Sensor-Based

Indoor Air Purification System

Software Design Specification

2022.05.15.

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

LIST OF FIGURES

LIST OF TABLES

1. Preface

This document provides a description of the design used to implement Sensor-Based Indoor Air Purification System software. This chapter contains information pertaining to the audience, scope, structure and purpose of the design document.

1.1 Readership

This document has the main readership SE-2022-1-Team11, which develops a Sensor-Based Indoor Air Purification System. However, professors, assistants, and students in introductory software engineering classes can also become readers. The readership can also be any stakeholder to understand system design.

1.2 Scope

This design specification is used in the program implementation phase and system description in software engineering as the definition of a design used to implement applications that allow users to measure indoor air quality and control related devices through mobile devices.

1.3 Objective

The primary purpose of this software design document is to provide a description of the technical design aspects of sensor-based indoor air purification systems. This document provides the design of the system architecture and the behavior of the program for the implementation of the overall system. The overall system consists of four main systems: front-end, back-end, database, and Arduino program, and a separate chapter in this document provides an overview of the architecture of each system. In addition, we provide a design for the protocol used to interact with each system.

1.4 Document Structure

1. Preface: this chapter includes readership, scope of this document, objective, and document structure.

2. Introduction: this chapter specifies several tools used for this document, several diagrams used in this document and the references, and object of this project.
3. Overall System Architecture: this chapter describes the overall architecture of the system using context diagram, sequence diagram, and use case diagram.
4. System Architecture - Frontend: this chapter specifies architecture of the frontend system using class diagram and sequence diagram.
5. System Architecture - Backend: this chapter specifies architecture of the backend system using class diagram and sequence diagram.
6. Protocol Design: This chapter specifies the protocol design used for communication between the front-end and back-end and Arduino programs..
7. Database Design: this chapter specifies database design using ER diagrams.
8. Testing Plan: this chapter specifies the testing plan.
9. Development Plan: This chapter specifies the tools, development environment, and restrictions used for development.
10. Supporting Information: this chapter describes document history.

2. Introduction

The project began with the growing demand for IoT and the development of smart homes in modern society. We aim to make the air at home safer and fresher. It can also be done automatically and directly controlling the air at home anytime, anywhere. The designs that would be used when implementing the system would be provided in this document and would follow the requirements specified in the Software Requirements Specifications document provided before.

2.1. Objectives

This chapter defines and describes the diagram and tolls we used during the design phase and the scope of this project.

2.2. Applied Diagrams

2.2.1. UML

The Unified Modeling Language(UML) is the most popular way to show the design of a system efficiently. The project uses UML to show good visualization information to readers.

2.2.2. Use Case Diagram

Use Case Diagram is an efficient way to see how users relate to and interact with the system within the system. Use Case Diagram gives a guide to how different types of users and systems should behave. It does not give a specific step-by-step, but it gives a summary of use cases, actors, and the relationship between systems.

2.2.3. Sequence Diagram

A sequence diagram is an action diagram that expresses how a particular action interacts with an object in some order. It has the advantage of indicating what scenarios the systems currently exist are operating in. The sequence diagram allows you to know the use cases such as APIs in detail and model logic such as API calls from other systems, so it is good to understand scenarios.

2.2.4. Context Diagram

The context diagram shows what is outside of the system to be built and what is outside of the entity interacting with the system. For purposes, it identifies external objects interacting with the system. Establish the scope of the system. It is not intended to understand the inside of the system.

2.2.5. Data Flow Diagram

A data flow diagram (DFD) is a diagram that illustrates how data is transformed along each process in software, and is very useful in the analysis and design of software and information

systems. It is also referred to as a data flow chart or a data flow chart. DFD is one of the most commonly used tools for modeling systems and can be very useful when its functions are very complex and important compared to data.

2.3. Applied tools

When creating diagrams, online tools such as diagrams.net and visual-paradigm.com were used, and there are also manually produced diagrams.

2.3.1. Diagram.net

diagrams.net is a free and open source cross-platform graph drawing software developed in HTML5 and JavaScript. Its interface can be used to create diagrams such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams

2.3.2. Visual-paradigm.com

Visual Paradigm is a UML CASE Tool supporting UML 2, SysML and Business Process Modeling Notation from the Object Management Group. In addition to the modeling support, it provides report generation and code engineering capabilities including code generation.

2.4 Project Scope

This service aims to control the air environment among smart homes. Supports sensor-based automatic air control or anytime, anywhere manually accessible control. Because the foundation of this project is sensors, it uses information from sensors to control the machine to create a safe and pleasant air environment.

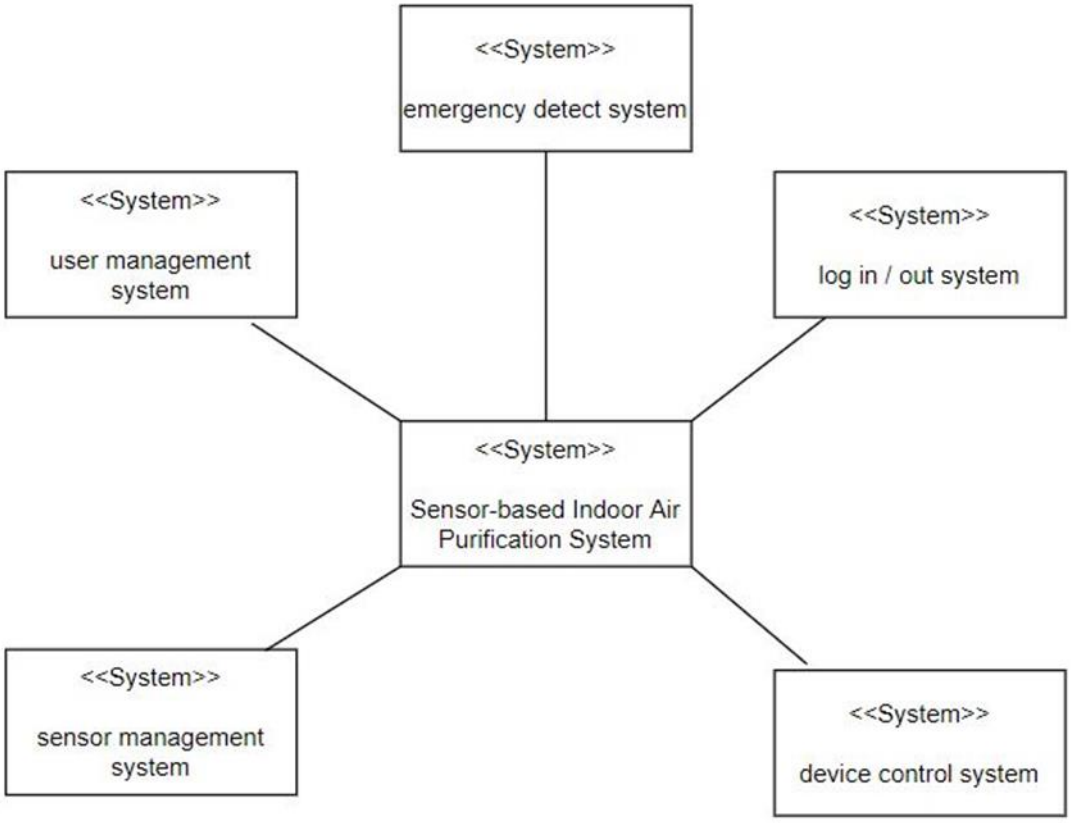
3. System Architecture - Overall

3.1 Objectives

The objective of this chapter is to provide information of how the system is organized

