Sensor based Indoor Air Purification System

Jeongmin Cha
Hangyu Kim
Joonsun Back
Seyeon Park
Sungjoon Lee
Jingyu Lee
Maike Helbig

Agenda

1. Background
2. Purpose
3. SRS(Requirement specification)
4. SDS(Design specification)

Background

Purpose

SRS Overall

1. System interface

2. User interface

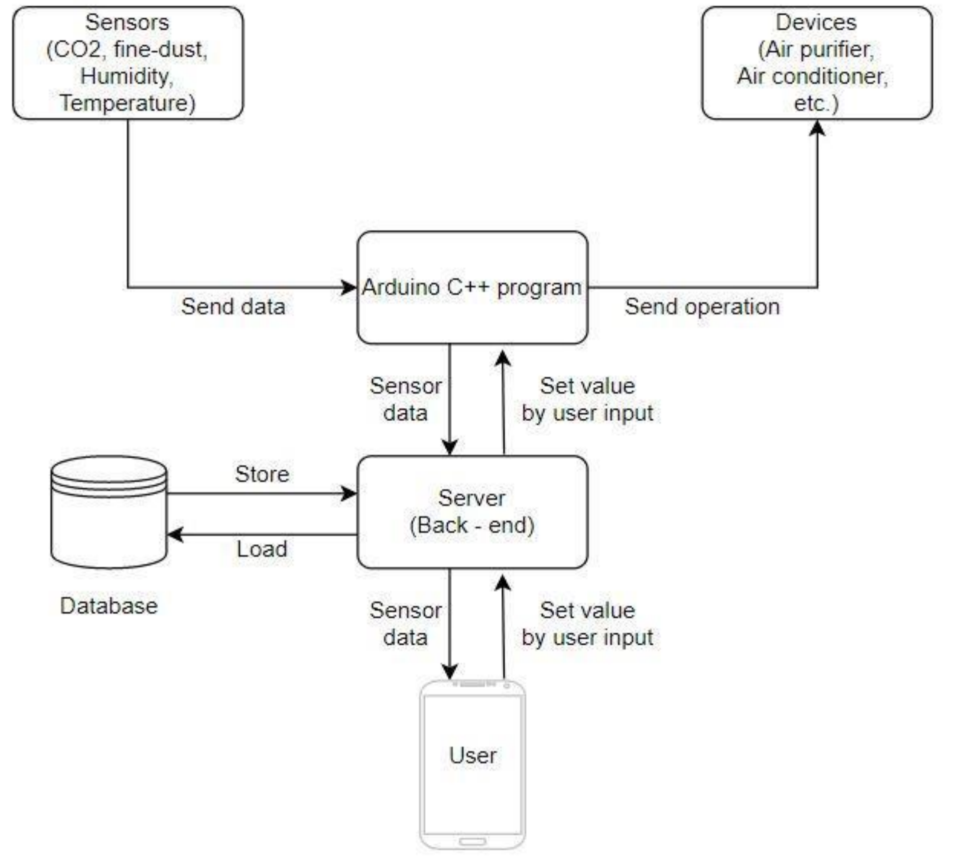
3. Hardware interface

4. Communications Interface

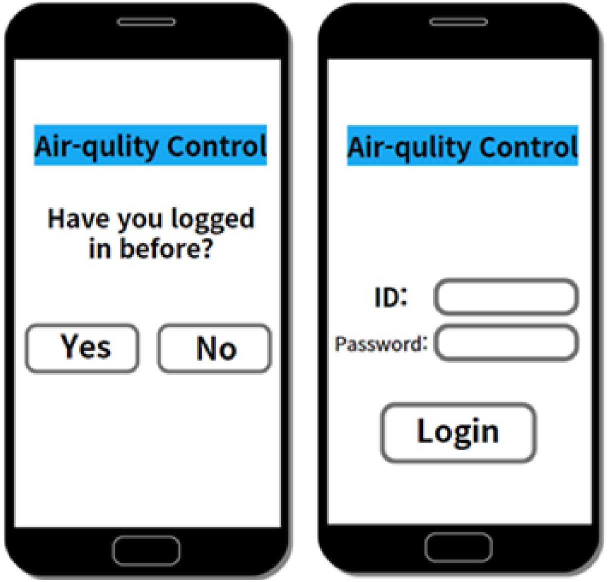
5. Memory Constraints

6. Operations

Overall System architecture

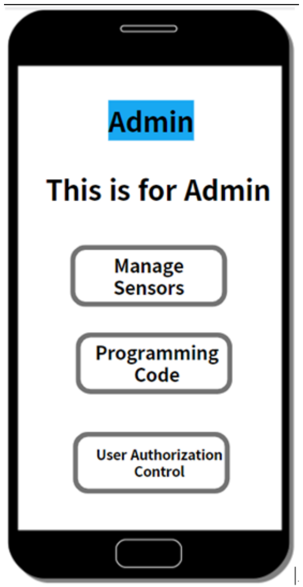


User Interface



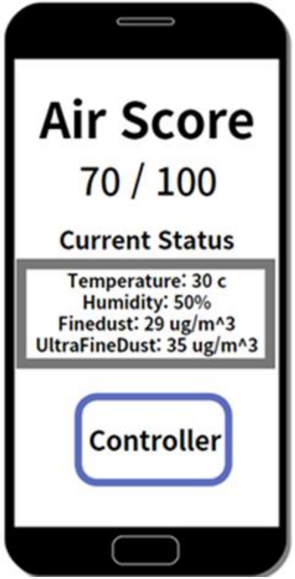
Have you logged in before?

[Login Page]



This is for Admin

[Admin Page]

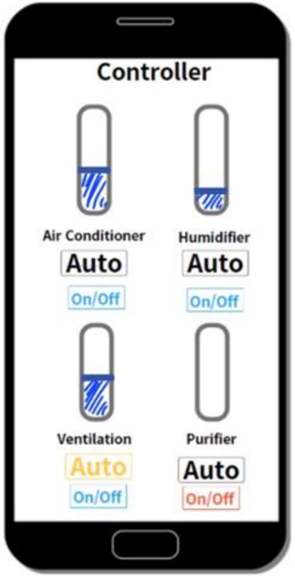


70 / 100

Current Status

Temperature: 30 c
Humidity: 50%
Finedust: 29 ug/m^3
UltraFineDust: 35 ug/m^3

[Main Page]



[Control Page]

Hardware Interface

Smart Phone – Android 5.0 , IOS 6.16

Computer – X86 ,

Air controlling Devices – WiFi, Aduino

Air monitoring Sensors – WiFi, Aduino

Arduino Board – Ver. Nano R3, WiFi, DB

Software Interfaces

Android
Studio

Communication Interface

User - Air Controlling Devices

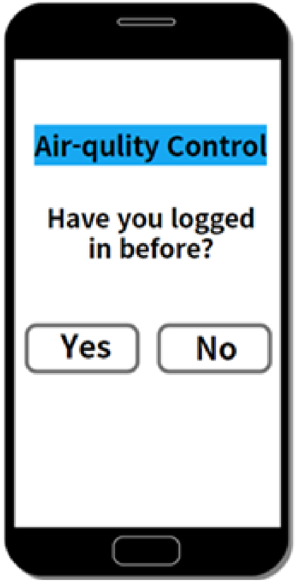
User - Backend

Backend - Arduino Board

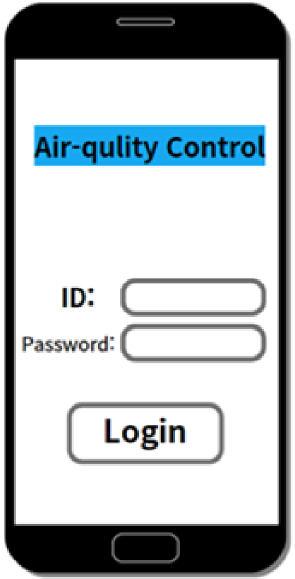
Arduino - (Sensors + Air controlling Devices)

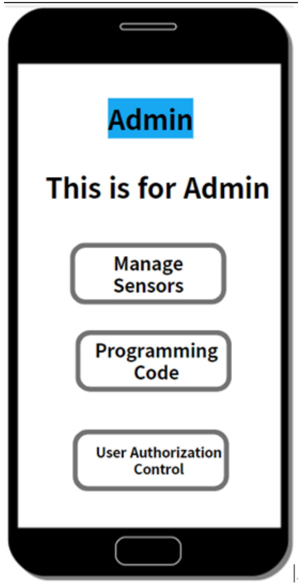
Functional Requirements

- Use Case

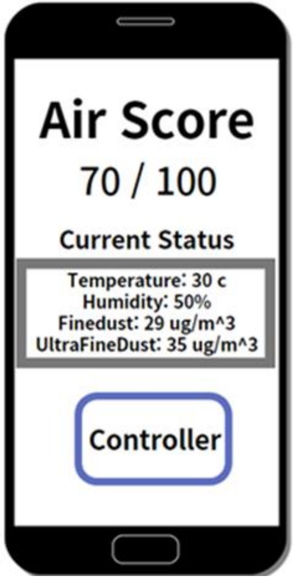

Air-qulity Control

Have you logged in before?


