3.1.1. Privacy Requirements..... | - 39 - |
| 3.3.3.1.2. Safety Requirements | - 39 - |
| 3.4. Performance Requirements..... | - 39 - |
| 3.5. Design Constraints | - 39 - |
| 3.6. Standards Compliance..... | - 40 - |
| 3.7. Organizing the Specific Requirement | - 40 - |
| 3.7.1. Context Model | - 40 - |
| 3.7.2. Interaction Model..... | - 41 - |
| 3.7.3. Behavior Model | - 42 - |
| 3.7.3.1. Data Flow Diagram..... | - 42 - |

| | |
|--|--------|
| 3.7.3.2. Sequence Diagram..... | - 42 - |
| 3.8. System Architecture..... | - 43 - |
| 3.9. System Evolution..... | - 43 - |
| 3.9.1. Limitation and Assumption..... | - 44 - |
| 3.9.2. Evolutions of Hardware and Change of User Requirements..... | - 44 - |
| 4. Supporting Information | - 44 - |
| 4.1. Software Requirement Specification | - 44 - |
| 4.2. Document History | - 45 - |

LIST OF FIGURES

| | |
|---------------------------------------|--------|
| [Figure 1] Login page | - 16 - |
| [Figure 2] Admin page..... | - 17 - |
| [Figure 3] Main page | - 18 - |
| [Figure 4] Control page | - 19 - |
| [Figure 5] Use case diagram | - 29 - |
| [Figure 6] Data flow diagram | - 35 - |
| [Figure 7] Context model | - 40 - |
| [Figure 8] Interaction model..... | - 41 - |
| [Figure 9] Sequence diagram..... | - 42 - |
| [Figure 10] System architecture | - 43 - |

LIST OF TABLES

| | |
|---|--------|
| [Table 1] Table of acronyms and abbreviations | - 7 - |
| [Table 2] Table of terms and definitions | - 7 - |
| [Table 3] User interface – Login page | - 16 - |
| [Table 4] User interface – Admin page..... | - 17 - |
| [Table 5] User interface – Main page | - 18 - |
| [Table 6] User interface – Controller page | - 19 - |

| | |
|--|--------|
| [Table 7] Hardware interface - Smartphone | - 20 - |
| [Table 8] Hardware interface - Computer..... | - 20 - |
| [Table 9] Hardware interface – Air controlling devices | - 20 - |
| [Table 10] Hardware interface – Air monitoring sensors..... | - 20 - |
| [Table 11] Hardware interface – Arduino Board..... | - 21 - |
| [Table 12] Software interface – Firebase | - 21 - |
| [Table 13] Software interface - Arduino IDE | - 22 - |
| [Table 14] Communication between user and air controlling devices | - 23 - |
| [Table 15] Communication between User and Backend | - 23 - |
| [Table 16] Communication between backend and Arduino Board..... | - 24 - |
| [Table 17] Communication between Arduino Board and Air control / monitoring devices(sensors)..... | - 24 - |
| [Table 18] Log in and authorization Process..... | - 25 - |
| [Table 19] Main Page..... | - 26 - |
| [Table 20] Admin Page | - 27 - |
| [Table 21] Controller Page | - 27 - |
| [Table 22] Emergency detection | - 28 - |
| [Table 23] Table of Data Dictionary - User..... | - 29 - |
| [Table 24] Table of Data Dictionary - System Control | - 30 - |
| [Table 25] Table of Data Dictionary - Purifier | - 30 - |
| [Table 26] Table of Data Dictionary - Air conditioner..... | - 31 - |
| [Table 27] Table of Data Dictionary - Ventilation | - 31 - |
| [Table 28] Table of Data Dictionary - Humidifier..... | - 32 - |
| [Table 29] Table of Data Dictionary - Emergency detection..... | - 33 - |
| [Table 30] Table of Data Dictionary - Authorization | - 33 - |
| [Table 31] Arduino board memory specification | - 37 - |
| [Table 32] Document History | - 45 - |

1. Introduction

1.1. Purpose

1.1.1. Purpose of Documents

This document contains the requirements and specifications for the development of the “Sensor-Based Indoor Air Purification System” project. This document provides information necessary for development by specifying the services and operation methods that the system must provide, as well as various restrictions and standards, and can be used as a guide for users to understand and use the system appropriately. In addition, the ultimate purpose of this document is to provide a clear understanding of the system to all stakeholders by describing the interests of different types of readers in the system.

1.1.2. Purpose of Systems

Year by year air quality keeps deteriorating to the extent that it is worsening people’s health and life quality. Due to this, people try to keep good air quality by controlling devices such as air conditioners, humidifiers, ventilators, or air purifiers. However, the problem is that people do not know how to control those in order to maintain good air quality, and also they can’t control each device at the same time.

Therefore our team came up with the idea of a system that can solve this problem, by building an application that monitors current air quality status, and controls air quality controlling devices all at once, both manually or automatically, by using IOT and embedded systems. As a result, users of this application will be able to maintain desirable air quality wherever and whenever.

1.2. Scope

This document is concerned with describing the requirements for our team’s air purification system. This includes the requirements for three softwares essential to this system: The smartphone application responsible for the communication between user and system, the software running on the backend of the system coordinating the air quality according to the User’s preferences and the software handling the communication of the required hardware components such as devices and sensors. Furthermore, it will be described how these separate

entities will be communicating and what sort of hardware and other software will be required to make the system run according to its requirements.

1.3. Definitions, Acronyms, and Abbreviation

The following table contains the acronyms and abbreviations used in this requirements specification document.

[Table 1] Table of acronyms and abbreviations

| Acronyms & Abbreviations | Explanation |
|--------------------------|------------------------------------|
| IDE | Integrated Development Environment |
| OS | Operating System |
| DB | DataBase |
| IDE | Integrated Development Environment |

The following table defines certain technical terms used in this document.

[Table 2] Table of terms and definitions

| Terms | Definition |
|-----------------|--|
| Arduino | An open-source hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices. |
| Fine dust | particulate matter that can be found in the air that is incredibly small |
| Ultra fine dust | particulate matter of nanoscale size (less than 0.1 μm or 100 nm in diameter).(Also known as ultra fine particle) |
| Firebase | Google-backed application development software that enables developers to develop iOS, Android and Web apps. |
| Flash memory | Non-volatile computer memory that can be electrically erased and reprogrammed |
| SRAM | SRAM is a memory device that stores its contents only while power is supplied, and |

| Terms | Definition |
|---------------------|--|
| | recorded data is not erased as long as power is supplied. The memory capacity is small, but the response speed is very fast. |
| EEPROM | EEPROM is an electrically erasable PROM. There is no need for a dedicated eraser to erase data, and it can be written and erased using a single ROM writer. However, EEPROM is very slow compared to flash memory and has a limit on the number of repeated writes. |
| Non-volatile memory | Computer memory that retains stored information even when power is not supplied. |
| Boot loader | A program that is executed before the operating system is started, completes all related tasks necessary for the kernel to boot properly, and has the purpose of finally starting the operating system. |

1.4. References

- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEExplore Digital Library
<http://ieeexplore.ieee.org/Xplore/guesthome.jsp>
- Team 11. “Software Requirement Specification”. SKKU, Last Modified: May. 1, 2022.
https://github.com/cjmin1233/2022spring_42class_team11/tree/main/doc/Team11_RS.docx

1.5. Overview

Humans have enjoyed tremendous convenience since the 21st century with the development of excellent science and technology and medical systems. Like the word ubiquitous, mankind wants to have access to the Internet everywhere and control everything everywhere. Our team developed it with a focus on controlling people's homes everywhere in line with the concept of smart homes. This program allows for overall control of the air condition in the house. Users can register up to four types of devices. Air conditioner, humidifier, ventilation system, air purifier. The user may control the devices in the auto mode and may control on-off by himself.