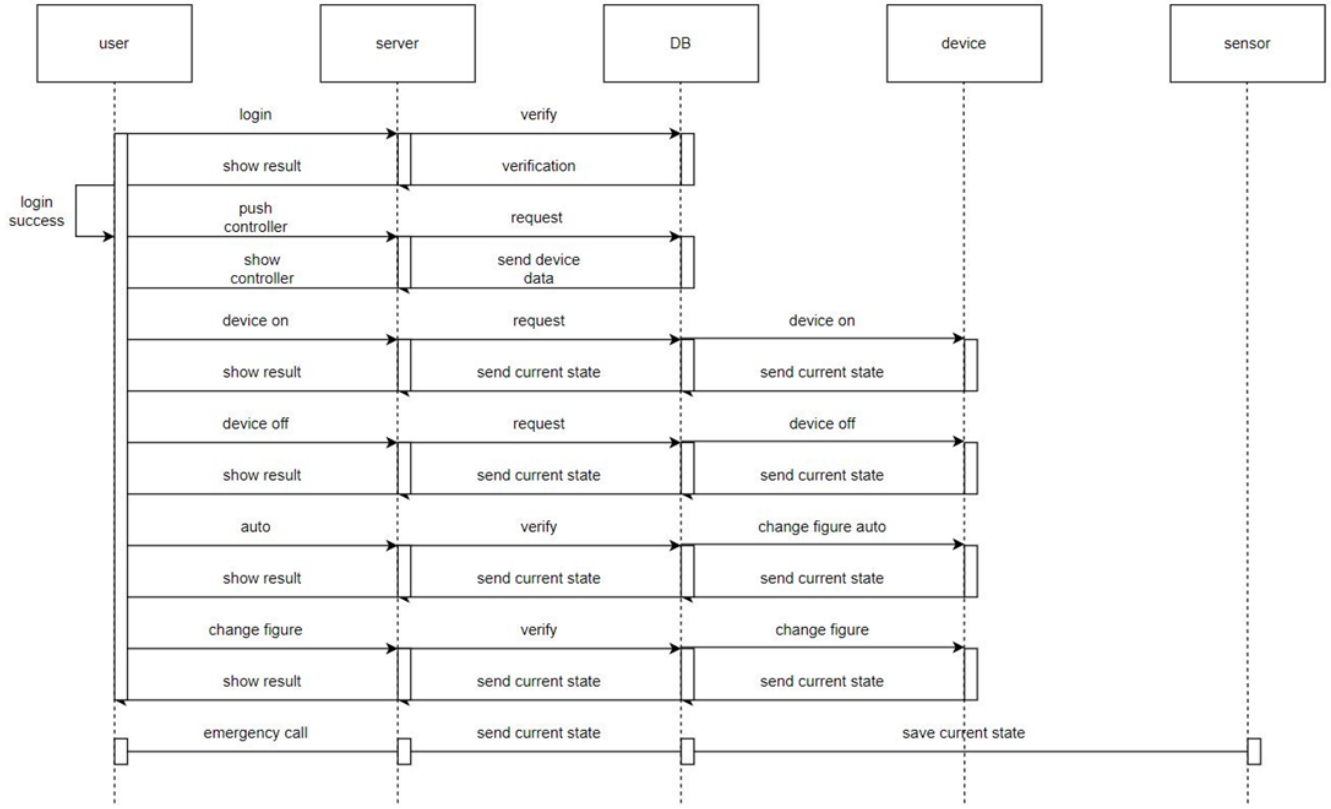
3.2 System Organization

3.2.1 Context Diagram



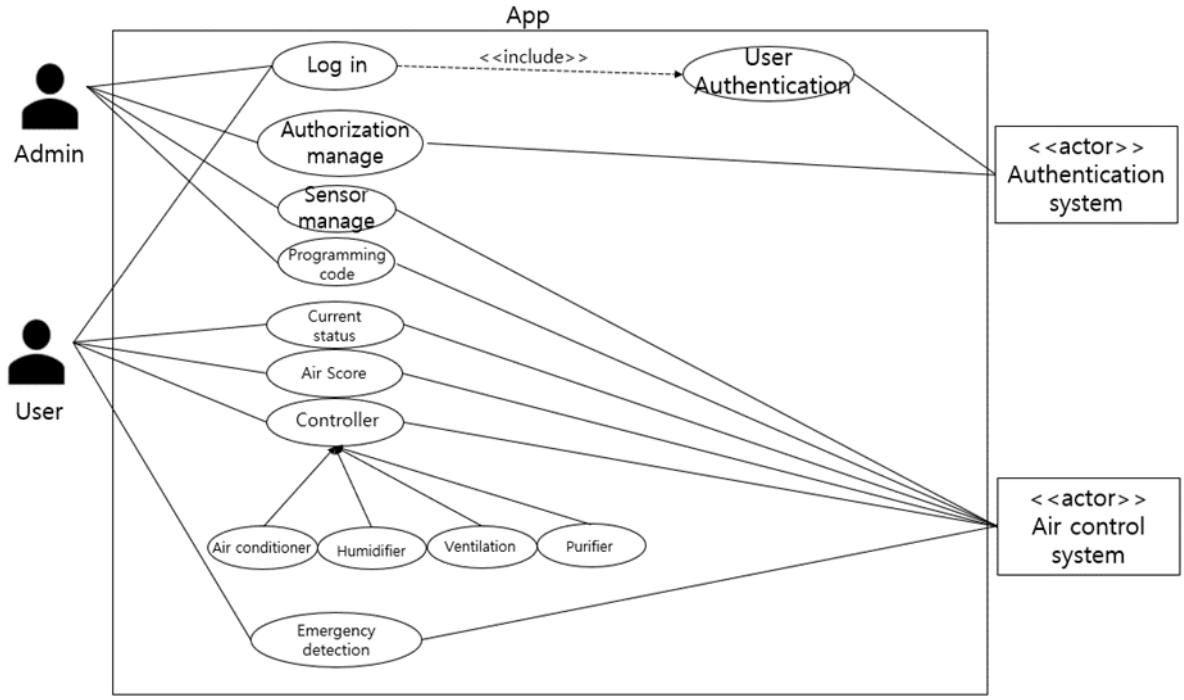
[Figure 1] System Architecture - Overall Context Diagram

3.2.2 Sequence Diagram



[Figure 2] System Architecture – Overall Sequence Diagram

3.2.3 Use Case Diagram



[Figure 3] System Architecture – Overall Use Case Diagram

4. System Architecture - Frontend

4.1 Objectives

This chapter covers the conceptual model of a mobile app program running on the front end of a system. A mobile app consists of four main components, each of which independently interacts with the backend system and makes up the overall system.

4.2 Components

4.2.1 Admin Page

The admin class can be controlled with priority rights to the overall system. You can add or delete devices or sensors, add new code to Arduino, and control the user's operation.

4.2.1.1 Attribute

These are the attributes that the login class has.

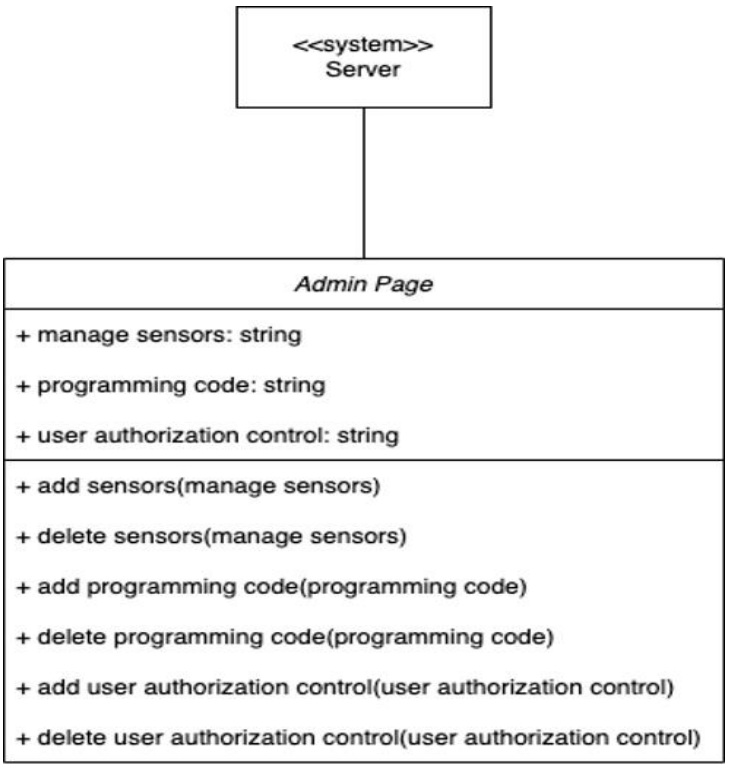
- manage sensors: add or delete sensors
- programming code: add or delete programming code
- user authorization control: add or delete user authorization control

4.2.1.2 Methods

There are the methods that the admin class has.

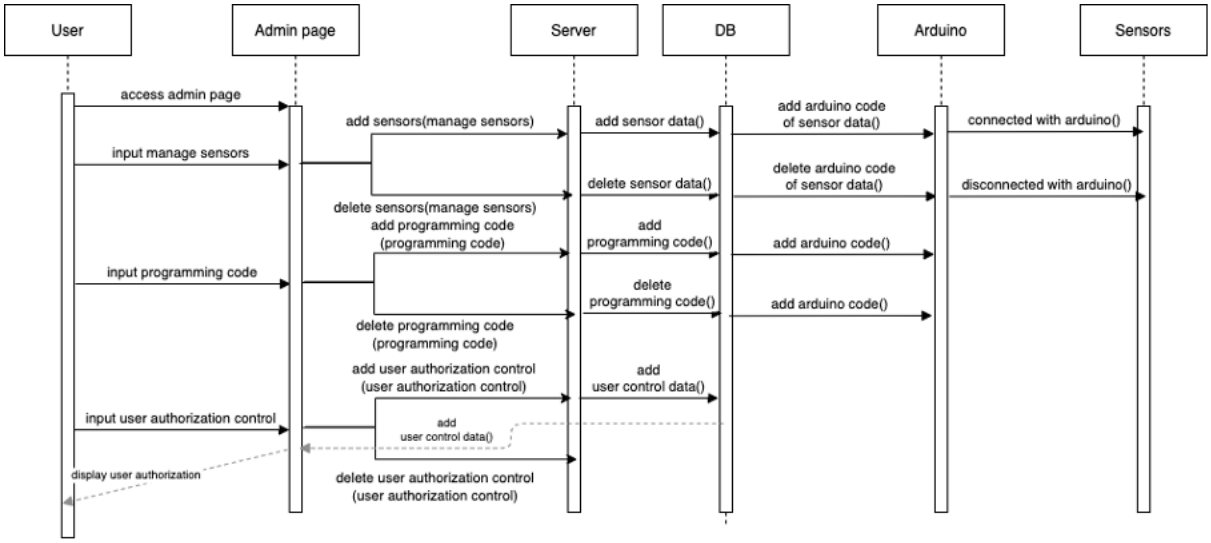
- add sensors()
- delete sensors()
- add programming code()
- delete programming code()
- add user authorization control()
- delete user authorization control()

4.2.1.3 Class Diagram



[Figure 4] Admin Page Class Diagram

4.2.1.4 Sequence Diagram



[Figure 5] Admin Page Sequence Diagram

4.2.2 Login Page

The Login class allows the user to sign up or log in. Depending on the ID of the person who logs in, it can be moved to the user's main page or to the administrator page.

4.2.2.1 Attributes

These are the attributes that the login class has.

- Logged in before: check logged in before or first time
- Id: input user’s id
- password: input users password

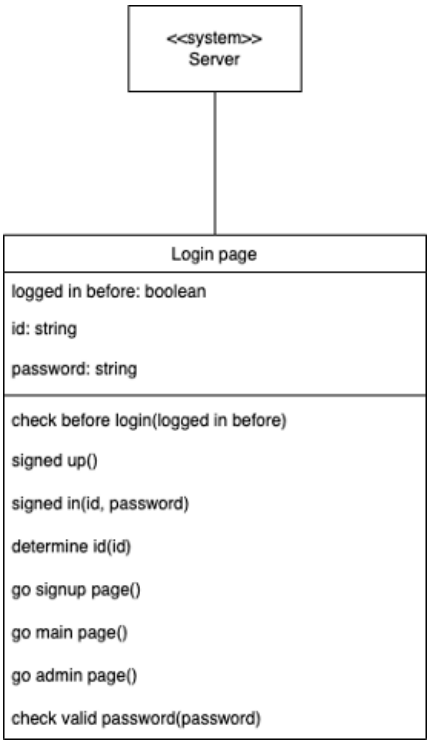
4.2.2.2 Methods

There are the methods that the admin class has.

- check before login()
- signed up()
- signed in()

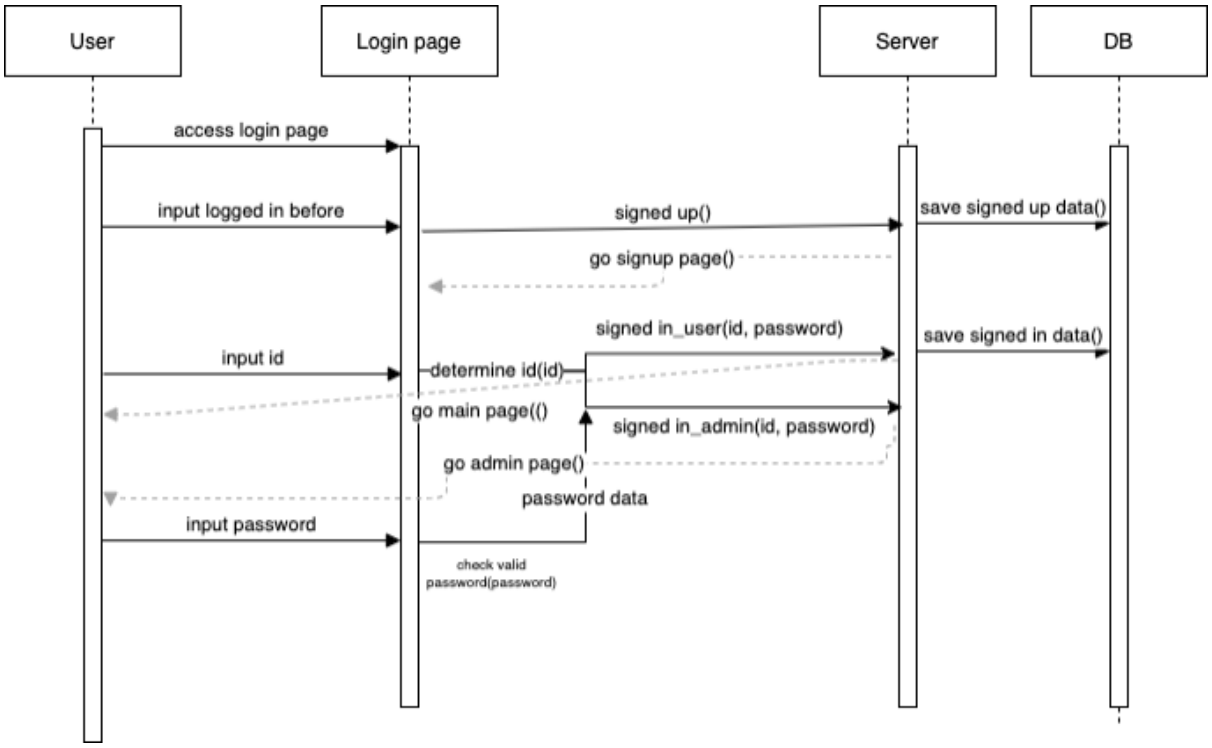
- determine id()
- go signin page()
- go admin page()
- check valid password()