Air-qulity Control


Admin

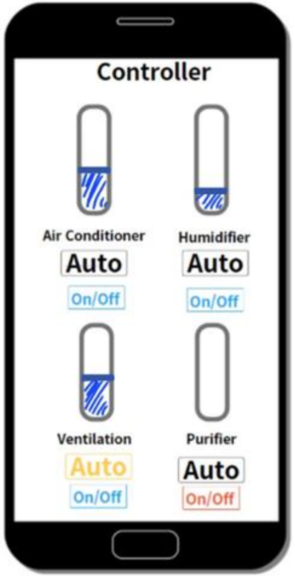
This is for Admin


Air Score

70 / 100

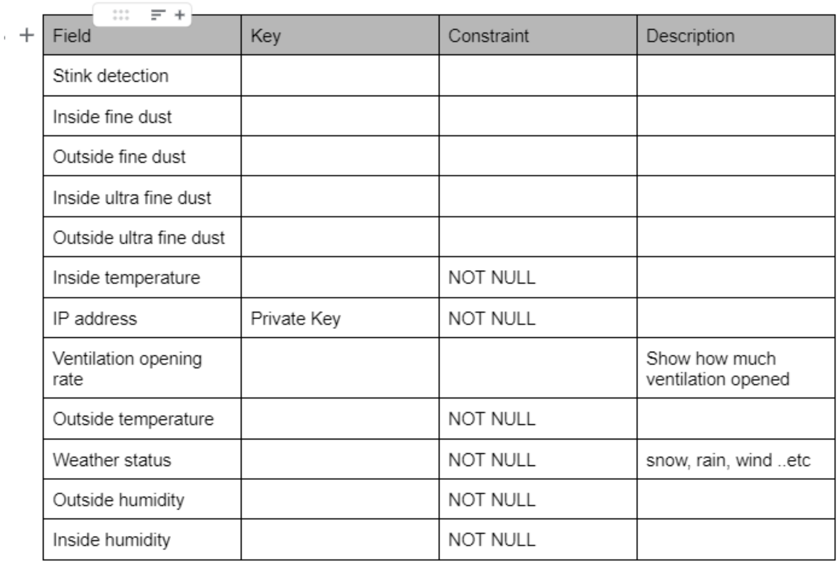
Current Status

Temperature: 30 c
Humidity: 50%
Finedust: 29 ug/m^3
UltraFineDust: 35 ug/m^3


Controller

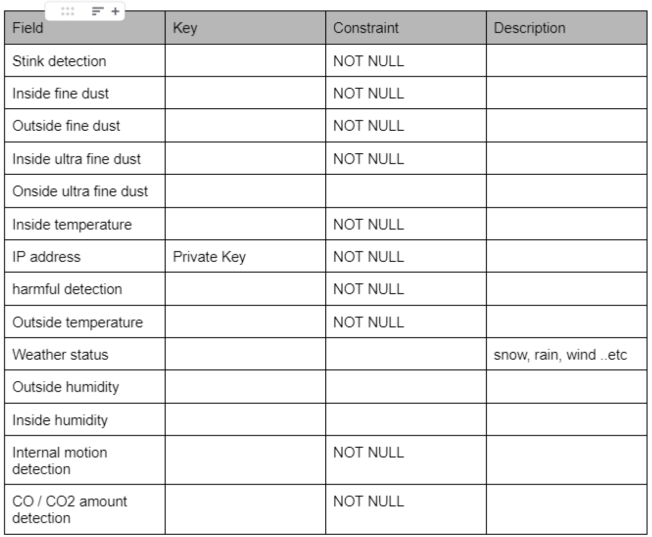
Functional Requirements

[Table 27] Table of Data Dictionary - Ventilation



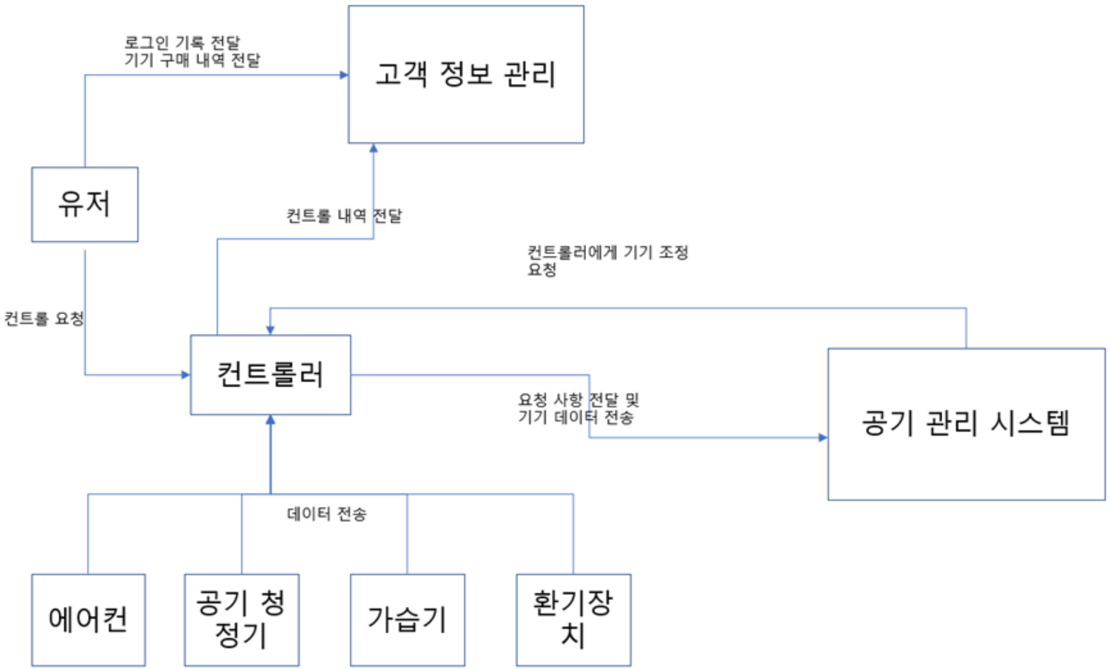
Field	Key	Constraint	Description
Stink detection			
Inside fine dust			
Outside fine dust			
Inside ultra fine dust			
Outside ultra fine dust			
Inside temperature		NOT NULL	
IP address	Private Key	NOT NULL	
Ventilation opening rate			Show how much ventilation opened
Outside temperature		NOT NULL	
Weather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

[Table 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	

Data Flow Digaram



고객 정보 관리

로그인 기록 전달
기기 구매 내역 전달

유저

컨트롤 내역 전달

컨트롤러에게 기기 조정
요청

컨트롤 요청

컨트롤러

요청 사항 전달 및
기기 데이터 전송

공기 관리 시스템

데이터 전송

에어컨

공기 청
정기

가습기

환기장
치

Nonfunctional Requirements

Product Requirements

Organizational Requirements

External Requirements

Product Requirements

- Usability
- Dependability
- Efficiency
- Security

Organizational Requirements

- Development Requirements
- Environmental Requirements
- Operational Requirements