A brief summary of the structure of this document is as follows. Chapter 2 describes the user interface, system interface, hardware and software interface, and memory constraints and operating principles and more for the program. Chapter 3 provides more detailed information on specification. It provides detailed information on each interface, as well as functional requirements such as diagrams and data dictionaries for use cases. It also includes nonfunctional requirements for developers and operators.

2. Overall Description

2.1. Product perspective

This system is a self-contained system designed to synchronize indoor air quality control across multiple devices and provide a centralized interface for users to both check the state of the air quality as well as do manual adjustments in accordance to their personal preferences.

2.1.1. System Interfaces

The overall system is composed of smaller systems that work together to accomplish the requirements. First there are sensors that measure the air quality and report those values to the backend. Similarly, there are devices such as heaters or humidifiers that receive commands from the backend to adjust the specific air quality parameters. Both the sensors and the devices are connected to an Arduino board that has the logic of how to deal with both incoming data from the backend and outbound data from the sensors. The backend itself runs analytical software to use the data obtained by the sensors and transfer them into commands for the devices based on the desired air quality values. The backend also receives the commands from the Users. Lastly, there is a database that stores values that the backend obtained from the sensors to be able to make decisions based on trends in the change of air quality parameters.

2.1.2. User Interface

The User interacts with the system over the smartphone application. The User interface is an

application running on the User's smartphone. It will be accessible after authenticating the user's identity. The User interacts with it over touch. The app will consist of multiple pages that serve their respective purposes and that the user can navigate between. Pages with technical detail will be clearly separated from pages with core functionalities to keep Users with less technical expertise from being overwhelmed. All input into the app regarding air quality parameters will be regulated by sliders rather than direct input to avoid Users input unrealistic and potentially harmful values.

2.1.3. Hardware Interface

The software for the backend will be running on a computer equipped with an x86 processor. The Arduino board used for this system needs to be wifi-capable to transmit data to the backend over the internet.

2.1.4. Software Interface

The analytical backend software will need to run on top of Windows 10 or a more recent version of it. The application running on the User's smartphone requires Android 5.0 and up or IOS 6.1.6 and up.

2.1.5. Communications Interface

The system has multiple components that all have to communicate with one another. The User and the backend communicate with each other over the Internet. Therefore, both User and backend will need a working internet connection to function properly. The backend and the Arduino board managing the sensors and devices communicate over the Internet, too. In order to avoid hacking attacks, the data traffic will be encrypted using symmetric key cryptography where the key for communication will be preinstalled on both the backend and the Arduino board. The Arduino board itself will be connected to its sensors and devices over cable. It can accept up to eight input/output channels. This implies that if devices and sensors are too far apart or exceed the number of eight, additional Arduino boards will be needed. Database and backend will not need an additional communication interface, as they will be located on the

same machine and communicate internally.

2.1.6. Memory Constraints

The database should have enough memory to at least store a day's worth of data. How much memory that exactly is cannot be determined this early on in development and it can vary with the number of sensors and how often measurements from the sensors are being sent to the database. For the sake of completion, an estimate will be given, assuming the measurement of 6 different values in the form of a 32-bit value every minute. In that scenario, roughly 7GB of memory will be required for the database.

2.1.7. Operations

2.1.7.1. Operations of the Admin

The Admin will be able to delegate privileges across other users to define what actions they are allowed to perform. The Admin will be able to add or remove devices and sensors from the system. The Admin will be able to upload new code to the Arduino to micromanage sensor and device behavior. The Admin will also be able to execute all of the User's operations. The Admin will be able to define standard values for the air quality parameters within certain limitations. The Admin will be able to add emergency scenarios at which the system is supposed to send an alarm message to all Users.

2.1.7.2. Operations of the User

The User will be able to perform actions in accordance with the privilege that has been given to them by the Admin. With full privilege, Users will be able to adjust the temperature, humidity, ventilation and filtration by adjusting individual device activity over the app. Users will also be able to simply display the current state of the air quality. The User will be able to turn devices on and off.

2.2. Product functions

2.2.1. Measurement and Transmission Module

For air purification, the function of measuring the current state must be accompanied first. Devices involved in air quality are designed based on all available devices, although, when actually applied, different environments may vary from place to place. In the future, it is not considered separately because it can be simply adjusted on/off in the process of application.

Each device operates independently. There are a total of four sensors that will be received by measuring data from the device: carbon dioxide sensor, fine dust sensor, humidity sensor, and temperature sensor. Each sensor will be connected to and received by the device, or measurement will be performed by directly installing the sensor.

Specifically, first, the temperature and humidity sensor may not be limited to one according to the size of a space, and since it is essential to enter an air conditioner or an air purifier, measurement data values may be obtained without additional sensors being installed in a space. Since fine dust sensors are important not only inside the space but also outside data, they not only directly measure fine dust inside the space through fine dust sensors, but also collect and analyze fine dust data in the area where the space provided by top institutions belongs. Finally, like fine dust sensors, carbon dioxide sensors must measure and transmit internal and external data to conduct analysis for accurate air purification. However, since internal figures are more important than fine dust, data are always collected with the corresponding sensors in places where there is a risk of carbon dioxide detection, such as the kitchen.

The data measured by each sensor is physically linked to the Arduino board and collects all data based on the C++ program.

2.2.2. Data Analysis Module

The user receives the desired input first through the device. The input can be directly adjusted by the user, which is transmitted to the backend. At the same time, the Arduino board sends sensor data to the backend. It also receives information about data from the database. Based on the three pieces of information, find the optimal numerical value.

First, the current state of air is analyzed based on the data received from the sensor and the database therebetween. Here, it checks whether any abnormal value does not exist, and stores the value in the database. Next, a command value required for each device is derived by analyzing the current air quality state of the user's input value. For example, if it is too humid due to the weather, run the dehumidifier in an appropriate amount to control the humidity. If the user inputs to lower the humidity, the dehumidifier is also operated. The appropriate humidity analyzed through continuous feedback or the humidity is lowered to a value set by the user.

If the user's input is automatic, the device is adjusted by receiving data for an optimal air quality set based on previous data taken from the database. There are two ranges of data for this: general standards and user trends, and customized standards in consideration of the composition. At first, only general criteria will exist, but as data accumulates, a narrower range of criteria is clearly shown to suit the user.

2.2.3. Numerical adjustment module

The Arduino board receives the value determined by analyzing it at the back end. The Arduino board transmits the set sensor value to the device again in consideration of the user's input. When the device is finally operated, the above process is repeated at specific time intervals to repeat the trend of air quality.

2.2.4. Validity Module

There is a possibility of errors in each communication process, such as the process of reading data and collecting it into a C++ program, the process of inputting data directly by the user, storing and loading sensor data into a database at the backend, and lowering the operation from the Arduino board to the device. Or, there is a possibility that the user's input is abnormal. To prevent this, set the range of normal data and the range of allowed data, and check it in the process of turning over the data to prevent errors. In the case of indoor air, it is necessary to check the effectiveness because it is a part that can have a great impact on the human body in the long term.

2.3. User Characteristics

2.3.1. Admin Characteristics

The Admin will be expected to have at least some degree of technical knowledge. The system can technically be used by people that are not technology-savvy, but they would be restricted to only using the original setup, which is limited in the number of devices and sensors. Also, since a lot of customizability is provided, the Admin has to at least be aware of the impact that his adjustments can have on the system.