4.2.2.3. Class Diagram



[Figure 6] Login Page Class Diagram

4.2.2.4 Sequence Diagram



[Figure 7] Login Page Sequence Diagram

4.2.3 Main Page

4.2.3.1 Attributes

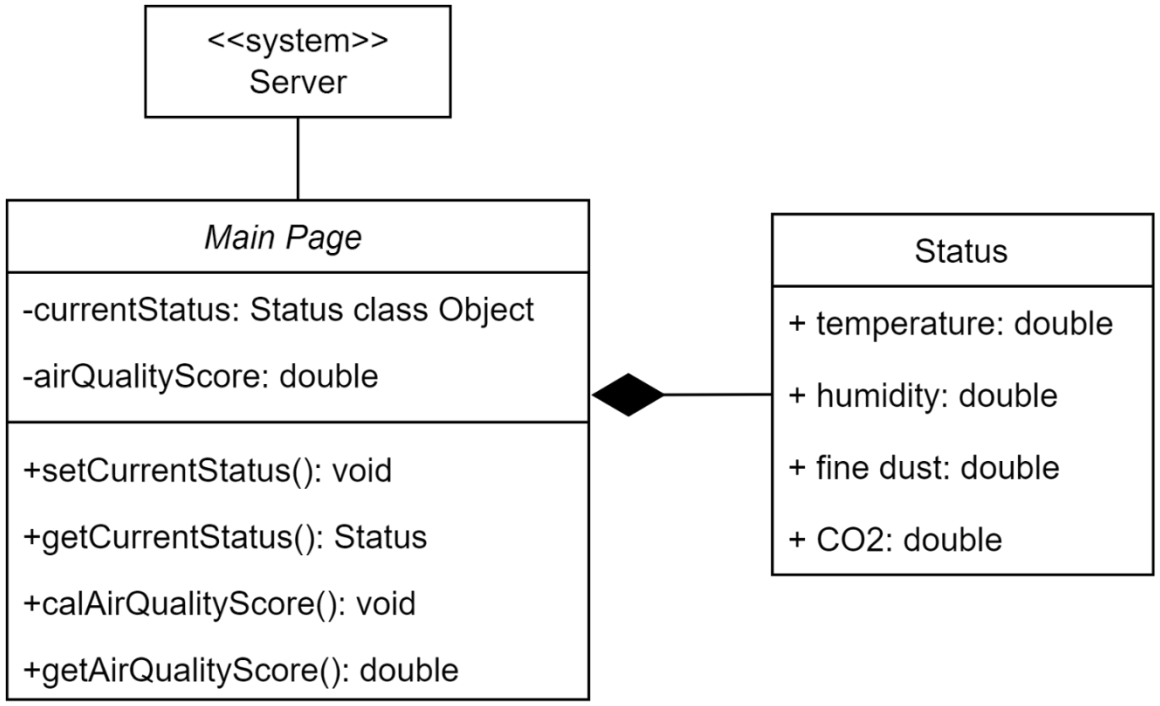
- currentStatus: Status Class Object. As properties of the object, it has temperature, humidity, fine dust concentration, and CO2 concentration.
- airQualityScore: Comprehensive score for air quality

4.2.3.2 Methods

These are methods that Main Page class has.

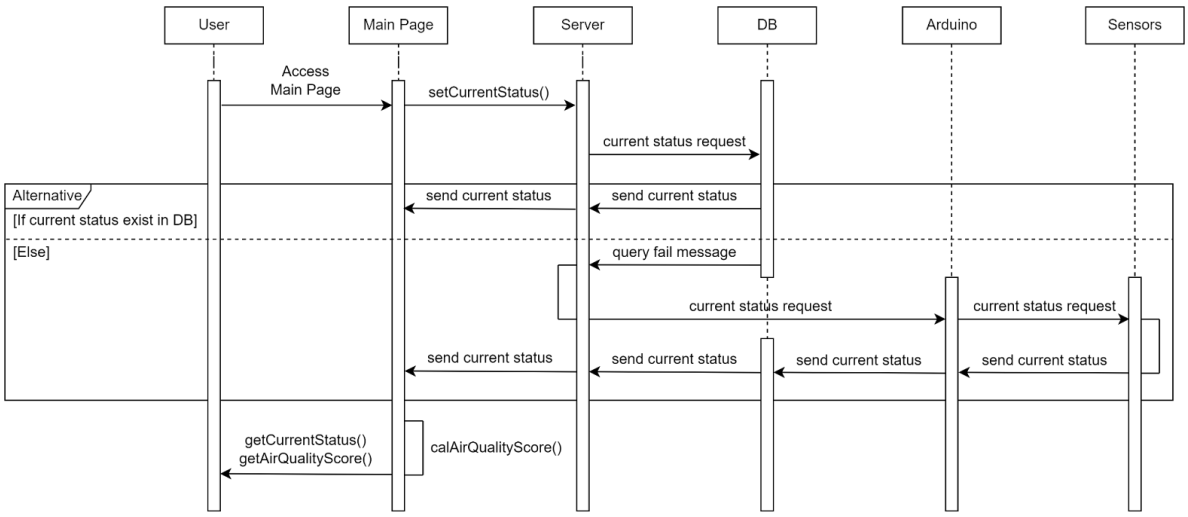
- setCurrentStatus(): A method to receive the current air quality from the server.
- getCurrentStatus(): Returns the value of private field currentStatus by copying.
- calAirQualityScore(): A method that calculates the overall air quality score based on the current air condition.
- getAirQualityScore(): Returns a value of private field airQualityScore by copying

4.2.3.3. Class Diagram



[Figure 8] Main Page Class Diagram

4.2.3.4 Sequence Diagram



[Figure 9] Main Page Sequence Diagram

4.2.4 Control Page

4.2.4.1 Attributes

- airConditioner: A child class of the device class object.
- humidifier: A child class of the device class object.
- ventilation: A child class of the device class object.
- purifier: A child class of the device class object.

4.2.4.2 Methods

- setDevices(): It is called when the Control Page component is created and plays a role in initializing the device objects that exist in the field of the Control Page.

4.2.4.3 SubComponent: < device class object >

4.2.4.3.1 Attributes

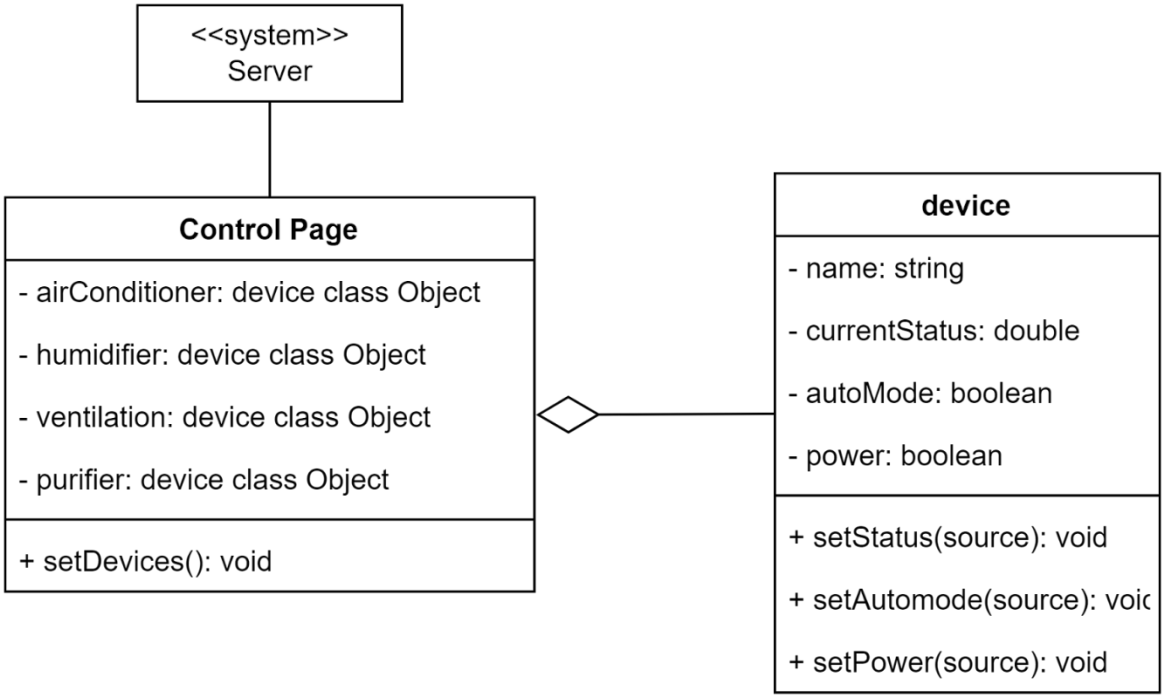
- name: set device's name
- currentStatus: Stores the most current value of status associated with the device.
- autoMode: Stores whether the device is in auto mode.
- power: Stores whether the device is on/off.