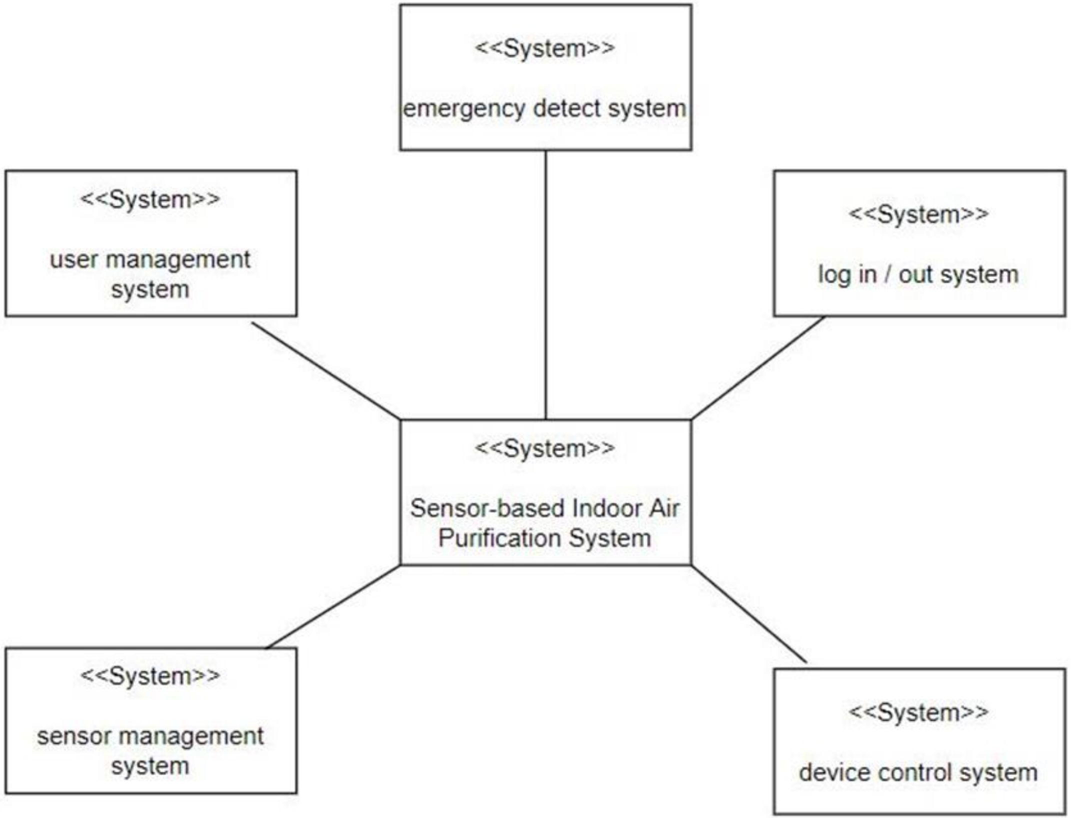
External Requirements

- Safety requirement

- Privacy Requirements

System Architecture - Overall

Context Diagram



<<System>>

emergency detect system

<<System>>

user management system

<<System>>

log in / out system

<<System>>

Sensor-based Indoor Air Purification System

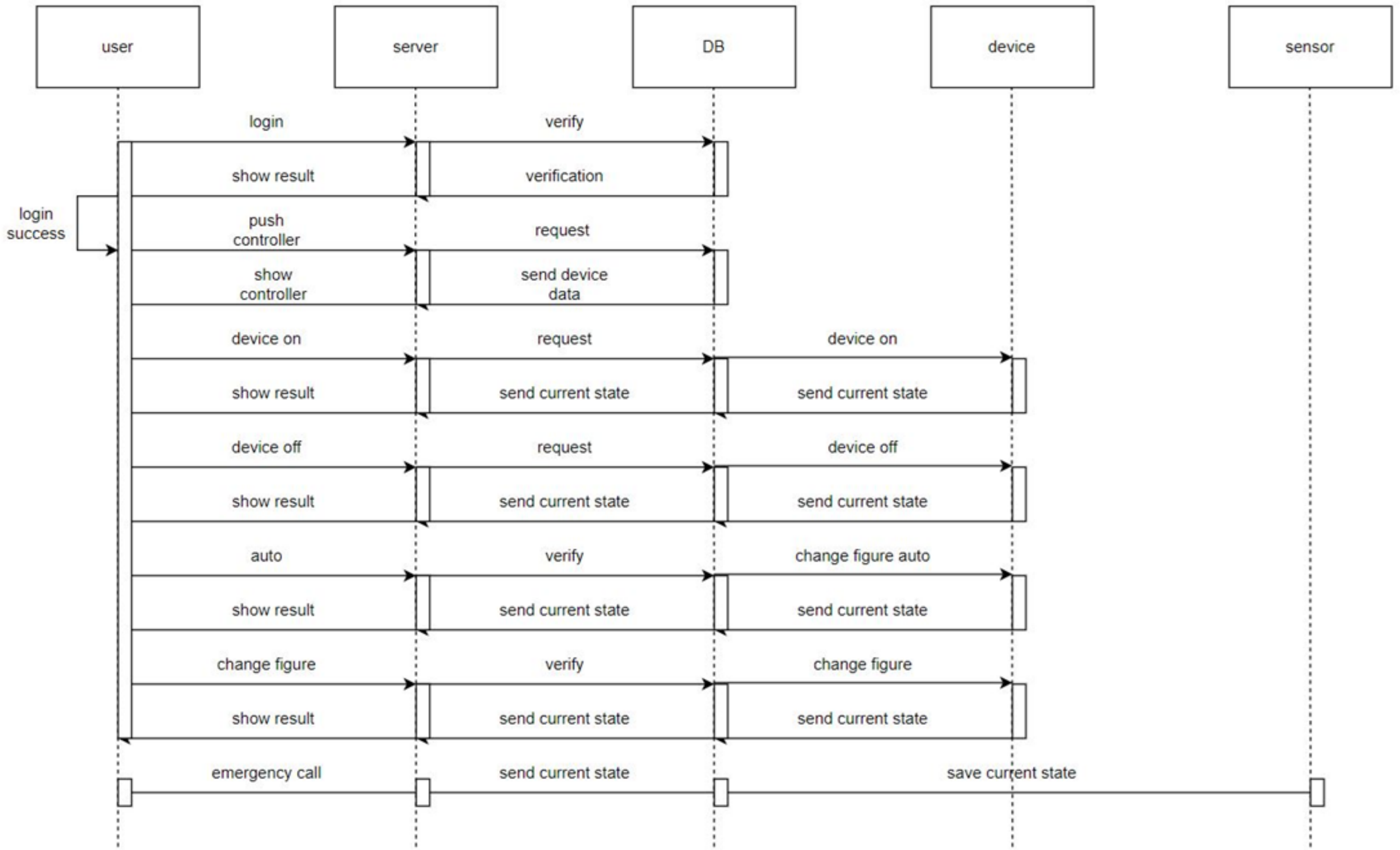
<<System>>

sensor management system