2.3.2. Standard User Characteristics

The standard User only needs the basic ability to operate a smartphone app. They need to be registered on the system to access it.

2.4. Constraints

The system will be designed and implemented based on the contents mentioned in this document. Other details are designed and implemented by selecting the direction preferred by the developer, but the following items are observed.

- Use the technology that has already been widely proven.
- Avoid using technology or software that requires a separate license or pays for royalty. (Exclude this provision if this is the only technology or software that the system must).
- Decide in the direction of seeking improvement of overall system performance.
- Decide in a more user-friendly and convenient direction.
- Consider future scalability and availability of the system
- Optimize the source code to prevent waste of system resources
- Consider future maintenance and add sufficient comments when writing the source code
- Each hardware has the flexibility to be well integrated and applied in a practical home.

- Connect each device using the Enviro Monitor open source.
- It builds servers so that databases can be analyzed smoothly in real time.
- It sets universal standards for air quality but allows manual adjustment to suit individual characteristics.
- Since the user is diverse, men and women of all ages, a front design that can generally feel intimacy is applied.
- When configuring the Arduino board, it is designed so that issues such as short circuits do not occur.
- It pursues universality by recognizing the diversity of each connected device and designing it through universal characteristics.

2.5. Assumptions and Dependencies


This document assumes the state of the system as it is when it is first put together and delivered. Due to the system's high modifiability through the Admin, some of the functionalities might change or be expanded upon. Admin discretion is advised.

3. Specific Requirements

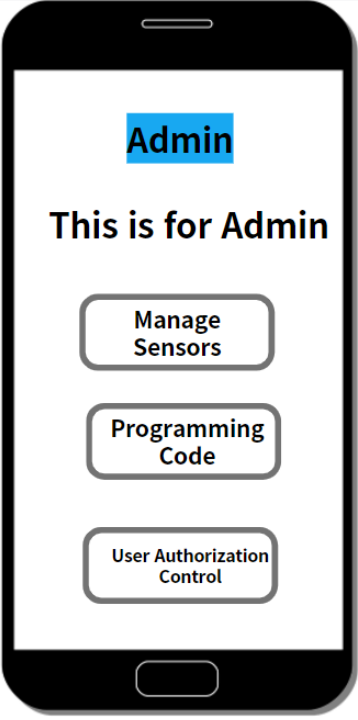
3.1. External Interface Requirements

3.1.1. User Interfaces

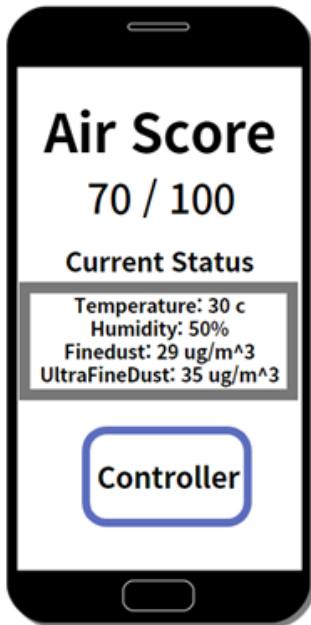
[Table 3] User interface – Login page

| Name | Login page |
|------------------------------------|--|
| Sample image |  <p>[Figure 1] Login page</p> |
| Purpose / description | <p>Make user login or sign up</p> <p>Can move to main page or admin page</p> |
| Input source | User (by clicking button) |
| Output destination | Smartphone (react to user input) |
| Range / Accuracy / Margin of error | <p>Application error (ex. due to CPU shortage)</p> <p>Database error(unable to connect to database)</p> |
| Time | <p>Synchronous change of page</p> <p>Synchronous Login</p> |
| Data type | Text + Button |

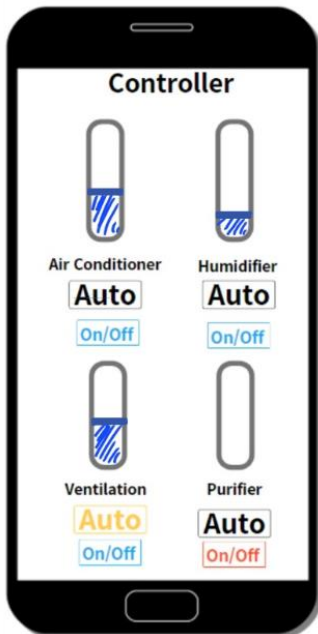
[Table 4] User interface – Admin page

| Name | Admin page |
|------------------------------------|--|
| Sample image |  <p data-bbox="837 1122 1086 1155">[Figure 2] Admin page</p> |
| Purpose / description | <p data-bbox="815 1223 1107 1252">Can add or remove sensors</p> <p data-bbox="699 1285 1224 1314">Can adjust air quality control programming code</p> <p data-bbox="799 1317 1123 1346">Can control user authorization</p> |
| Input source | User (by clicking button) |
| Output destination | Back-End (react to user input) |
| Range / Accuracy / Margin of error | Application error (ex. due to CPU shortage) |
| Time | Synchronous change of page(option) |
| Data type | Text + Button |

[Table 5] User interface – Main page

| Name | Main page |
|------------------------------------|--|
| Sample image |  <p>[Figure 3] Main page</p> |
| Purpose / description | Show current score / current status Can move to controller page |
| Input Source | User (by clicking button to move to another page) |
| Output Destination | Smart phone(changed page) |
| Range / Accuracy / Margin of error | Application error (ex. due to CPU shortage) |
| Time | Synchronous change of page |
| Data type | Text + Image |

[Table 6] User interface – Controller page

| Name | Control page |
|------------------------------------|--|
| Sample image |  <p>[Figure 4] Control page</p> |
| Purpose / description | Control each device by choosing automatic or manual mode |
| Input Source | User (clicking button(make it automatic or not) + scrolling bar (Control it manually)) |
| Output Destination | Smartphone |
| Range / Accuracy / Margin of error | Application error (ex. Due to CPU shortage) |
| Time | Synchronous change of page |
| Data type | Text + Image |

3.1.2. Hardware Interfaces

[Table 7] Hardware interface - Smartphone

| Name | Smart phone |
|-----------------------|---|
| Purpose / Description | Must satisfy condition for application to work properly |
| Software Condition | Android OS 5.0 and above IOS 6.1.6 and above |
| Input Source | User |
| Output Destination | Back-End |

[Table 8] Hardware interface - Computer

| Name | Computer |
|-----------------------|--|
| Purpose / Description | To run back-end. |
| Condition | Use x86 Processor |
| Input Source | User(smart phone) |
| Output Destination | Database / Arduino board(or Arduino IDE) |

[Table 9] Hardware interface – Air controlling devices

| Name | Air controlling devices |
|-----------------------|--|
| Purpose / Description | Must be able to communicate with application in order to enable application to control devices |
| Condition | Support Wi-Fi connection |
| Input Source | Arduino Board |

[Table 10] Hardware interface – Air monitoring sensors

| Name | Air controlling devices |
|-----------------------|--|
| Purpose / Description | Must be able to connect to system to send information about current air status |
| Condition | Support Wi-Fi connection |

| Name | Air controlling devices |
|--------------------|---|
| Input Source | Indoor Air |
| Output Destination | Arduino Board |
| Types | CO2 Sensor / Fine-dust(ultra fine-dust) sensor / Humidity sensor / temperature sensor |