4.2.4.3.2 Methods

Every field is updated when parameter of methods is "server"

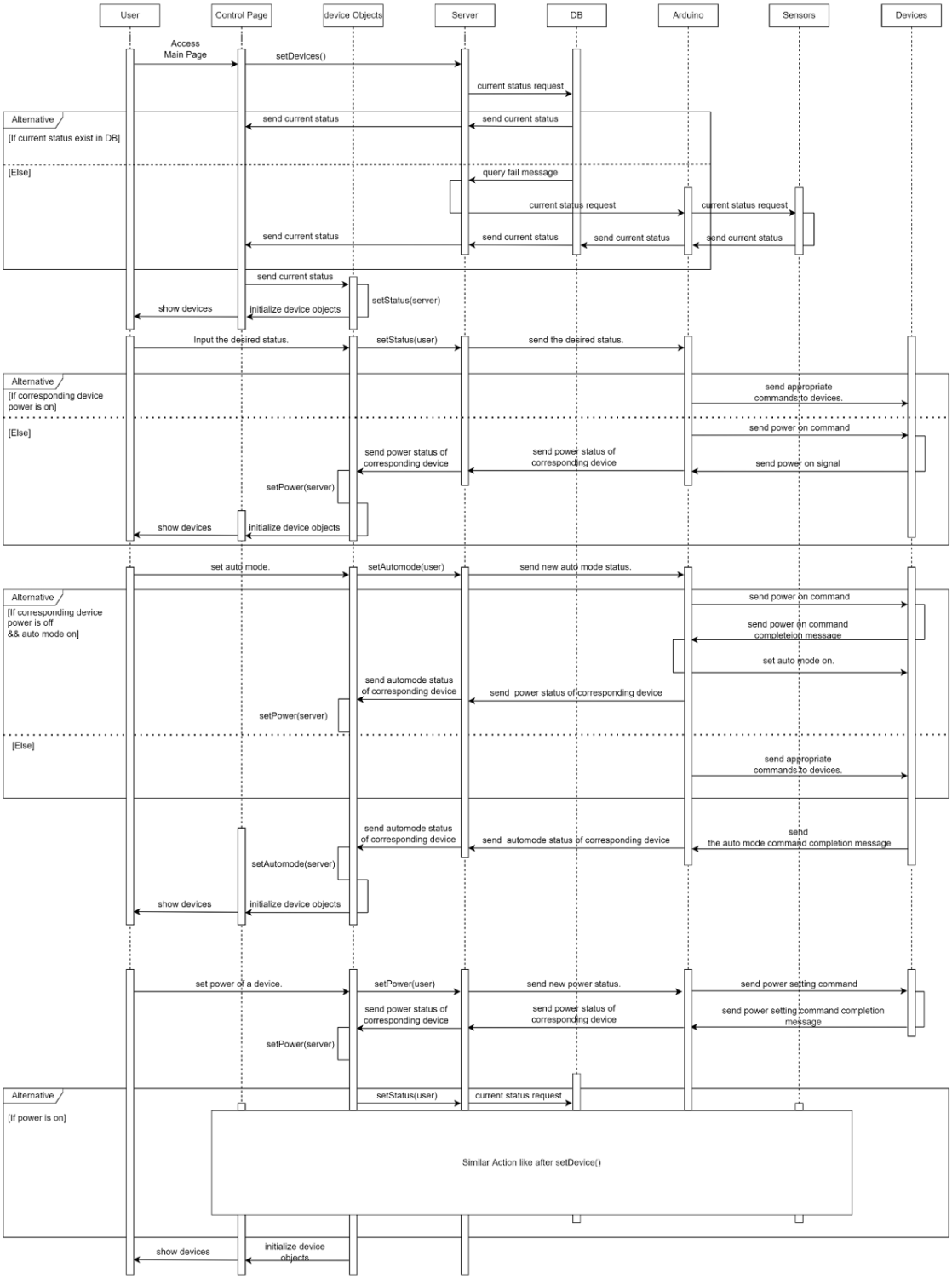
- setStatus(source): set fields currentState, automode, power. if the source is "user", send the message to the server, else (when source is "server") do not.
- setAutomode(source): set auto mode on/off if the source is "user", send the message to the server, else (when source is "server") do not.
- setPower(source): set power on/off. if the source is "user", send the message to the server, else (when source is "server") do not.

4.2.4.4 Class Diagram



[Figure 10] Control Page Class Diagram

4.2.4.5 Sequence Diagram



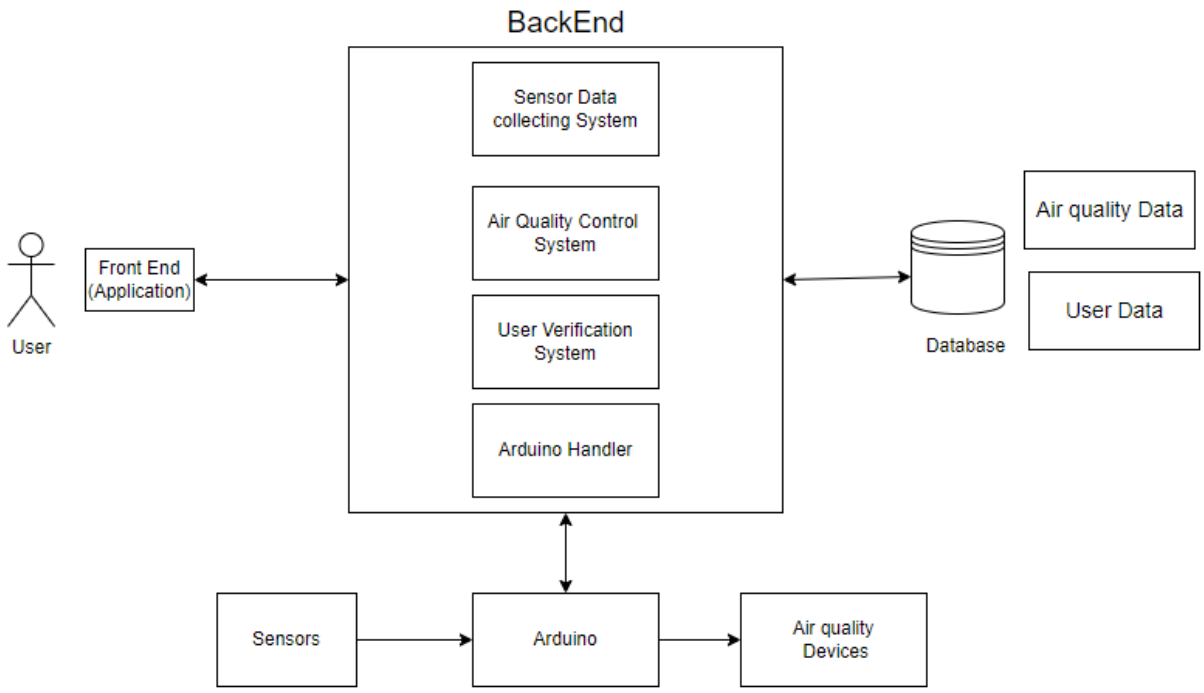
[Figure 11] Control Page Sequence Diagram

5. System Architecture - Backend

5.1 Objectives

The objective of this chapter is to describe structure of the backend

5.2 Overall Architecture

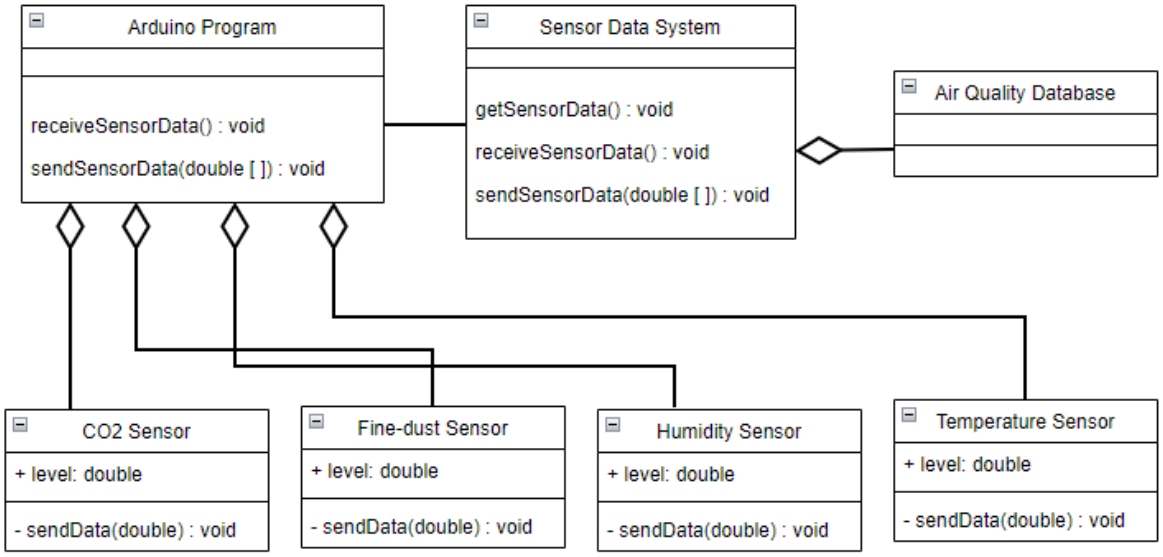


[Figure 12] System Architecture - Backend

5.3 Subcomponents

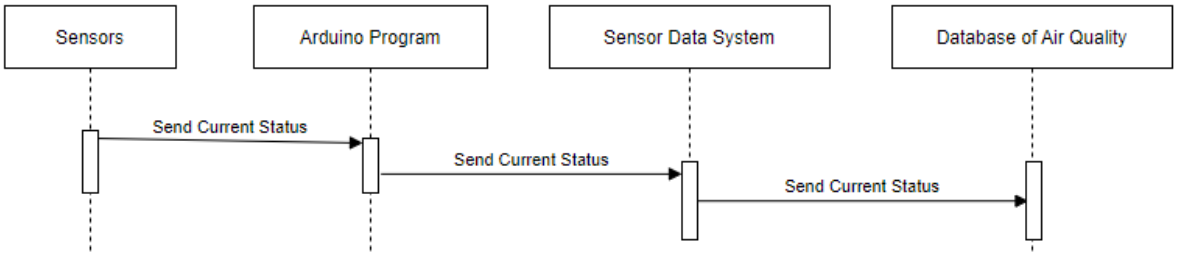
5.3.1 Sensor Data Storing System

5.3.1.1 Class Diagram



[Figure 13] Sensor Data Storing System Class Diagram

5.3.1.2 Sequence Diagram



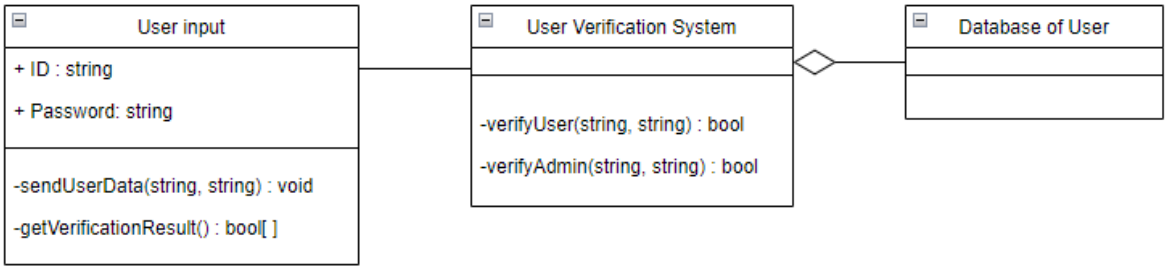
[Figure 14] Sensor Data Storing System Sequence Diagram

5.3.1.3 Description

BackEnd includes a system that can receive sensor data and store it in the air quality database. Sensor data will be transferred from each sensor to the arduino, and then to the sensor data system, and will eventually be stored in the air quality database for later use.