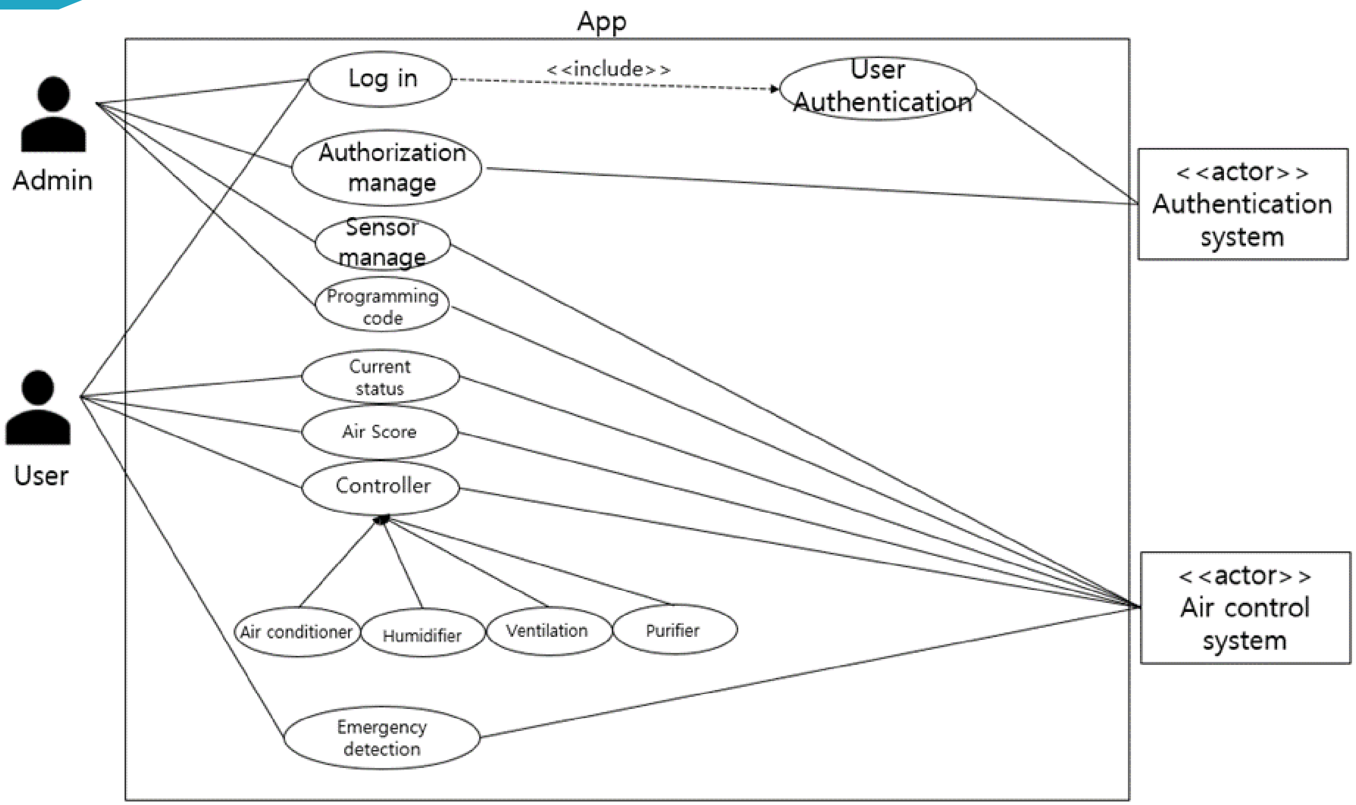
<<System>>

device control system

Sequence Diagram

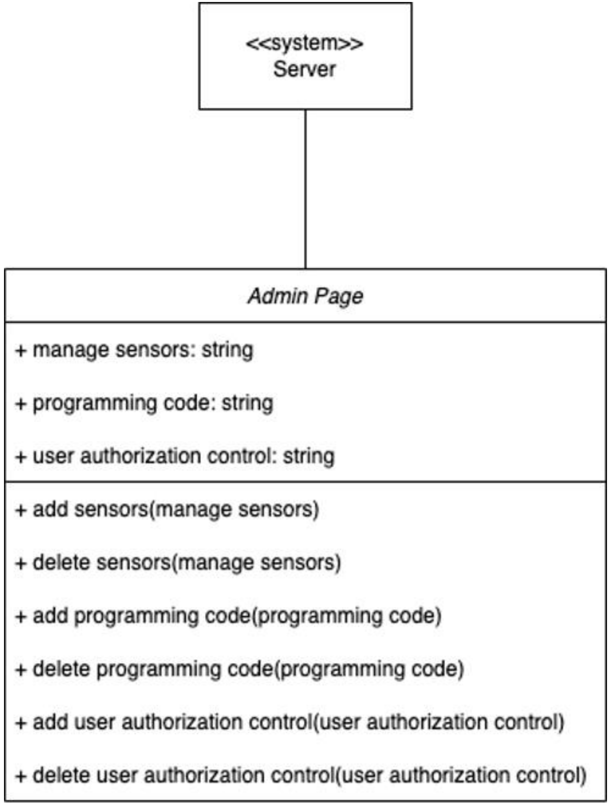


Use case Diagram



System Architecture - Front end

1. Admin Page - Class Diagram



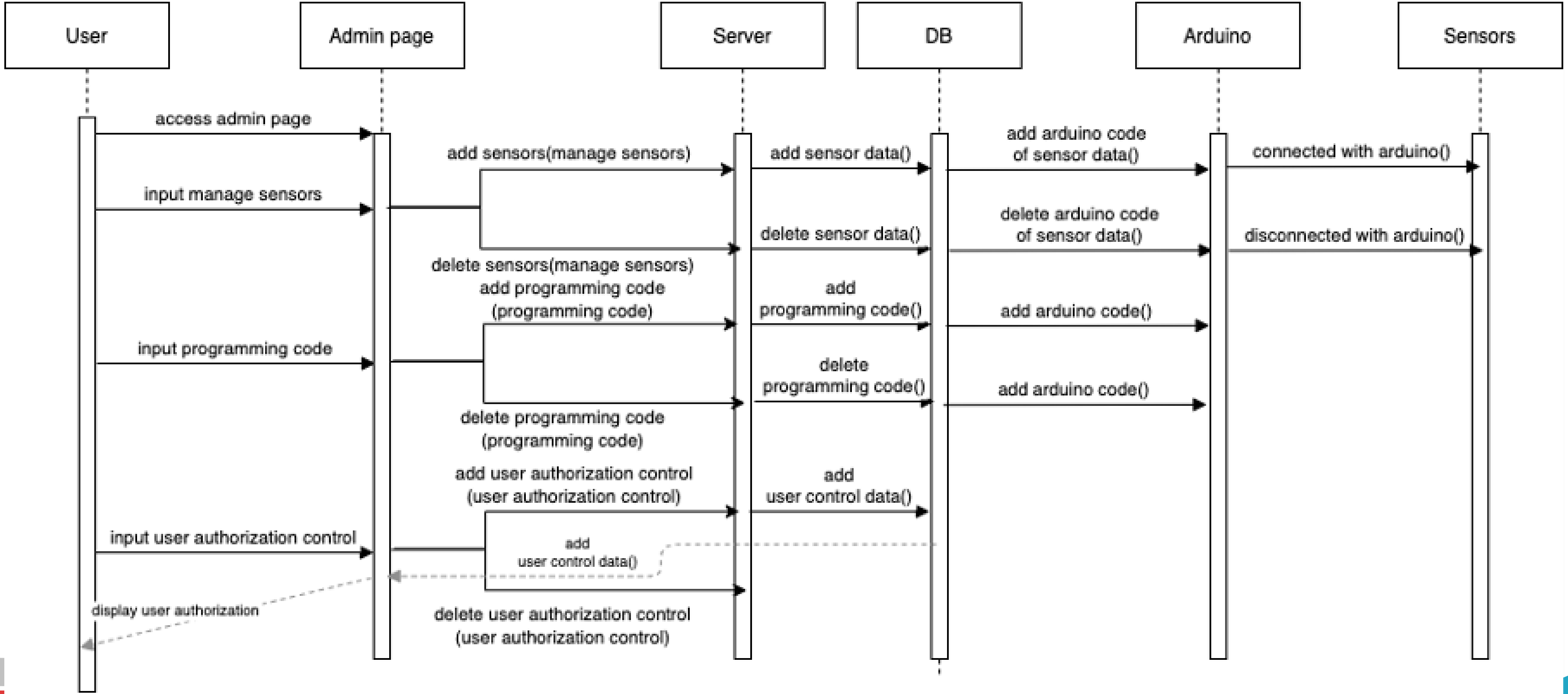
<<system>>
Server

Admin Page

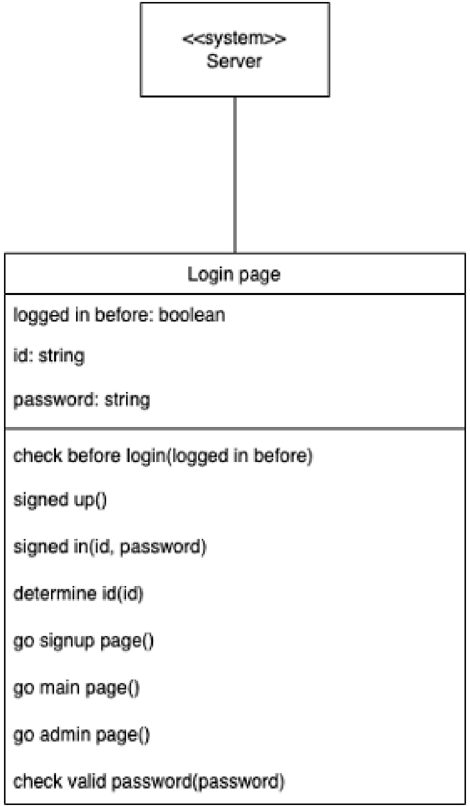
+ manage sensors: string
+ programming code: string
+ user authorization control: string