[Table 11] Hardware interface – Arduino Board

| Name | Arduino Board |
|-----------------------|---|
| Purpose / Description | To gather information from air monitoring sensors and control air controlling devices |
| Version | Arduino Nano R3(proper for Air monitoring) |
| Condition | Support Wi-Fi connection |
| Input source | Sensor / User |
| Output destination | Database(Information collected from sensor) / Air controlling Devices(to control it) |
| Maximum Connection | Up to 8 connections (including device and sensor) |

3.1.3. Software Interfaces

[Table 12] Software interface – Firebase

| Name | Firebase(real-time Database for Android studio) |
|-----------------------|---|
| Purpose / Description | Database for storing air quality information |
| Version | 8.0.0 |
| Input source | System |

| | |
|------------------------------------|---|
| Output destination | System (System -> Firebase -> System) |
| Range / Accuracy / Margin of error | Depends on Firebase performance, status |
| Units of Measure | Query |
| Time | Synchronous reaction to given input |
| Data type | Query |

[Table 13] Software interface - Arduino IDE

| Name | Arduino IDE |
|------------------------------------|---|
| Purpose / Description | To program codes for Arduino Board |
| Version | 1.6.0 |
| Input source | C++ code from User |
| Output destination | Arduino Board(Hardware) |
| Range / Accuracy / Margin of error | Depend on Network condition / Arduino IDE performance |
| Data type | C++ language |

3.1.4. Communication Interfaces

[Table 14] Communication between user and air controlling devices

| Name | User and air controlling devices |
|-----------------------|--|
| Purpose / Description | User sends a command to control air controlling devices(It can be manual or automatic) |
| Process (Manual) | User -> Back-End -> Arduino Board -> Air controlling device |
| Process (Automatic) | User -> Back-End -> Arduino Board -> Air monitoring sensor -> Arduino Board -> Back-end -> Database -> Back-End -> Arduino Board -> Air controlling device |

[Table 15] Communication between User and Backend

| Name | User and Backend |
|-----------------------|--------------------------------------|
| Purpose / Description | User sends a command data to Backend |
| Connection Method | Internet |

[Table 16] Communication between backend and Arduino Board

| Name | Backend and Arduino board |
|-----------------------|---|
| Purpose / Description | backend sends data to arduino board and vice versa |
| Connection Method | Internet |
| Data Encryption | Use symmetric key cryptography(key prestalled in backend and arduino) |

[Table 17] Communication between Arduino Board and Air control / monitoring devices(sensors)

| Name | Arduino board and devices(sensors) |
|-----------------------|---|
| Purpose / Description | Air monitoring sensors send data to arduino board and arduino board sends commands to air controlling devices |
| Connection Method | Cables |

3.2. Functional Requirements

3.2.1. Use Case Example

[Table 18] Log in and authorization Process

| Use case name | Description |
|----------------|---|
| Actor | User, Admin, Authentication system. |
| Description | The user should enter an ID and password to log in. |
| Normal course | <ol style="list-style-type: none">1. The app asks the users if they have logged in before.2. If the user has not logged in before.<ol style="list-style-type: none">2.1. The authentication system requires an ID and password from the user.2.2. When the user enters an ID and password, the system validates the ID and password in the database and approves the login if it matches.3. If the user has logged in before.<ol style="list-style-type: none">3.1. The authentication system allows the user to automatically log in based on the previous record without asking the user for an ID and password. |
| Precondition | The user has downloaded the app, but not logged in. |
| Post Condition | <ol style="list-style-type: none">1. If the log in succeeds, move to the main page. |

| | |
|-------------|--|
| | 2. If the user is admin, move to the admin page. |
| Assumptions | The user already signed up their information. |

[Table 19] Main Page

| Use case name | Description |
|----------------|--|
| Actor | User, Air control system. |
| Description | The user may check the current air score and use several functions. |
| Normal course | <p>1. The app displays Air Score, current status (temperature, humidity, fine dust, ultra-fine dust), and a controller button on the screen.</p> <p>2. Current status shows data received from each sensor.</p> <p>3. If the user clicks the controller button, move to the Controller page.</p> |
| Precondition | The user is already logged in. |
| Post Condition | - N/A |
| Assumptions | - N/A |

[Table 20] Admin Page

| Use case name | Description |
|----------------|--|
| Actor | Admin, Air control system. |
| Description | Admin can add or remove sensors, modify the system code, and control user authorizations. |
| Normal course | 1. The app displays Manage sensors button, Programming code button, and user authorization control button. |
| Precondition | The admin is already logged in. |
| Post Condition | - N/A |
| Assumptions | - N/A |

[Table 21] Controller Page

| Use case name | Description |
|---------------|---|
| Actor | User who touched the controller button on the main page, Air control system. |
| Description | It displays the auto function on/off button of the 4 related devices. It shows statuses of four machines and the control page of each machine. |
| Normal course | 1. List the auto function on/off buttons of the air conditioner, Humidifier, Ventilation, and Purifier. |

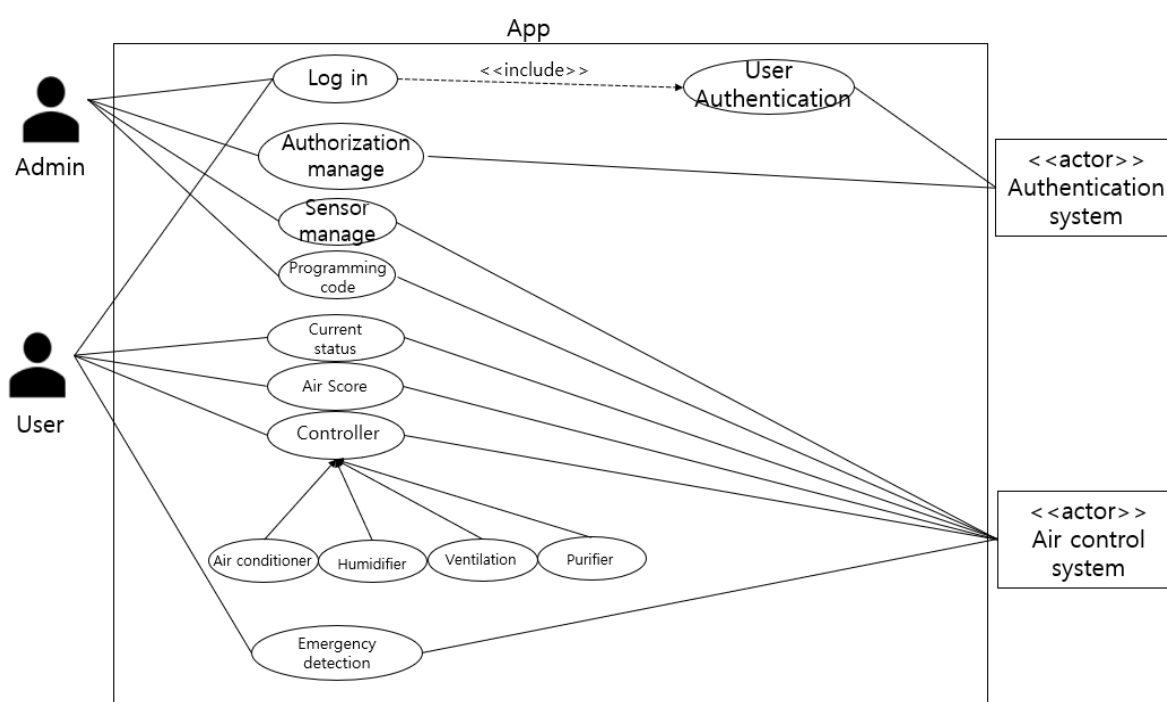
| | |
|----------------|--|
| Precondition | The user should be in a logged-in status and connected to the network. |
| Post Condition | - The user must access the controller page. |
| Assumptions | - N/A |