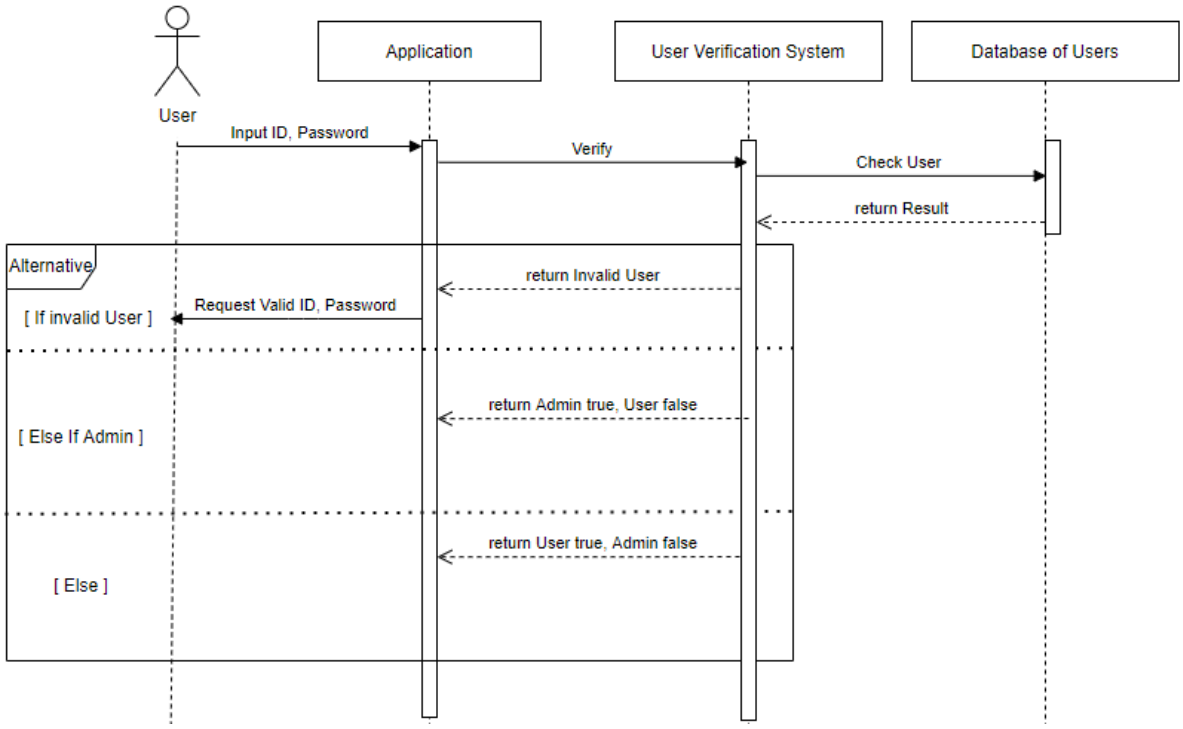
5.3.2 User Verification System

5.3.2.1 Class Diagram



[Figure 15] User Verification System Class Diagram

5.3.2.2 Sequence Diagram



[Figure 16] User Verification System Sequence Diagram

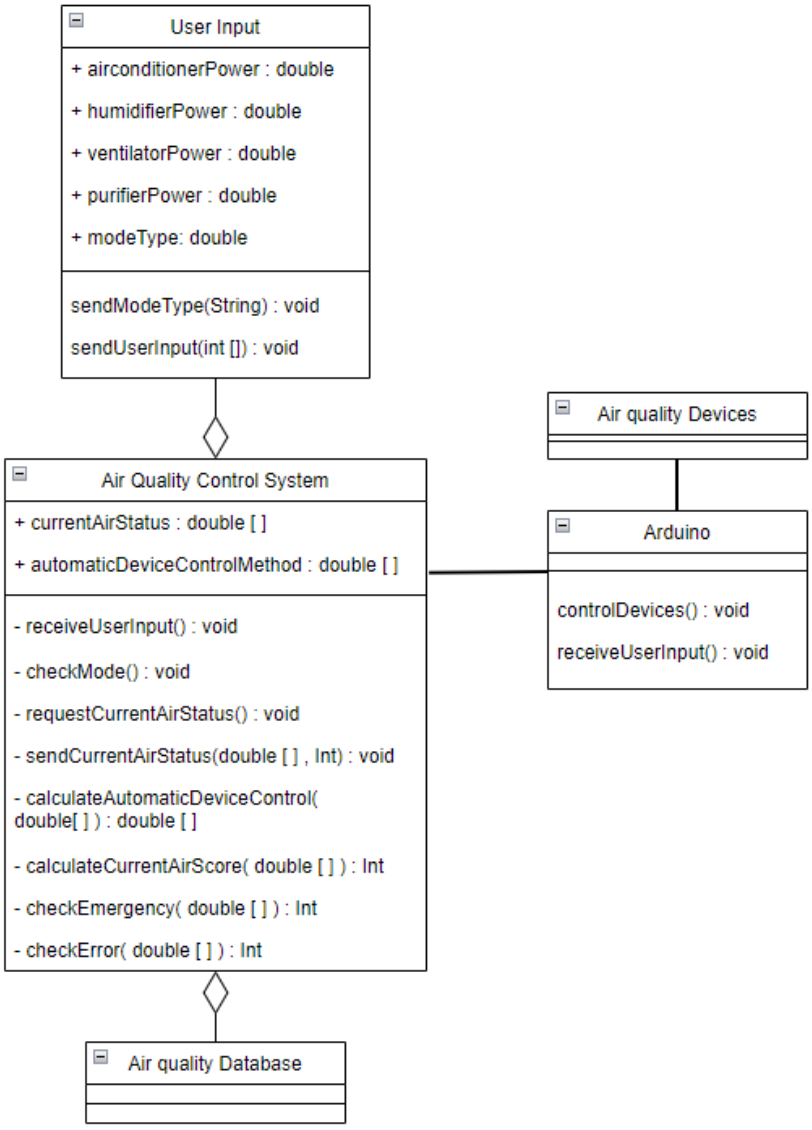
5.3.2.3 Description

Backend includes a system that can verify the user and admin. In order to do this, a system receives input data(ID, Password) from users and verifies it by referring to a database that

stores information about users. The system checks if the given ID and Password is from a valid user or admin, and requests valid ID and Password if it is invalid.

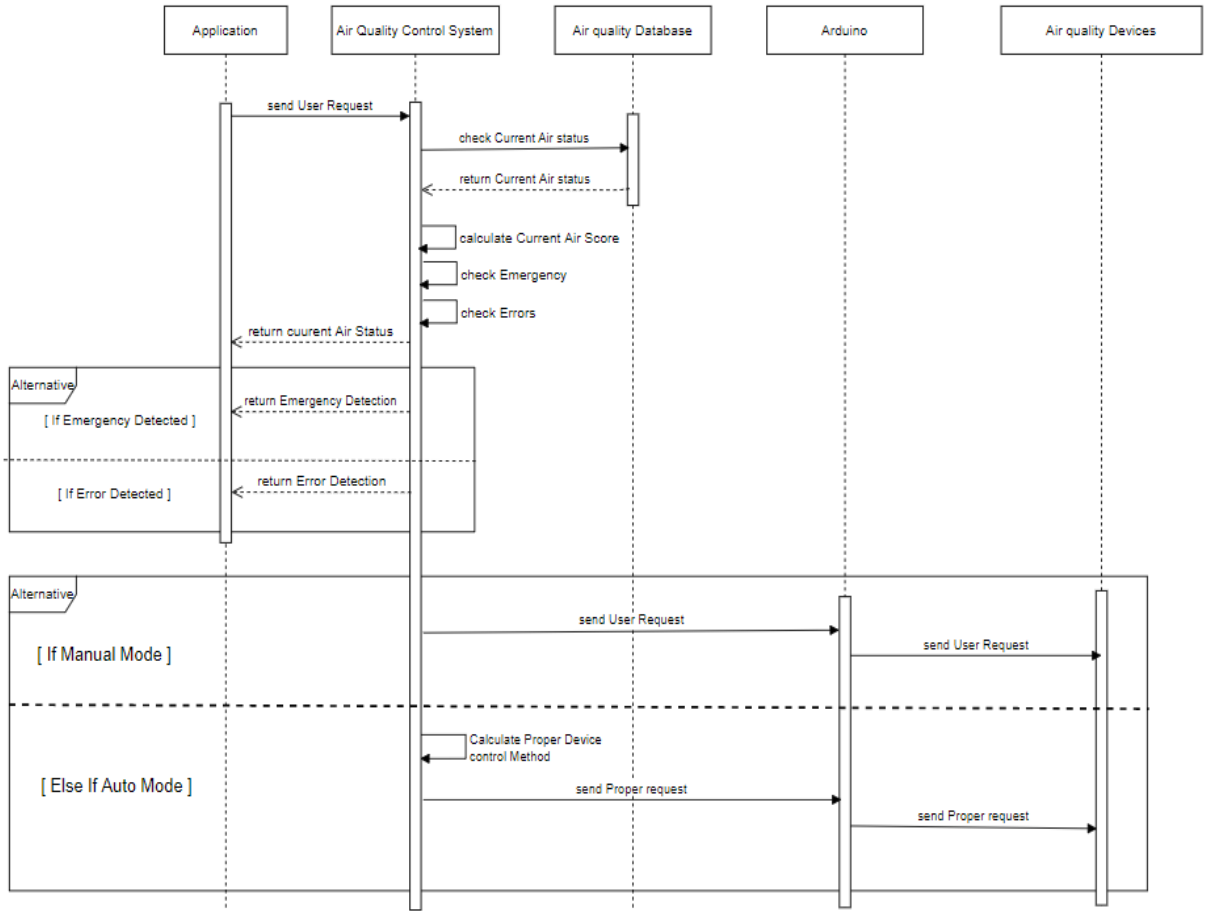
5.3.3 Air quality control system

5.3.3.1 Class Diagram



[Figure 17] Air quality control system Class Diagram

5.3.3.2 Sequence Diagram



[Figure 18] Air quality control system Sequence Diagram

5.3.3.3 Description

Backend includes a system related to air quality control. As the user opens and logins to the application, Backend checks the air quality database continuously in order to send current air status including current air score, and to check if there are emergencies or errors. If the User makes a request to operate air quality control devices using either manual mode or automatic mode, Backend receives requests and performs functions to send requests to Arduino. In case of manual mode, it sends requests directly to Arduino. However in the case of automatic