+ add sensors(manage sensors)
+ delete sensors(manage sensors)
+ add programming code(programming code)
+ delete programming code(programming code)
+ add user authorization control(user authorization control)
+ delete user authorization control(user authorization control)

1. Admin Page - Sequence Diagram



1. Login Page - Class Diagram



<<system>>
Server

Login page

logged in before: boolean

id: string

password: string

check before login(logged in before)

signed up()

signed in(id, password)

determine id(id)

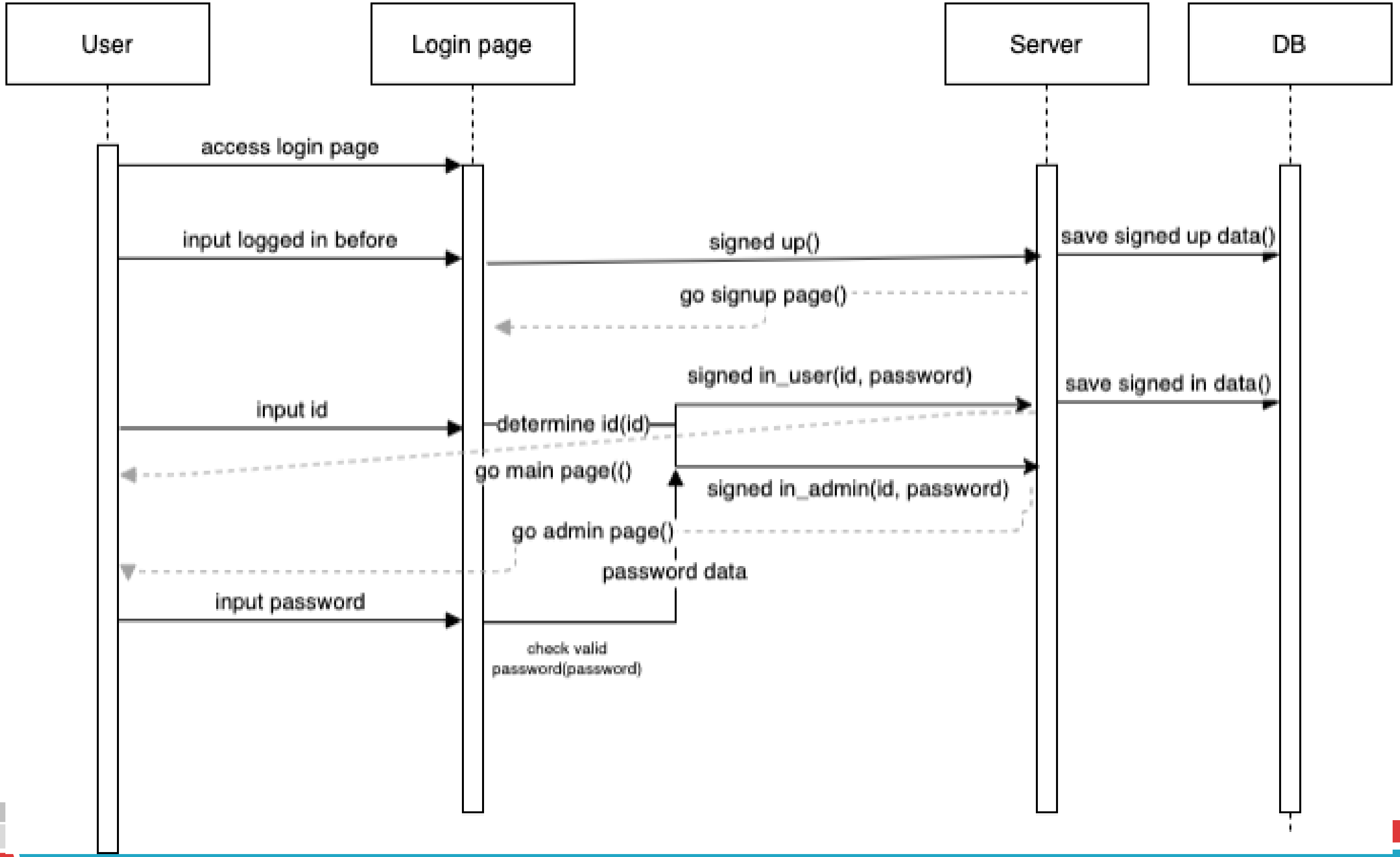
go signup page()

go main page()

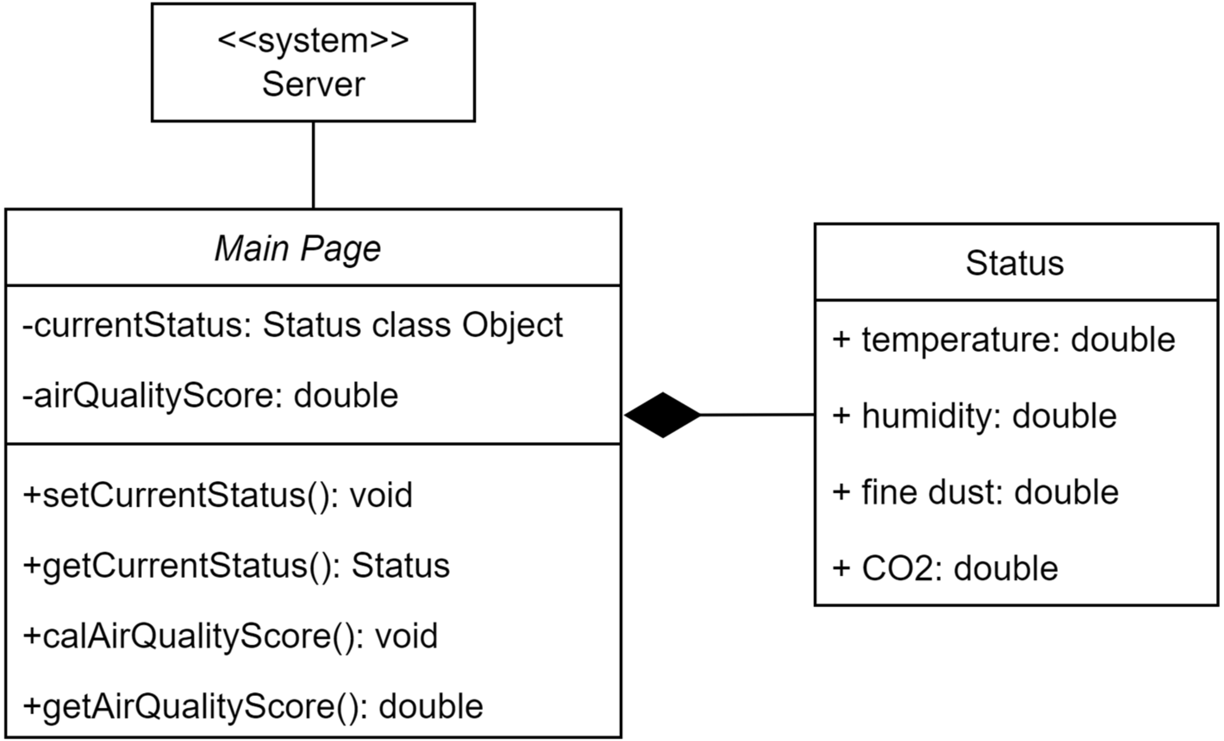
go admin page()

check valid password(password)