[Table 22] Emergency detection

| Use case name | Description |
|---------------|---|
| Actor | User who wants to use the Emergency alert, Air control system. |
| Description | Recognize the emergency situation through the data received from the sensors, notify the user, and report to 119. |
| Normal course | <ol style="list-style-type: none"> 1. Check the data received from the temperature sensor and the CO2 sensor if it exceeds the specific value. 2. If the data exceeds the specific value, recognize the emergency situation, and report to 119. 3. Operate alert device and sprinklers in the house. 4. Notify the user through the sound and vibration of the mobile device. |
| Precondition | <ul style="list-style-type: none"> - The user may turn on the emergency alert function from the main page. - The device must be connected to Wi-Fi. |

| | |
|----------------|-------|
| Post Condition | - N/A |
| Assumptions | - N/A |

3.2.2. Use Case Diagram



[Figure 5] Use case diagram

3.2.3. Data Dictionary

[Table 23] Table of Data Dictionary - User

| Field | Key | Constraint | Description |
|-------|-------------|------------|-------------|
| ID | Private Key | NOT NULL | user's ID |

| | | | |
|-----------------|-------------|----------|---------------------|
| PASSWORD | | NOT NULL | user's password |
| IP ADDRESS | Private Key | NOT NULL | user's ip address |
| NICK NAME | | NOT NULL | user nickname |
| CONTROL MACHINE | | | air control machine |
| STATUS | | | current status |

[Table 24] Table of Data Dictionary - System Control

| Field | Key | Constraint | Description |
|------------------|-----|------------|---------------------|
| CONTROL CATEGORY | | NOT NULL | Purifier |
| | | | Air conditioner |
| | | | Ventilation |
| | | | Humidifier |
| | | | Emergency detection |
| | | | Authorization |

[Table 25] Table of Data Dictionary - Purifier

| Field | Key | Constraint | Description |
|-----------------|-----|------------|-------------|
| Fine dust | | NOT NULL | |
| Ultra fine dust | | NOT NULL | |
| temperature | | NOT NULL | |

| | | | |
|-------------------------|-------------|----------|--|
| Stink detection | | | |
| Harmful detection | | | |
| IP address | Private Key | NOT NULL | |
| Dust removal efficiency | | | |
| O3 density | | NOT NULL | |

[Table 26] Table of Data Dictionary - Air conditioner

| Field | Key | Constraint | Description |
|-------------------------|-------------|------------|-------------|
| Cooling efficiency | | | |
| Carbon emission | | | |
| Inside temperature | | NOT NULL | |
| IP address | Private Key | NOT NULL | |
| Electricity consumption | | NOT NULL | |
| Outside temperature | | NOT NULL | |
| Humidity | | NOT NULL | |

[Table 27] Table of Data Dictionary - Ventilation

| Field | Key | Constraint | Description |
|------------------|-----|------------|-------------|
| Stink detection | | | |
| Inside fine dust | | | |

| | | | |
|--------------------------|-------------|----------|----------------------------------|
| Outside fine dust | | | |
| Inside ultra fine dust | | | |
| Outside ultra fine dust | | | |
| Inside temperature | | NOT NULL | |
| IP address | Private Key | NOT NULL | |
| Ventilation opening rate | | | Show how much ventilation opened |
| Outside temperature | | NOT NULL | |
| Weather status | | NOT NULL | snow, rain, wind ..etc |
| Outside humidity | | NOT NULL | |
| Inside humidity | | NOT NULL | |

[Table 28] Table of Data Dictionary - Humidifier

| Field | Key | Constraint | Description |
|---------------------|-------------|------------|------------------------|
| Inside temperature | | | |
| IP address | Private Key | NOT NULL | |
| Dust in filter | | | |
| Outside temperature | | | |
| Weather status | | NOT NULL | snow, rain, wind ..etc |
| Outside humidity | | NOT NULL | |
| Inside humidity | | NOT NULL | |

[Table 29] Table of Data Dictionary - Emergency detection

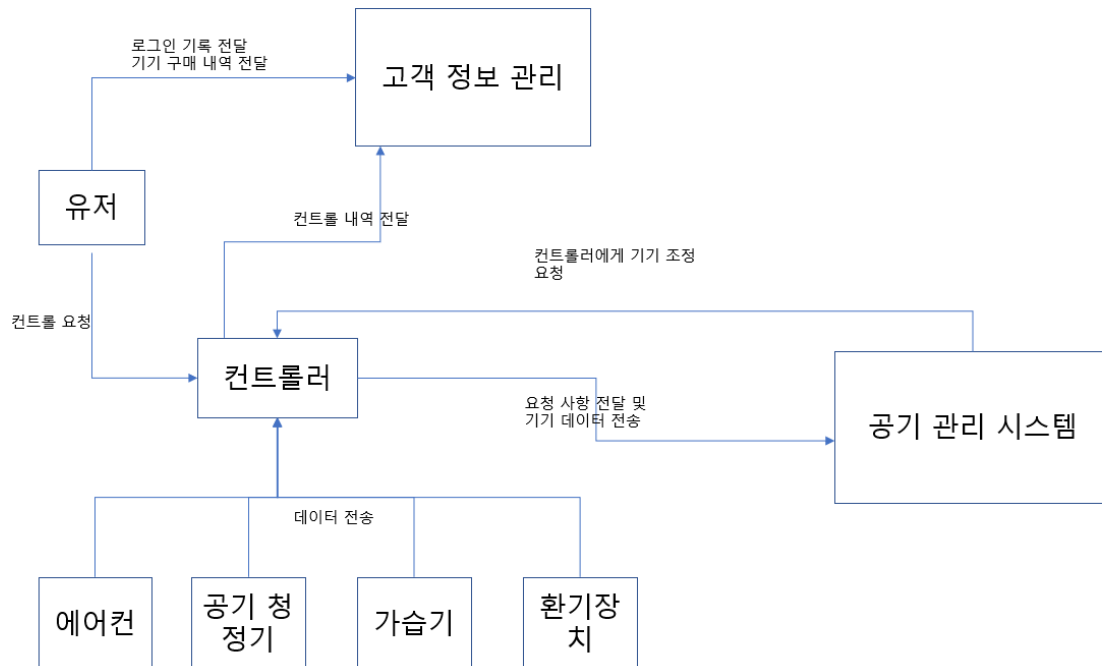
| Field | Key | Constraint | Description |
|---------------------------|-------------|------------|------------------------|
| Stink detection | | NOT NULL | |
| Inside fine dust | | NOT NULL | |
| Outside fine dust | | NOT NULL | |
| Inside ultra fine dust | | NOT NULL | |
| Onside ultra fine dust | | | |
| Inside temperature | | NOT NULL | |
| IP address | Private Key | NOT NULL | |
| harmful detection | | NOT NULL | |
| Outside temperature | | NOT NULL | |
| Weather status | | | snow, rain, wind ..etc |
| Outside humidity | | | |
| Inside humidity | | | |
| Internal motion detection | | NOT NULL | |
| CO / CO2 amount detection | | NOT NULL | |

[Table 30] Table of Data Dictionary - Authorization

| Field | Key | Constraint | Description |
|-------|-------------|------------|-------------|
| ID | Private Key | NOT NULL | |

| | | | |
|----------------------|-------------|----------|----------------------------|
| IP address | Private Key | NOT NULL | |
| Password | | NOT NULL | |
| Nickname | | NOT NULL | |
| System usage history | | | |
| User's machine type | | NOT NULL | Show what machine user has |
| Log in history | | NOT NULL | |

3.2.4. Data Flow Diagram



[Figure 6] Data flow diagram

3.3. Nonfunctional Requirements

3.3.1. Product Requirements

The product requirements specify the requirements related to the quality of operation of the product. Product requirements include usability, efficiency, dependability, and portability requirements. The requirements for each item are as follows.