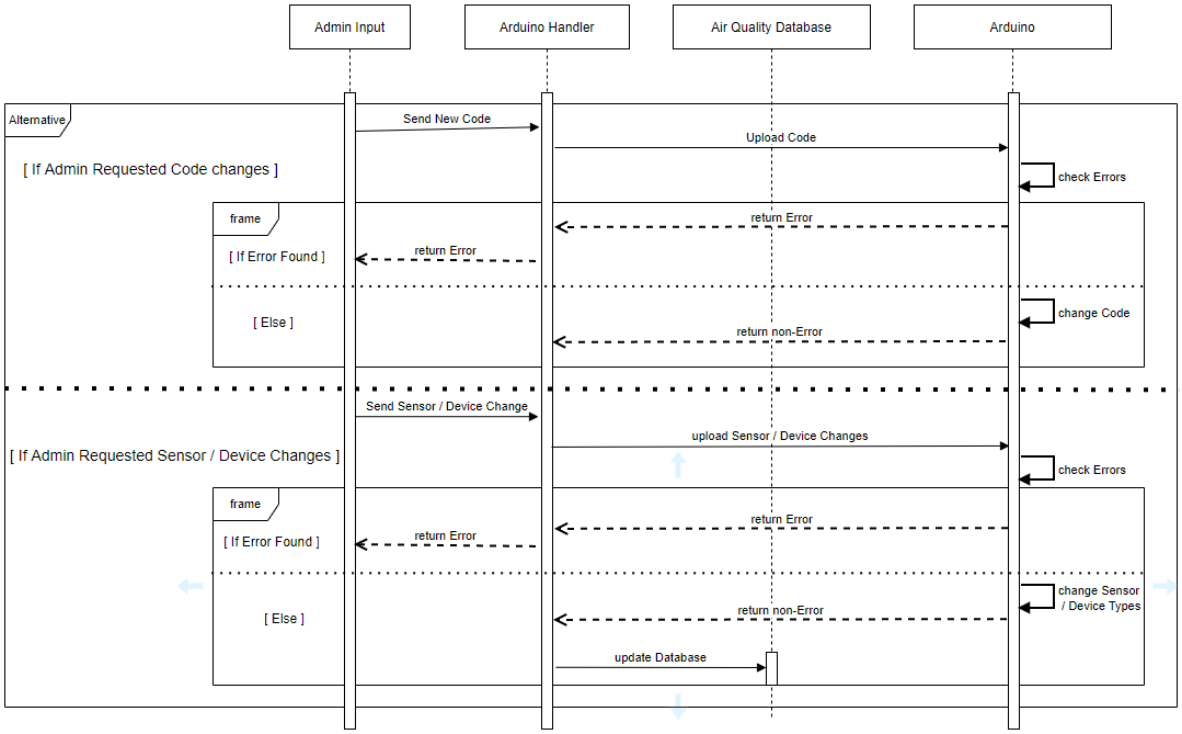
mode, it calculates the proper control method for each device by referring to current air status, and then sends a result to the Arduino.

5.3.4 Arduino Handler

5.3.4.1 Class Diagram

[Figure 19] Arduino Handler Class Diagram

5.3.4.2 Sequence Diagram



[Figure 20] Arduino Handler Sequence Diagram

5.3.4.3 Descriptions

Backend includes a system that controls and handles Arduino, called Arduino handler. There are two types of requests made from admin, which are code change request and sensor(device) change request. For each request, the Arduino handler receives input from the admin and sends the request to the Arduino. Then, the Arduino checks if there are errors to the request and sends the result back to the Arduino Handler. According to the result, if there is an error, the Arduino Handler either sends the result back to the Admin. If not, in case of sensor(device) changes, it updates the database that matches new sensor types.

6. Protocol Design

6.1. Objectives

This chapter explains the protocols that are used for interaction between each subsystem and describes the whole structure of those protocols.

It is important to note that communication is done mainly over HTTP, though the system is not a web-service in the traditional sense. HTTP is used since it is an easily readable protocol with a lot of support for modifications for one's own purposes. All HTTP responses and requests will be parsed using custom code on client, backend and Arduinos. Actions will be performed based on the method field which will trigger a function call on the respective host.

6.2. User-Backend Communication

6.2.1. User to Backend

6.2.1.1 Registration Attempt

[Table 1] Register request

Attribute	Detail	
Protocol	HTTP	
Request body	Name	User's name
	ID	User's ID
	Password	User's password
	Type	User type(0 : Admin, 1 : normal user)
	Phonenumber	User's phone number

	Address	User’s address

[Table 2] Register response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 404 (Not found)	
Success response body	Authorization number	Personal identification number
	Message	Success message
Failure response body	Message	Failure message

6.2.1.2. Log-in Attempt

[Table 3] Log-in request

Attribute	Detail	
Protocol	HTTP	
Request body	ID	User's ID
	Password	User's password
	Type	User type(0 : Admin, 1 : normal user)

[Table 4] Login response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request)
	HTTP 401 (Unauthorized)
	HTTP 404 (Not found)

Success response body	Access Token	Token for access
	Message	Success message
Failure response body	Message	Failure message

6.2.1.3. Get Data

[Table 5] Get admin data request

Attribute	Detail	
Protocol	HTTP	
Request body	Method	GetAdminData
	ID	User’s ID
	Cookie(Encrypted)	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted

[Table 6] Get admin data response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	SensorList	
	ProgramCode	
	UserList	
Failure response body	Message	Failure message

[Table 7] Get user data request

Attribute	Detail	
Protocol	HTTP	
Request body	Method	GetUserData
	ID	User's ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted

[Table 8] Get user data response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request)
	HTTP 401 (Unauthorized)
	HTTP 403 (Forbidden)

	HTTP 500 (Internal Server Error)	
Success response body	AirScore	The total air score of current air quality
	Temperature	Value of the current temperature
	Humidity	Value of the current humidity
	Finedust	Value of the current finedust
	UltraFinedust	Value of the current ultrafinedust
	DeviceList	List of available devices
Failure response body	Message	Failure message

6.2.1.4. Activate/deactivate device

[Table 9] Activate/Deactivate device request

Attribute	Detail	
Request body	Method	OnOffDevice
	ID	User's ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted
	DeviceID	ID of the device to activate of deactivate

[Table 10] Activate/Deactivate device response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request)
	HTTP 401 (Unauthorized)

	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	None	
Failure response body	Message	Failure message

6.2.1.5. Add/Delete Device/Sensor

[Table 11] Manage device request

Attribute	Detail	
Method	Method	ManageDevice
Request body	ID	User’s ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted
	DeviceID	ID of the device to delete
	NewDeviceObject	New device object to add

[Table 12] Manage device response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	Message	Success message
Failure response body	Message	Failure message

[Table 13] Manage sensor request

Attribute	Detail	
	Method	ManageSensor

Request body	ID	User’s ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted
	SensorID	ID of the sensor to delete
	NewSensorObject	New sensor object to add

[Table 14] Manage sensor response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 (Bad request)
	HTTP 401 (Unauthorized)

	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	Message	Success message
Failure response body	Message	Failure message

6.2.1.6. Load new Arduino Code

[Table 15] Load code request

Attribute	Detail	
Request body	Method	LoadCode
	ID	User's ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted
	NewCode	The new code file to upload

[Table 16] Load code response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	Message	Success message
Failure response body	Message	Failure message

6.2.1.7. Switch between auto and manual mode

[Table 17] Switch mode request

Attribute	Detail	
Request body	Method	SwitchMode

	ModeType	Mode type(0 : auto mode, 1 : manual mode)

[Table 18] Switch mode response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 500 (Internal Server Error)	
Success response body	None	
Failure response body	Message	Failure message

6.2.1.8. Adjust Values in manual mode

[Table 19] Adjust values in manual mode request

Attribute	Detail

Request body	Method	AdjustValues
	DeviceID	ID of the device that user want to adjust
	Value	Value that user want to adjust

[Table 20] Adjust values in manual mode response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 500 (Internal Server Error)	
Success response body	None	
Failure response body	Message	Failure message

6.2.1.9 Set Values for Auto Mode

[Table 21] Set Value for Auto Mode Request

Attribute	Detail	
Protocol	HTTP	
Method	SetAutoModeValue	
Request body	ID	User's ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted
	Attribute	The name of the attribute to be changed (e.g. Temperature)
	Value	New value for the specified attribute. (If attribute is temperature, this might be 23 C)

[Table 22] Set Value for Auto Mode Response

Attribute	Detail
Success Code	HTTP 200 OK

Failure Code	HTTP 403 (Forbidden)	The user does not have the appropriate privileges
	HTTP 500 (Internal Server Error)	The server could not handle the request
	HTTP 404 (Not found)	Parts of the request were not found (maybe unknown attribute name)
Success response body	Message	Success message
Failure response body	Message	Failure message

6.2.2. Backend to User

6.2.2.1 Emergency Notification

[Table 23] Emergency Notification Request

Attribute	Detail
Protocol	HTTP
Method	SendEmergency

Request body	EmergencyCode	Code number for different emergency events (Example: 0 -> Unknown Cause 1 -> Fire 2 -> Gas Leak …)
	Amount	Number of how many alarming value were measured
	Attribute1	Name of first attribute that has an alarming value
	Value1	Value of the first attribute that has an alarming value
	…	Continue Attribute/Value pairs until ‘Amount’ of pairs have been sent

[Table 24] Emergency Notification Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Server Error)	The client could not handle the request.

Success response body	Empty	The server only needs to know if the request failed or not. It expects no data in return.
Failure response body	Empty	

6.2.2.2 Send air quality data

[Table 25] Send Air Quality Request

Attribute	Detail	
Protocol	HTTP	
Method	SendAirQualityData	
Request body	Method	
	Amount	Number of how many different data value will be transmitted
	Attribute1	Name of the first transmitted attribute
	Value1	Value of the first transmitted attribute

	…	Continue Attribute/Value pairs until ‘Amount’ of pairs have been sent

[Table 26] Send Air Quality Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Server Error)	The client could not handle the request.
	HTTP 404 (Not Found)	If a to the client unknown attribute name was sent
Success response body	Empty	The server only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	if 404: Attributes	Send back the names of all available attribute names.
	if 500: Empty	The server cannot do anything about an internal client error and therefore expects no data.

6.2.2.3 Send various errors

[Table 27] Send Error Request

Attribute	Detail	
Protocol	HTTP	
Method	SendErr	
Request body	Error Code	A number indicating a well defined error. (e.g. 1 -> Device unreachable 2 -> Arduino Code Error …)
	ParameterList	A list of necessary parameters to handle the specified error. (e.g. if a device got disconnected, send device name and number)

[Table 28] Send Error Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Sever Error)	The client could not handle the request.

	HTTP 404 (Not Found)	If a to the client unknown error code was sent.
Success response body	Empty	The server only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	if 404: ErrorCode	Send back the error code unknown to client.
	if 500: Empty	The server cannot do anything about an internal client error and therefore expects no data.

6.3. Backend-Database Communication

Backend-Database communication will abide by the rules defined byFirebase. No low-level protocol needs to be defined for this interaction. As long as the backend follows the Firebase API for querying and inserting, communication will succeed.

6.4. Backend-Arduino Communication

6.4.1. Backend to Arduino

6.4.1.1 Upload new code

[Table 29] Upload New Code Request

Attribute	Detail

Protocol	HTTP	
Method	NewCode	
Request body	Code	The code supposed to run on the Arduino

[Table 30] Upload New Code Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Server Error)	The Arduino could not handle the request.
Success response body	Empty	The server only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	Empty	The server cannot do anything about an internal Arduino error and therefore expects no data.