1. Login Page - Sequence Diagram



3. Main Page - Class Diagram



<<system>>
Server

Main Page

-currentStatus: Status class Object

-airQualityScore: double

+setCurrentStatus(): void

+getCurrentStatus(): Status

+calAirQualityScore(): void

+getAirQualityScore(): double

Status

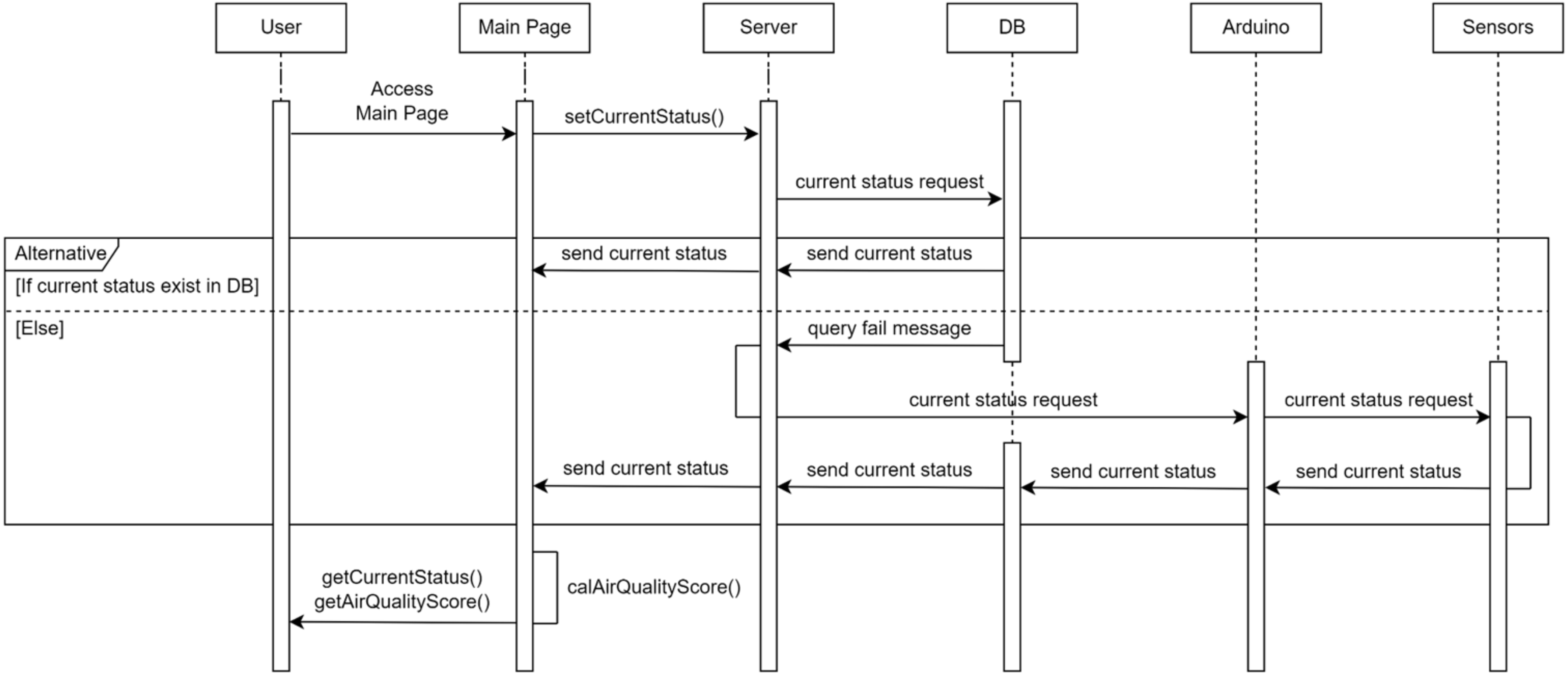
+ temperature: double

+ humidity: double

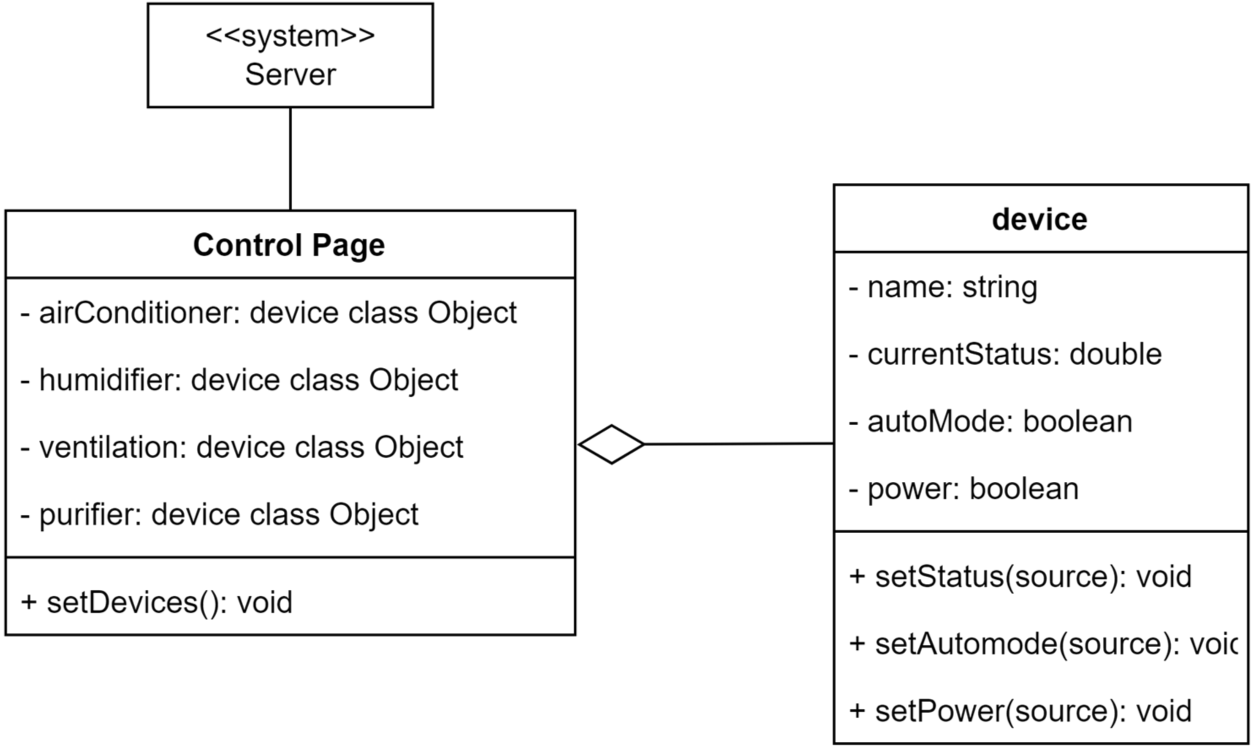
+ fine dust: double

+ CO2: double

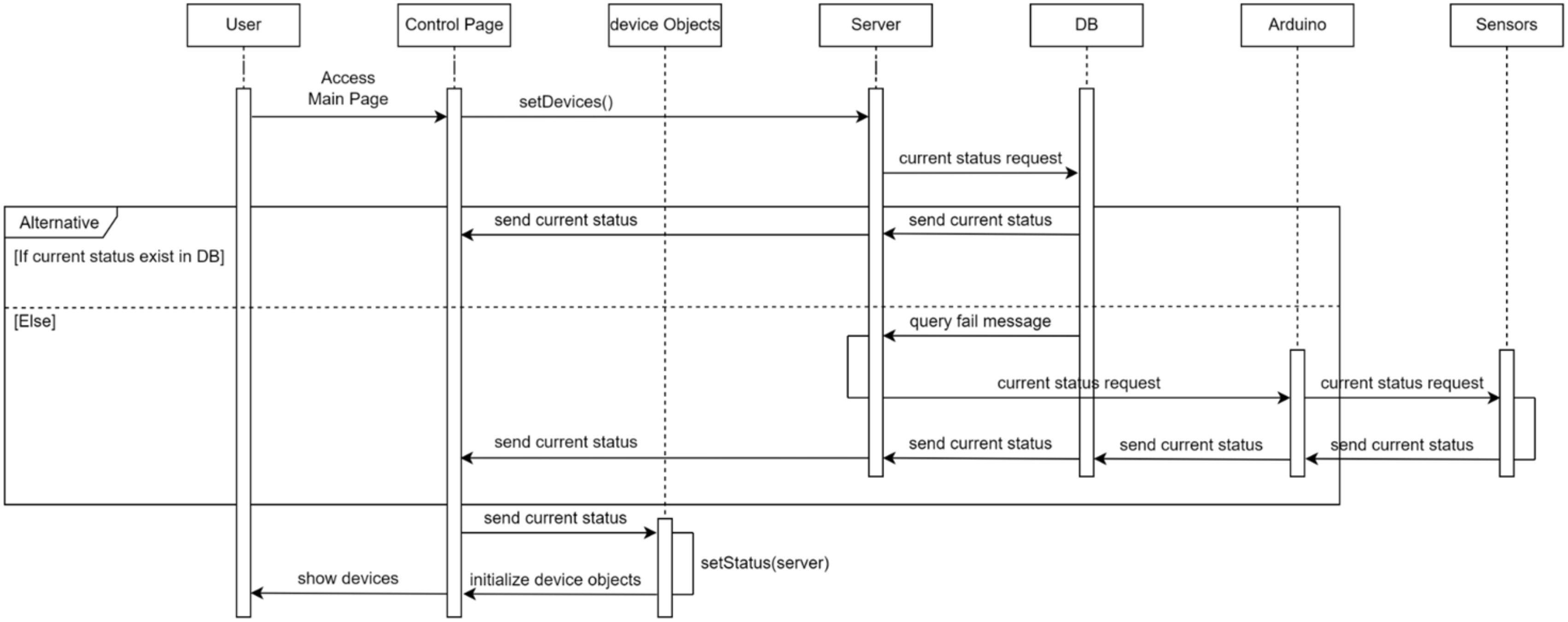
Main Page - Sequence Diagram



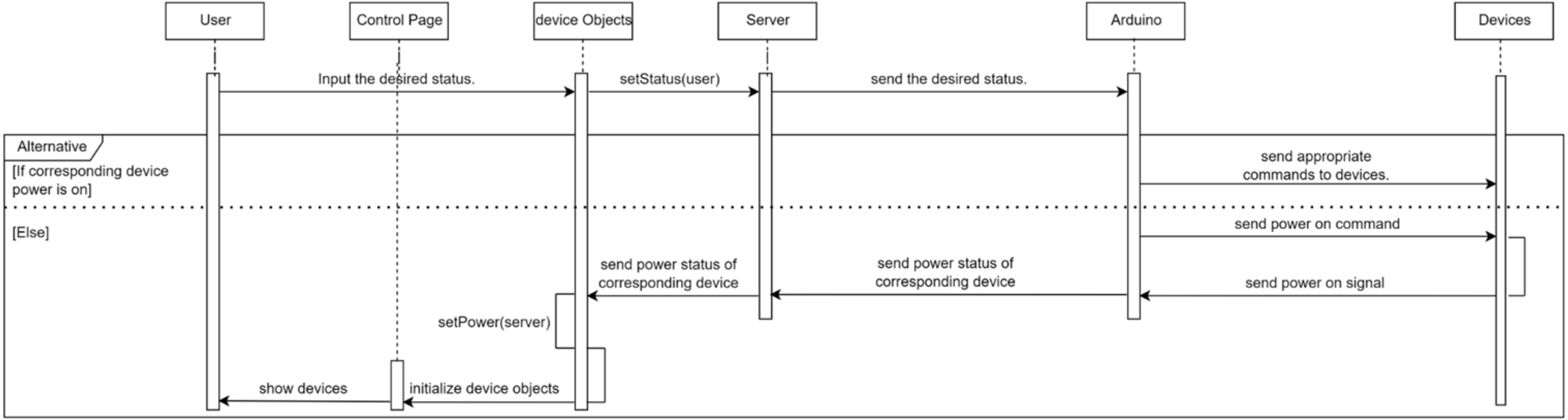