3.3.1.1. Usability Requirements

This system plays a role in helping the user to more easily control air quality related devices and to find the optimal air quality. Therefore, the user application of this system should be able to check and control all operation status of each device on one screen, check the numerical data of each item related to air quality, and determine the current overall air quality condition based on this. In addition, the user application should be able to inform the user of an operation to form an air condition better than the current air condition, or suggest a method for

performing the operation with a simple control.

Each numerical value of air quality should adopt a GUI that users can recognize at a glance. For this, you can use a GUI in the Slider format.

When the application is first executed, the user application should be able to introduce the user how to use it and provide the user with a user manual.

The user application should be able to notify the user of an error condition in which each device does not operate properly. Predictable error conditions may include equipment failure, preliminary procedures for operation such as filter replacement, and connection instability. The user application should notify the user of this error condition using a method such as a warning window, and if the error condition is resolved, check it and notify the user that the error has been resolved.

3.3.1.2. Dependability Requirements

This system does not bring critical crash that can affect the whole system when user operation fails. However, frequent operation failures reduce user usability, so the higher the dependability, the better. Therefore, in order to provide a good user experience, no more than 1 failure per 100 operations should be tolerated for this product. This failure rate is the failure rate for the integrated operation of the air quality control system and the user applications that make up whole system. Therefore, the failure rate for each system (application and air quality control system) should not exceed 0.5% to achieve that failure rate, when assuming that the failure of each system does not affect other systems.

3.3.1.3. Efficiency Requirements

The Arduino board used to configure the air quality control system of this system has the following memory specifications depending on the type.

[Table 31] Arduino board memory specification

| | ATmega328P | ATmega2560 |
|---------------------|--------------------------|-------------------------|
| Flash memory | 32KB (Boot loader 0.5KB) | 256KB (Boot loader 8KB) |
| SRAM | 2KB | 8KB |
| EEPROM | 1KB | 4KB |

As above, since the memory space of the Arduino board is smaller than that of a general PC or smartphone, the size of data used for code and operation should be appropriately used.

Assuming that ATmega2560 model board is used, the air quality control program code written in C++ is stored in the flash memory, so the size of the program code should not exceed 248KB excluding the size of the boot loader. In addition, free space should be left for future maintenance and change management.

Various data used as variables during code execution (air quality data collected from sensors, user input, etc.) are stored in SRAM, which should not exceed 8KB. In addition, semi-permanently stored data during program execution, such as user setting values of applications, are stored in EEPROM, and the program should be designed so that it does not exceed 4KB.

3.3.1.4. Probability Requirements

User applications should be able to be installed and used on all smartphones. The program should be written to work on both Android and IOS.

3.3.1.5. Security Requirements

It should not be possible for an external intruder to operate other users' devices. User ID and password for login are stored in DB and must be protected safely. At this time, the password must not be stored as it is, but must be encrypted and stored in the DB as an encryption key.

The administrator must not know the user's password, and for this, the password authentication method must be implemented as a one-way method. According to this method, anyone who knows the password can obtain the encryption key stored in the DB, but cannot obtain the password with the encryption key.

3.3.2. Organizational Requirements

Organizational requirements are requirements related to customer and developer organization policies or procedures. In this document, as organizational requirements, the development requirements specifying the system development language, etc., and the environmental requirements for specifying the development environment and standard processes were prepared.

3.3.2.1. Development Requirements

For writing user application programs, the Android platform uses JAVA to write programs, and the IOS platform uses Swift to write programs. Javascript is used to write the back-end program, and MySQL is used as the database management system. The air quality control program to be mounted on the Arduino board is written in C++.

3.3.2.2. Environmental Requirements

Android applications are developed using Android Studio, and IOS applications are developed using the Xcode integrated development environment. For backend development, it is developed in Linux OS environment using the Node.js platform and the Express.js framework for web communication with user applications. The air quality control program uses the Arduino IDE development environment.

3.3.2.3. Operational Requirements

Addition and removal of sensors connected to the Arduino board should be free. The air quality control program must be programmed to accommodate the changes in the collected data or upload a new program to the Arduino board to match the changes. User programs should also allow users to update online to display new air quality data.

3.3.3. External Requirements

3.3.3.1. Legislative Requirements

3.3.3.1.1. Privacy Requirements

User consent is required to collect and store indoor air quality data from sensors.

3.3.3.1.2. Safety Requirements

Information such as user login information, setting values, and historical data is centrally managed in one database. The data stored in the database should be backed up periodically in case of damage caused by a disaster or intrusion from outside. The backup cycle is performed once every 4 months, a total of 4 times a year, based on the season when the outside temperature or humidity changes significantly. User login information is encrypted and stored.

When the air quality control program automatically operates each device, it must be controlled so that overcurrent does not flow. Some devices (such as air conditioners) can consume a lot of power, so adjust the output when using them or control the number of devices running at the same time.

3.4. Performance Requirements

The system must satisfy the minimum performance that the user will not feel uncomfortable in managing devices through the application. For this, the user's device operation command through the application must be executed through the actual devices within 1 second.

3.5. Design Constraints

User applications should be designed to be usable on most smartphone and tablet pc ratios.

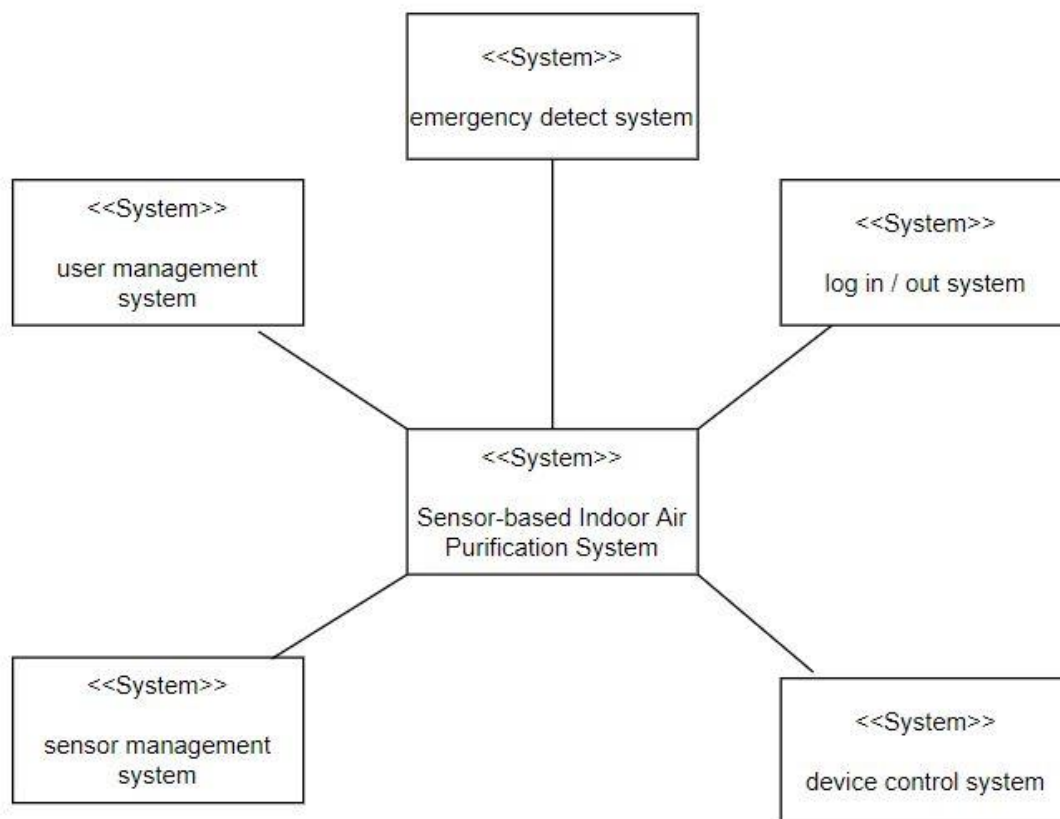
3.6. Standards Compliance

All programs in the system are written according to C++ standards with Arduino IDE which is a software environment in which the editor, compiler, and uploader are combined.