6.4.1.2 Send command to devices

[Table 31] Send Command Request

Attribute	Detail	
Protocol	HTTP	
Method	SendCommand	
Request body	DeviceID	Specify id of the device that the command is sent to.
	Command	A number that maps to a specific command(e.g. 0 -> turn off 1 -> turn on …)
	CommandParameters	List of parameters to perform the command.

[Table 32] Send Command Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Server Error)	The Arduino could not handle the request.

	HTTP 404 (Not Found)	No device with specified id or unknown command.
Success response body	Empty	The server only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	if 500: Empty	The server cannot do anything about an internal Arduino error and therefore expects no data.
	if 404: List of valid values	Return a list of available device ids and available commands for each device.

6.4.2. Arduino to Backend

6.4.2.1 Send Sensor Data

[Table 33] Send Sensor Data Request

Attribute	Detail	
Protocol	HTTP	
Method	SendData	
Request body	SensorID	Specify id of sensor from which the data is sent from.

	Attribute	Name of attribute being sent
	Value	Value of attribute

[Table 34] Send Sensor Data Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 404 (Not Found)	Sensor might not be registered properly on the server.
	HTTP 500 (Internal Server Error)	The server could not handle the request
Success Response Body	Empty	The Arduino only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	if 500: Empty	The Arduino cannot do anything about an internal server error and therefore expects no data.
	if 404: List of Sensors	Respond with list of all registered sensors.

6.4.2.2 Send Error

[Table 35] Send Error Request

Attribute	Detail	
Protocol	HTTP	
Method	SendErr	
Request body	Error Code	A number indicating a well defined error. (e.g. 1 -> Device unreachable 2 -> Code Error …)
	ParameterList	A list of necessary parameters to handle the specified error. (e.g. if a device got disconnected, send device name and number)

[Table 36] Send Error Response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 500 (Internal Server Error)	The server could not handle the request.

	HTTP 404 (Not Found)	If a to the server unknown error code was sent.
Success response body	Empty	The Arduino only needs to know if the request succeeded. It expects no data in return.
Failure Response Body	if 404: ErrorCode	Send back the error code unknown to server.
	if 500: Empty	The Arduino cannot do anything about an internal client error and therefore expects no data.

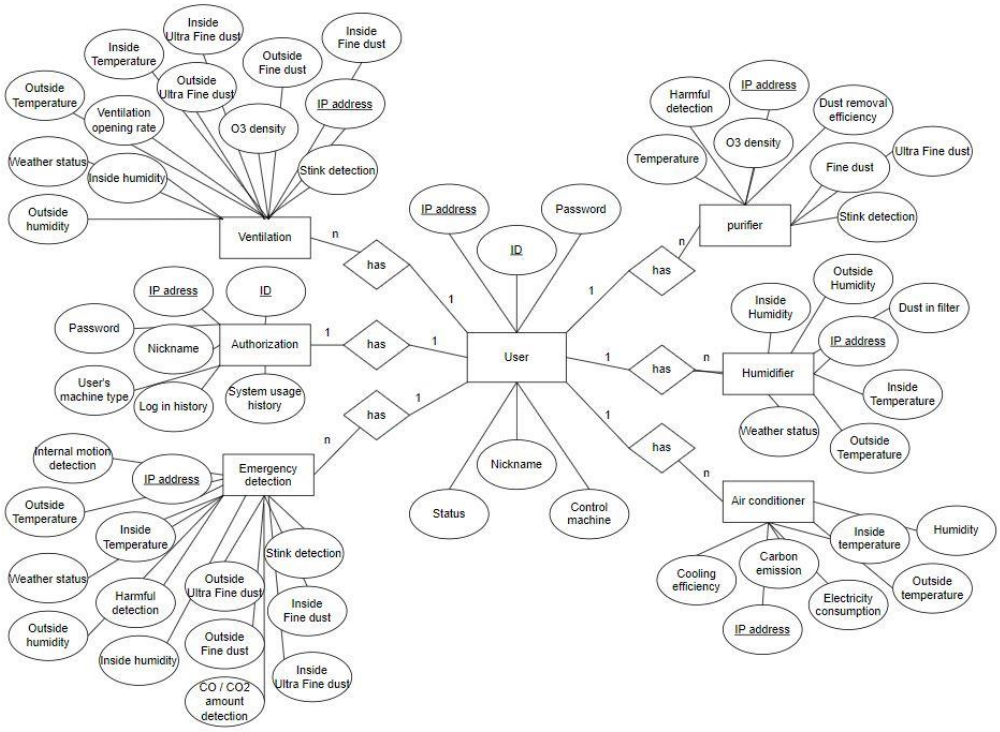
7. Database Design

7.1 Objectives

This section describes the system data structures and how these are to be represented in a database. This part describes entities first, and their relationship through ER-diagram (Entity Relationship diagram).

7.2 ER Diagram

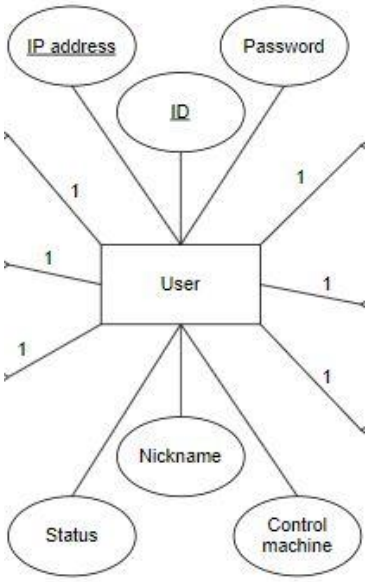
The system consists of seven entities: User, Purifier, Air conditioner, Humidifier, Ventilation, Emergency detection, Authorization. ER-diagram expresses each entity as rectangular and their relationship as a rhombus. The primary key(unique attribute) which uniquely identifies an entity is underlined. There are also entities with multiple primary keys.



[Figure 21] Database ER-Diagram

7.2.1 Entities

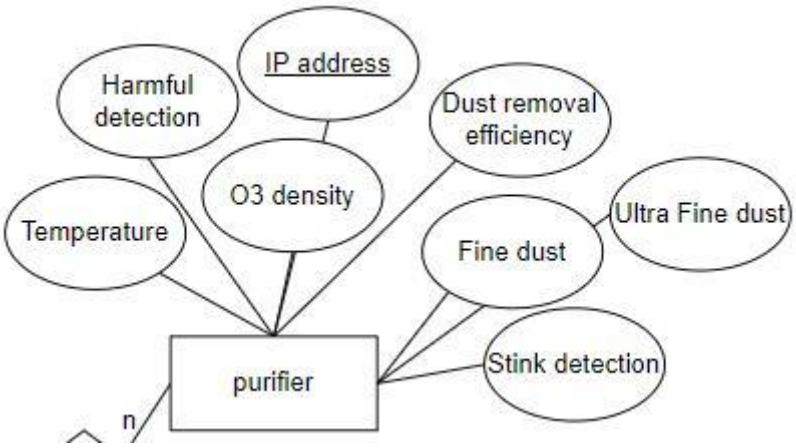
7.2.1.1 User



[Figure 22] ER diagram, Entity, User

User entity represents a user using Sensor-Base Indoor Air Purification System. It contains ID, IP address, Password, Status, Nickname, Control machine. ID and IP address are the primary key.

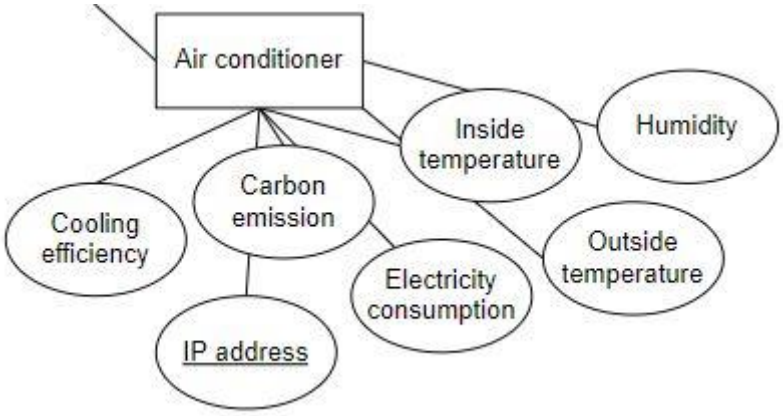
7.2.1.2 Purifier



[Figure 23] ER diagram, Entity, Purifier

Purifier entity is the object which is one of air-purifying devices. It contains IP address, Fine dust, Ultra fine dust, Temperature, Stink detection, Harmful detection, Dust removal efficiency, O3 density. IP address is the primary key.

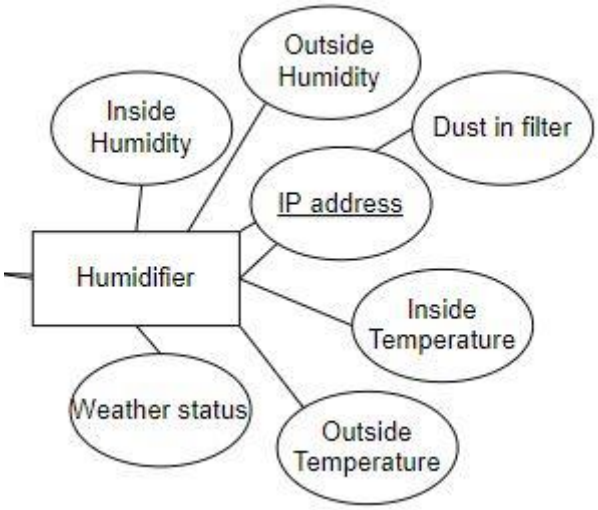
7.2.1.3 Air conditioner



[Figure 24] ER diagram, Entity, Air conditioner

Air conditioner entity is the object which is one of air-purifying devices. It contains IP address, Inside temperature, Outside temperature, Cooling efficiency, Carbon emission, Electricity consumption. IP address is the primary key.