Control Page - Class Diagram


<<system>>
Server
Control Page
- airConditioner: device class Object
- humidifier: device class Object
- ventilation: device class Object
- purifier: device class Object
+ setDevices(): void
device
- name: string
- currentStatus: double
- autoMode: boolean
- power: boolean
+ setStatus(source): void
+ setAutomode(source): voic
+ setPower(source): void

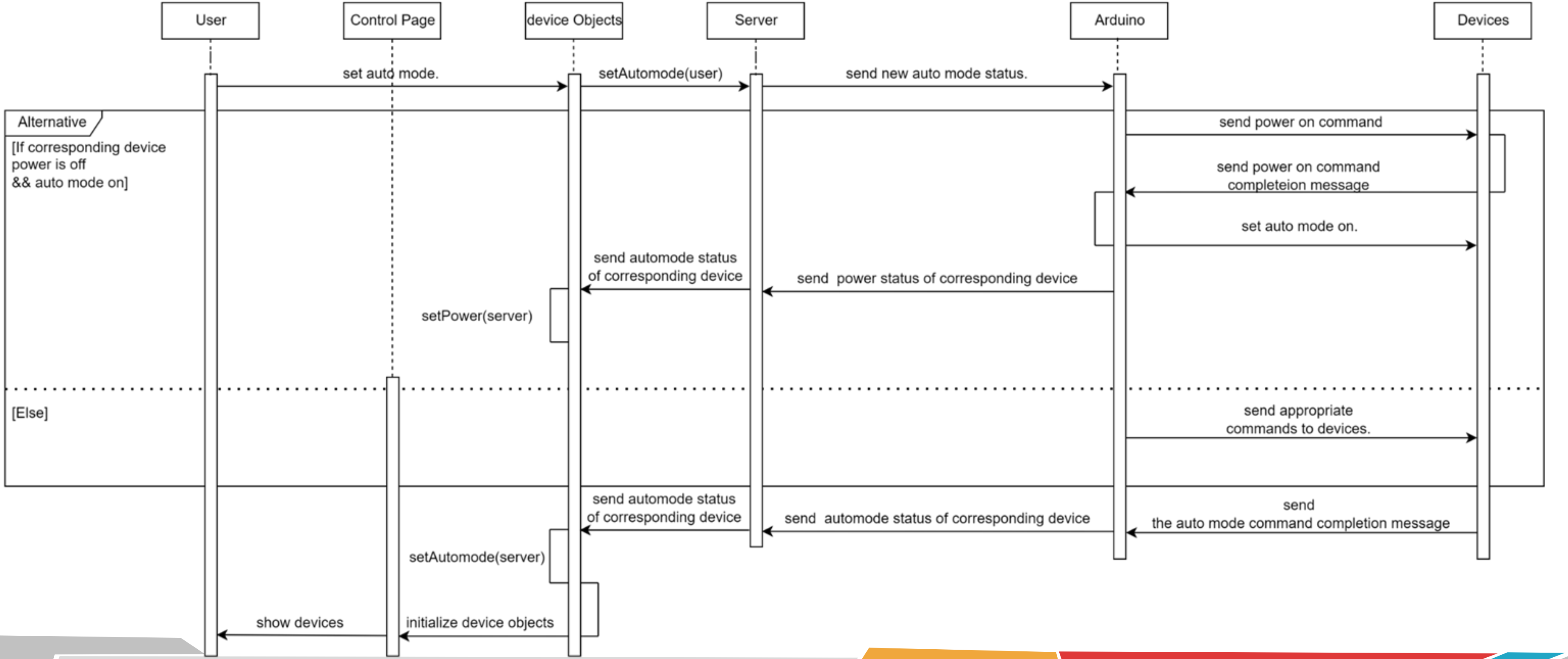
Control Page - Sequence Diagram 1.



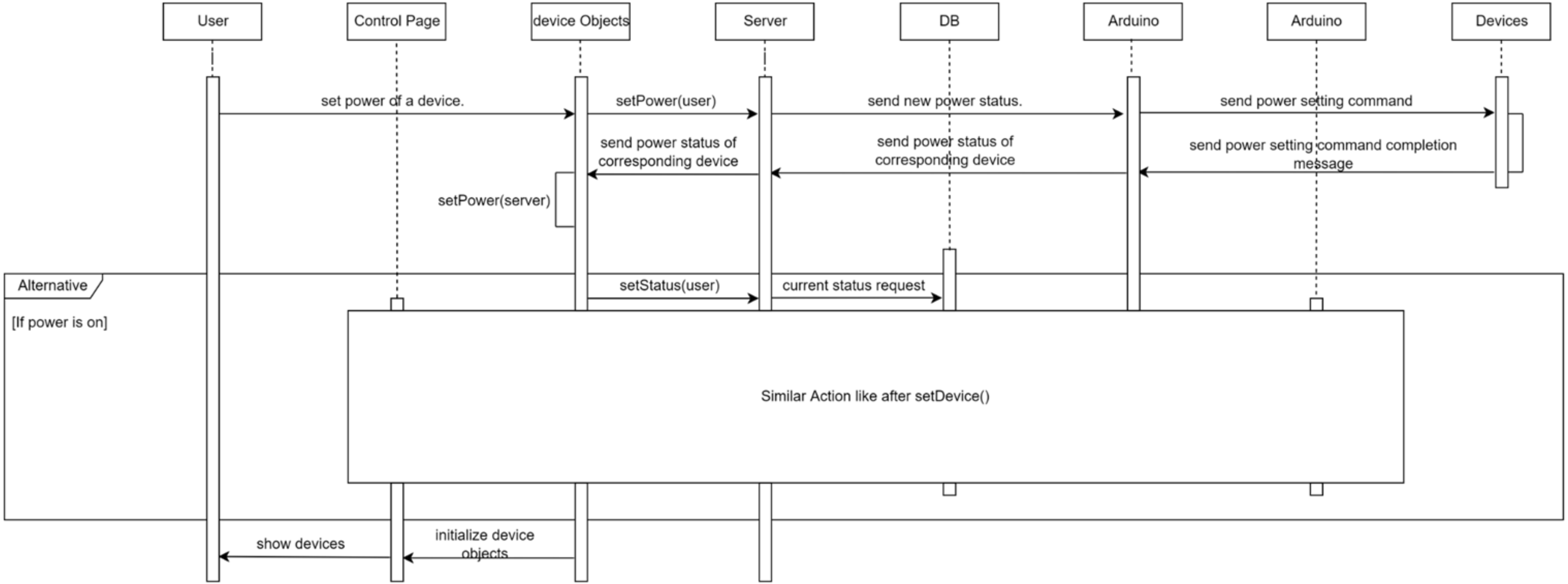
Control Page - Sequence Diagram 2.