Function and variable names in the program use camel notation and underscore notation apply to the real time database provided by Firebase server.

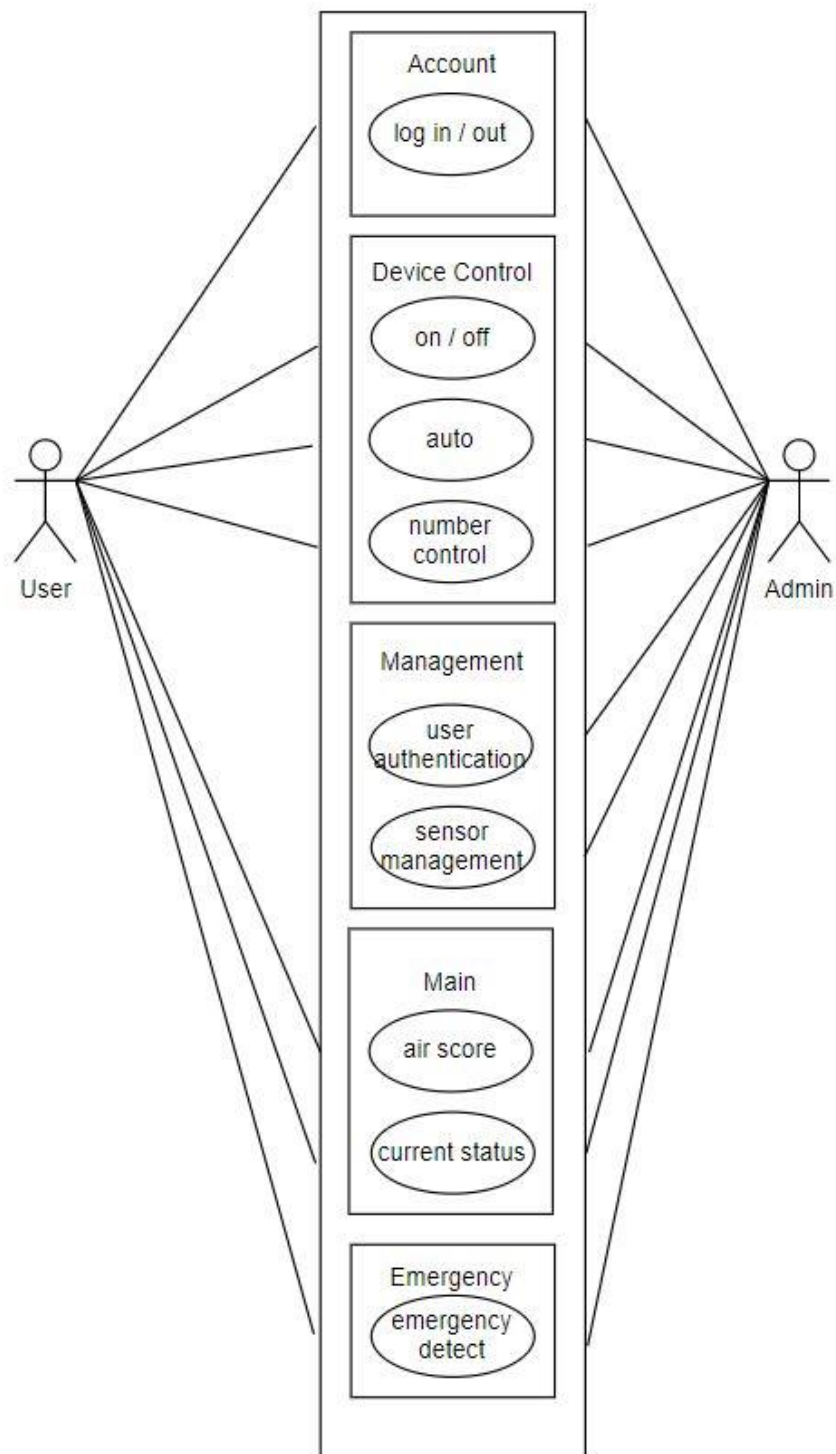
3.7. Organizing the Specific Requirement