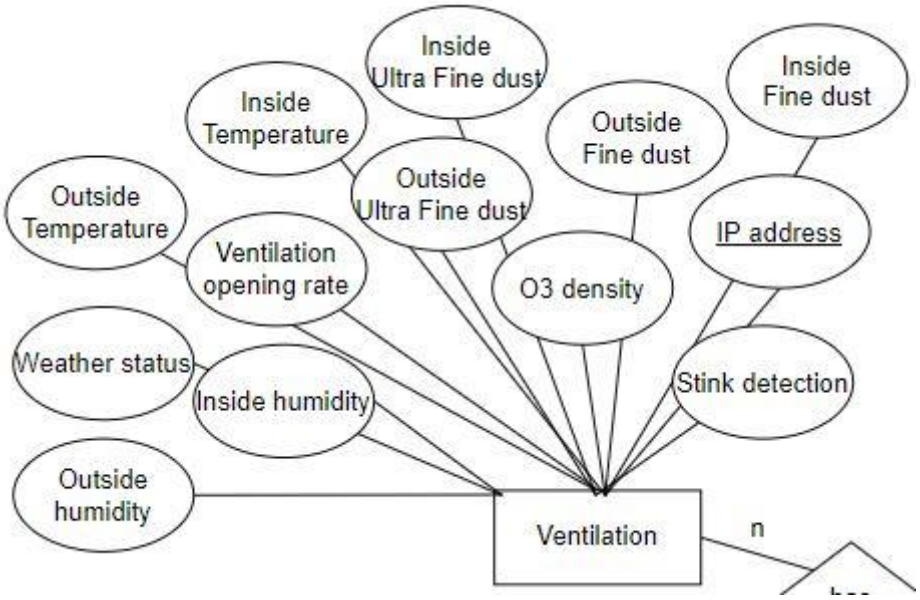
7.2.1.4 Humidifier



[Figure 25] ER diagram, Entity, Humidifier

Humidifier entity is the object which is one of air-purifying devices. It contains IP address, Dust in filter, Inside temperature, Outside temperature, Weather status, Inside humidity, Outside humidity. IP address is the primary key.

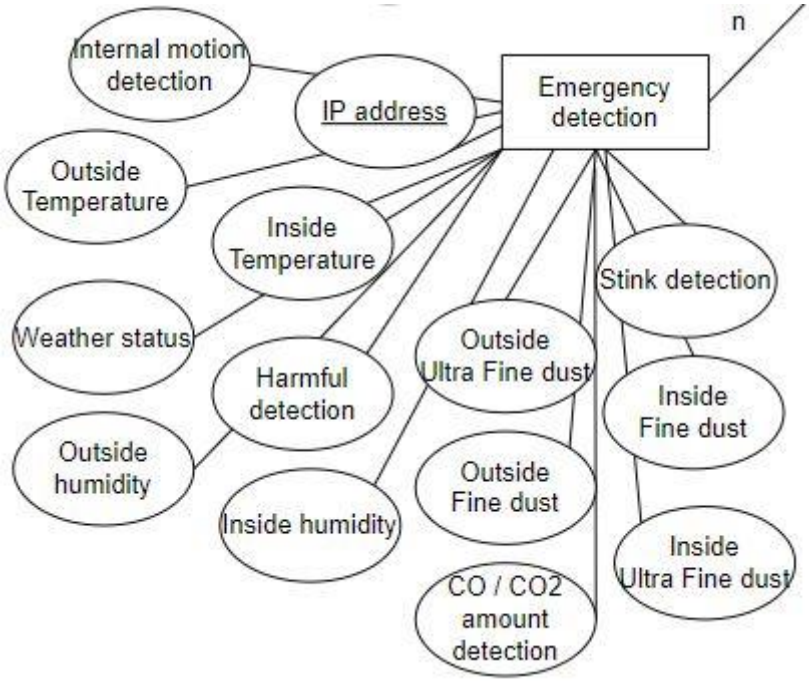
7.2.1.5 Ventilation



[Figure 26] ER diagram, Entity, Ventilation

Ventilation entity is the object which is one of air-purifying devices. It contains IP address, Stink detection, Inside Fine dust, Outside Fine dust, Inside Ultra fine dust, Outside Ultra fine dust, Inside temperature, Outside temperature,Ventilation opening rate, Weather status, Inside humidity, Outside humidity. IP address is the primary key.

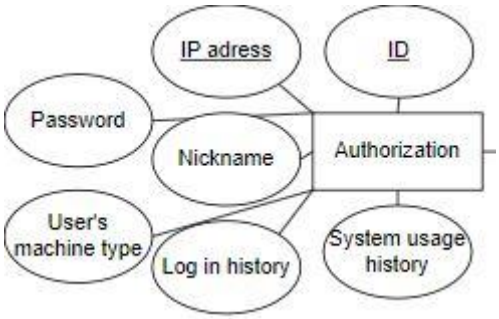
7.2.1.6 Emergency detection



[Figure 27] ER diagram, Entity, Emergency detection

Emergency detection entity is the object that detects air status by sensors and informs user of the emergency. It contains IP address, Stink detection, Inside Fine dust, Outside Fine dust, Inside Ultra fine dust, Outside Ultra fine dust, Inside temperature, Outside temperature, Harmful detection, Weather status, Inside humidity, Outside humidity, Internal motion detection, CO / CO2 amount detection. IP address is the primary key.

7.2.1.7 Authorization



[Figure 28] ER diagram, Entity, Authorization

Authorization entity is the object which saves the histories of a system. Authorized user only can log in. It contains ID, IP address, Password, Status, Nickname, System usage history, User's machine type, Log in history. ID and IP address are the primary key.

8. Testing plan

8.1 Objectives

This phase will explain plans for development testing, release testing, and user testing. These tests play an important role in increasing program completeness and enabling stable deployment by detecting program errors and supplementing flaws. It also helps determine the effort required to verify the quality of the application being tested when testing our project. Therefore, following guides the way of thinking our project is planning.

8.2 Testing policy

8.2.1 development testing

Development testing is a test that takes place in the development phase, and the main goal is to stabilize the program. At this stage, it presents policies for the modules to test whether developed well or not satisfying the content of the design specification implemented in code. The program is stabilized through static code analysis, data flow analysis, peer code review, unit testing. In this process, we will focus on three main factors. First is to evaluate how efficiently the program achieves what it does in terms of performance. Secondly, we will

evaluate how stable the program works in terms of Reliability. The third is to evaluate Security.

8.2.1.1 Performance

Since we need to prepare for an emergency, the time to manipulate the values of each device and the time to receive air quality from each sensor should not be long. Therefore, the time for the device to respond to the user's operation and the time for real-time air quality information to be updated is limited to 1 second.

8.2.1.2 Reliability

In order for the system to function properly Accurate current air quality conditions must be stored. The subcomponents and units that make up the system must first operate and connect correctly. Therefore, we need to go through development tests from the unit development stage and iteratively integrate to the system checking failure iteratively and very carefully.

8.2.1.3 Security

Securing information of users and the system is a crucial matter to be handled by developers. Regardless of the value of the information, it should be protected from unwanted visitors to the system. In this system, We must not allow other users to tamper with our device or obtain status information.

8.2.2 Release Testing

Release testing is the process that involves testing a particular release of a system. In release testing, each release undergoes a series of automated and manual tests to ensure the quality of the finished product. The objective of release testing is to check that the system meets its requirements and is good enough for external use (validation testing). After release testing,

we would fix the weakness of the old version by accepting responses of users and release the new version.

8.2.3 User Testing

User Testing is the stage where testing is conducted with real users and receiving opinions and advice from them. We should set up possible scenarios and realistic situations that can proceed with necessary user tests. Sensor-Base Indoor Air Purification System will collect opinions from 20 users.

8.2.4 Testing case

Test cases are set up based on three basic aspects: performance, reliability, and security. In this section, we check whether it is functionally workingTest cases should be built to produce the desired coverage of the software being tested. After setting this situation, we would distribute the Beta version to them and collect user reviews while carrying out our use cases test.

9. Development Plan

9.1 Objectives

This chapter illustrates the development environment and tools we used during development. The limitations and assumptions considered during development will also be described.

9.2 Frontend Environment

9.2.1 Android Studio

[Figure 29] Android Studio logo

It is the official integrated development environment for Google's Android operating system, and it is the most fundamental environment for developing the Sensor-Base Indoor Air Purification System application.It is an integrated development environment created by Google based on IntelliJ IDEA of JetBrains for Android app development. It replaced Eclipse and became the official IDE for Android. It is provided for free and supports Windows, macOS, and Linux. In this project it is used to build a structure of the application by setting visible actions to be followed according to user interaction in the application using XML layouts.

9.2.2 Flutter

[Figure 30] Flutter logo

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. It is used to make user interfaces in this project.

9.2.3 Adobe Photoshop

[Figure 31] Adobe Photoshop logo

It is a graphic software developed and published by Adobe Systems, it enables media editing and animation editing work by sharing files with other Adobe software. In this project, It provides appropriate layouts and icons.

9.2.4 Adobe Xd

[Figure 32] Adobe Xd logo

Adobe XD is a vector-based user experience design tool for web apps and mobile apps, developed and published by Adobe Inc. It is available for macOS and Windows, and there are versions for iOS and Android to help preview the result of work directly on mobile devices. We can use various layouts, grids, and icons, and various functions in this project.

9.3 Backend Environment

9.3.1 Firebase

[Figure 33] Firebase logo

It supports development of mobile and web applications by providing various features such as cloud storage, real-time database, machine learning kit, etc. We would use a real-time database system provided by Firebase. With this, we can store lots of air quality information in real-time. And also updated air quality information will be informed ro uses in real-time.

9.3.2 Arduino C++

[Figure 34] C++ logo

Arduino is an open source for designing and manufacturing single-board microcontroller and microcontroller kits. It is created for the purpose of making circuits with boards and parts and allowing anyone to easily access IT through the provided code environment. The Arduino board circuit diagram and code environment are provided as open sources. It is used to program codes for the Arduino Board in this project.

9.3.3 Github

[Figure 35] Github logo

It provides hosting for software development version control using Git. It enables teammates to develop a single project together, resulting in easy integration of components. We are now using GitHub for developing the Sensor-Base Indoor Air Purification System application and controlling versions of it.

9.4 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.

9.5 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.

10. Supporting Information

10.1. 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)

10.2. Document History

[Table 37] Document History

Date	Version	Description	writer

2021/05/08	0.1	Overall Style form	Sungjoon Lee
2021/05/08	1.0	Addition of 1	All
2021/05/08	1.1	Addition of 2	Joonsun Baek
2021/05/08	1.2	Addition of 3.1	Maike Helbig
2021/05/08	1.3	Addition of 3.2	Hangyu Kim
2021/05/08	1.4	Addition of 3.2.2	Maike Helbig
2021/05/08	1.5	Addition of 3.3	Seyeon Park
2021/05/09	1.6	Revision of 2.1~2.2	Maike Helbig
2021/05/09	1.7	Revision of 2.4~2.5	Seyeon Park
2021/05/09	1.8	Addition of 4.1	Joonsun Baek
2021/05/09	1.9	Addition of 4.2	Hangyu Kim
2021/05/10	1.10	Addition of 5.1~5.3	Jeongmin Cha
2021/05/11	1.11	Addition of 6.1~6.4	Jinkyu Lee
2021/05/11	1.12	Revision of 6.2~6.4	Joonsun Baek
2021/05/11	1.13	Revision of 3.2	Hangyu Kim
2021/05/11	1.14	Addition of 7.1~7.2	Jeongmin Cha
2021/05/11	1.15	Addition of 8.1~8.2	Jinkyu Lee
2021/05/12	1.16	Addition of 9~9.5	All
2021/05/12	1.17	Addition of 10	Hangyu Kim

2021/05/13	1.18	Revision of 8.1	Joonsun Baek
2021/05/13	1.19	Revision of 9.3~9.5	Seyeon Park
2021/05/14	1.20	Overall revision	All