3.7.1. Context Model



[Figure 7] Context model

3.7.2. Interaction Model



[Figure 8] Interaction model

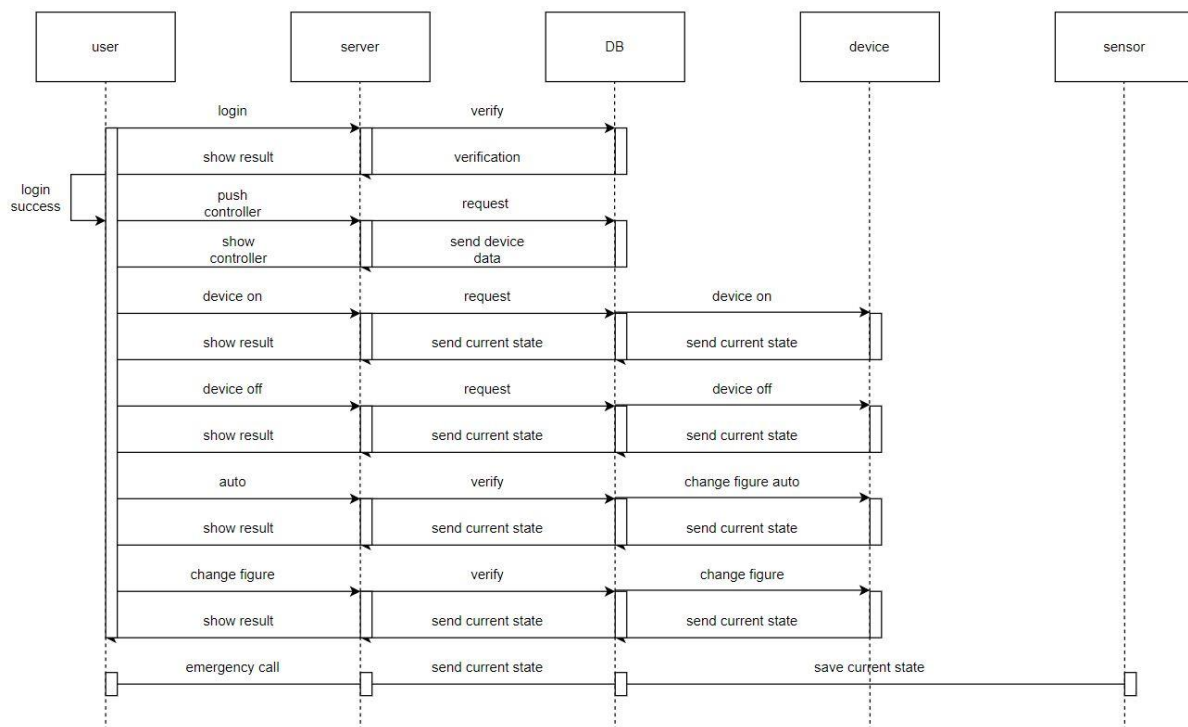
Please see also 3.2.2. Use case Diagram

3.7.3. Behavior Model

3.7.3.1. Data Flow Diagram

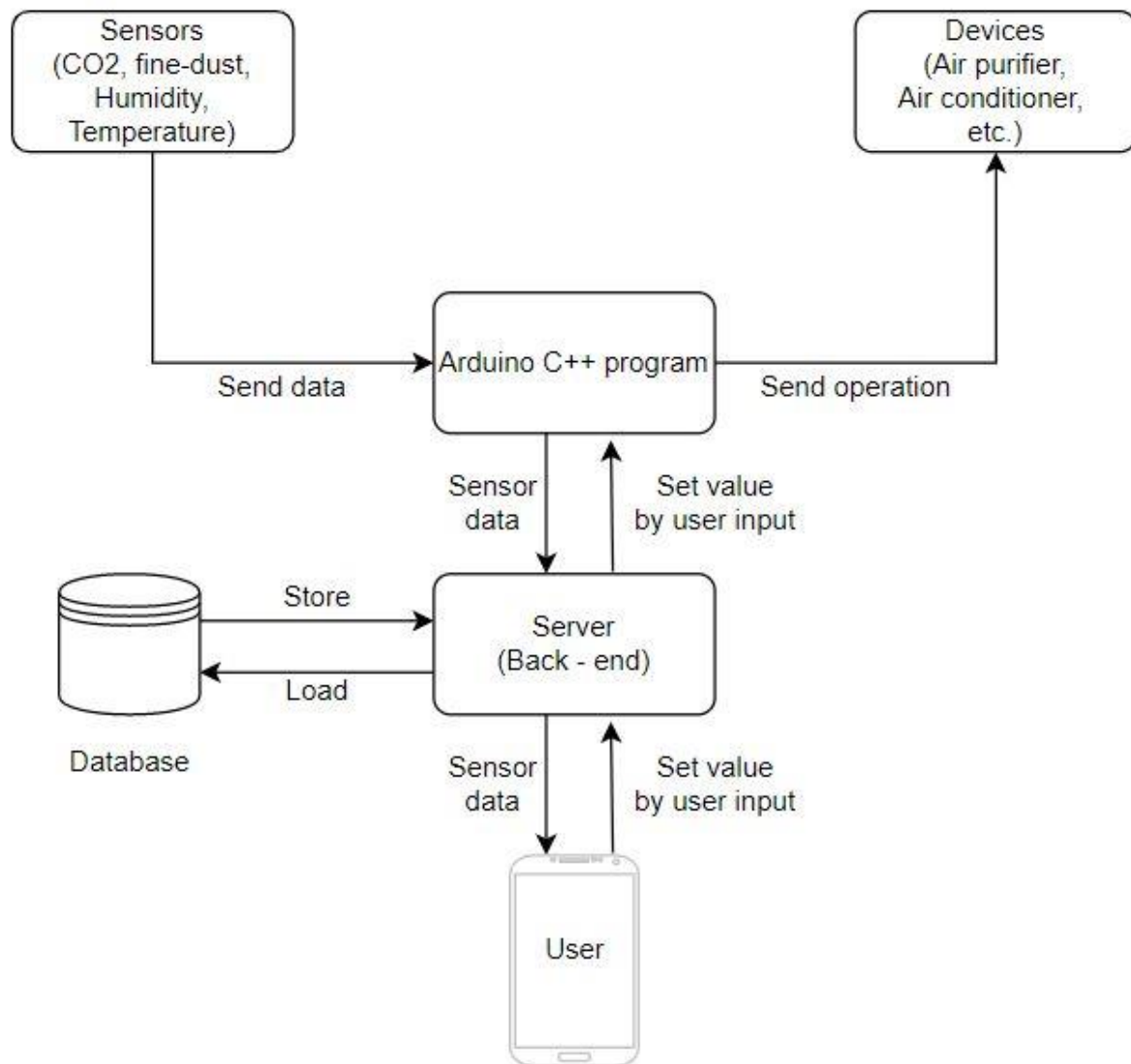
See 3.2.4. Data Flow Diagram

3.7.3.2. Sequence Diagram



[Figure 9] Sequence diagram

3.8. System Architecture



[Figure 10] System architecture

3.9. System Evolution

In this section, we describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, software updates, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.

3.9.1. Limitation and Assumption

- About administration page, it is considered to use only authorized administrator
- About managing sensors option, administrators can add or delete sensors.
- About programming code option, administrator can see programming code, but cannot change
- About user authorization control option, administrators can change each users' authorization.
- About the controller option, users can add or delete devices.
- About the controller option, users can adjust their own temperature, humidity, etc..
- About the auto option, the program adjusts devices with calculated optimal air quality.
- About the controller option, users can add only compatible devices.

3.9.2. Evolutions of Hardware and Change of User Requirements

As hardware develops, users can hold more than 8 devices, and the process works even if the sensor is far away. And as devices become more diverse, software developers need to make more devices compatible with this program.

4. Supporting Information

4.1. Software Requirement Specification

This software requirements specification was written in accordance with the IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830).

4.2. Document History

[Table 32] Document History

| Date | Version | Description | writer |
|------------|---------|----------------------|--------------|
| 2021/04/25 | 0.1 | Overall Style form | Sungjoon Lee |
| 2021/04/26 | 1.0 | Addition of 1.1.1 | Sungjoon Lee |
| 2021/04/26 | 1.1 | Addition of 1.1.2 | Joonsun Baek |
| 2021/04/26 | 1.2 | Addition of 1.2 | Maik Helbig |
| 2021/04/26 | 1.3 | Addition of 1.5 | Hangyu Kim |
| 2021/04/27 | 1.4 | Addition of 2.1~2.3 | Maik Helbig |
| 2021/04/27 | 1.5 | Addition of 2.4~2.5 | Seyeon Park |
| 2021/04/28 | 1.6 | Revision of 2.1~2.3 | Maik Helbig |
| 2021/04/28 | 1.7 | Revision of 2.4~2.5 | Seyeon Park |
| 2021/04/29 | 1.8 | Addition of 3.1 | Joonsun Baek |
| 2021/04/29 | 1.9 | Addition of 3.2 | Hangyu Kim |
| 2021/04/29 | 1.10 | Addition of 3.3~3.5 | Jeongmin Cha |
| 2021/04/29 | 1.11 | Addition of 3.6~3.9 | Jinkyu Lee |
| 2021/04/30 | 1.12 | Revision of 3.1 | Joonsun Baek |
| 2021/04/30 | 1.13 | Revision of 3.2 | Hangyu Kim |
| 2021/04/30 | 1.14 | Revision of 3.3~3.5 | Jeongmin Cha |
| 2021/04/30 | 1.15 | Revision of 3.6~3.9 | Jinkyu Lee |
| 2021/04/30 | 1.16 | Addition of 1.3, 1.4 | All |
| 2021/04/30 | 1.17 | Addition of 4 | Jinkyu Lee |
| 2021/05/01 | 1.18 | Overall revision | All |