User
Control Page
device Objects
Server
Arduino
Devices
Input the desired status.
setStatus(user)
send the desired status.
Alternative
[If corresponding device power is on]
send appropriate commands to devices.
[Else]
send power on command
send power status of corresponding device
send power status of corresponding device
send power on signal
setPower(server)
show devices
initialize device objects

Control Page - Sequence Diagram 3.

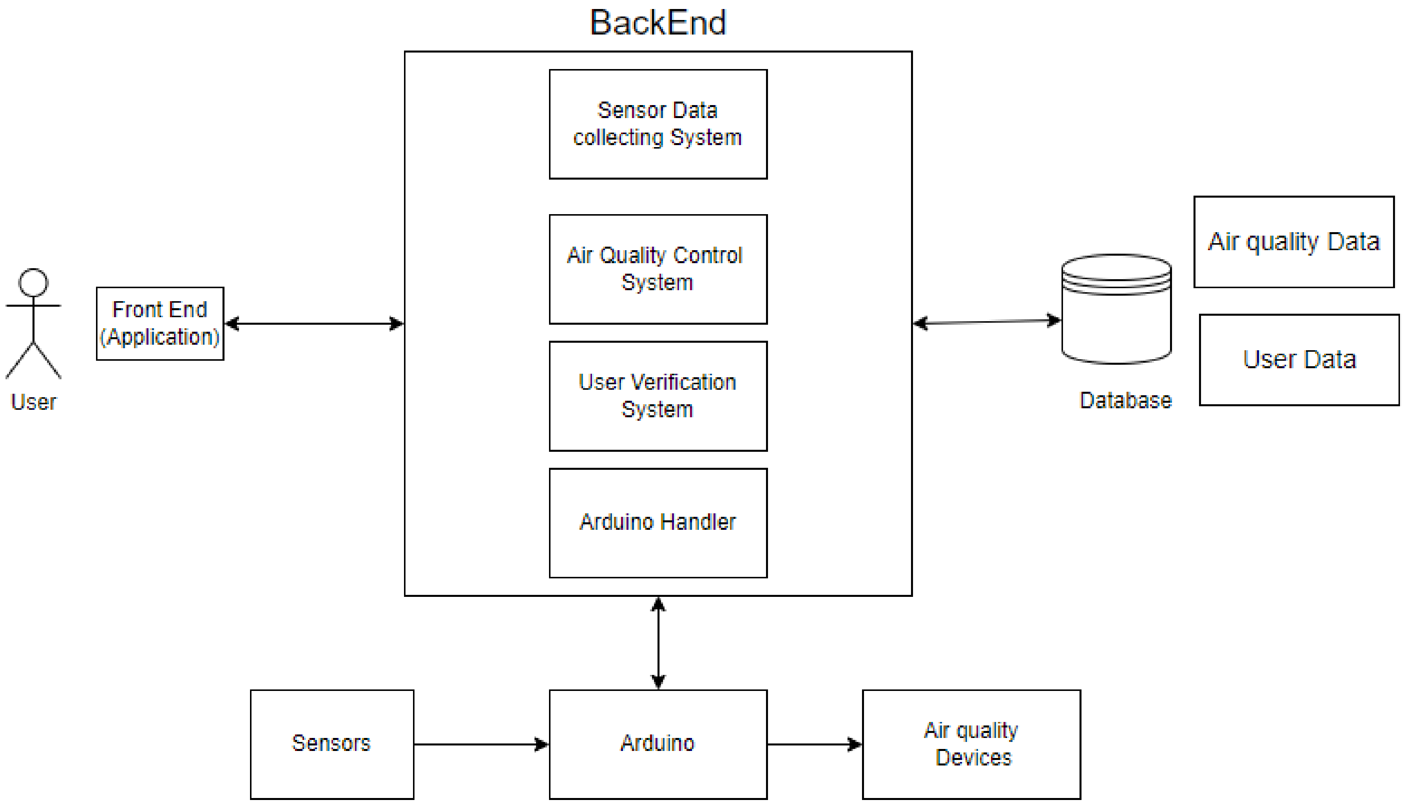

User
Control Page
device Objects
Server
Arduino
Devices
set auto mode.
setAutomode(user)
send new auto mode status.
Alternative
[If corresponding device power is off && auto mode on]
send power on command
send power on command completeion message
set auto mode on.
send automode status of corresponding device
send power status of corresponding device
setPower(server)
[Else]
send appropriate commands to devices.
send automode status of corresponding device
send automode status of corresponding device
send the auto mode command completion message
setAutomode(server)
show devices
initialize device objects

Control Page - Sequence Diagram 4.

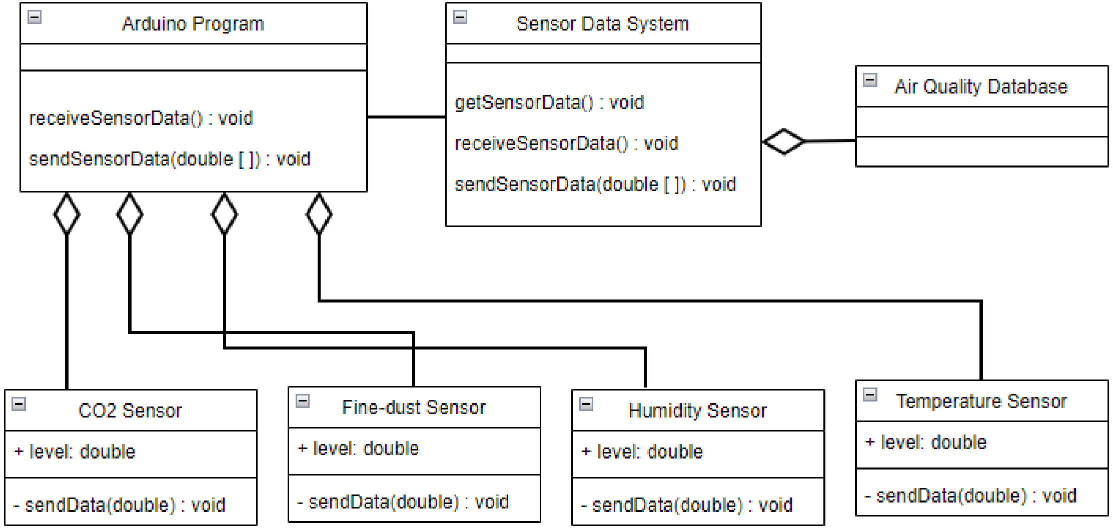

User
Control Page
device Objects
Server
DB
Arduino
Arduino
Devices
set power of a device.
setPower(user)
send new power status.
send power setting command
send power status of corresponding device
send power status of corresponding device
send power setting command completion message
setPower(server)
Alternative
[If power is on]
setStatus(user)
current status request
Similar Action like after setDevice()
show devices
initialize device objects

System Architecture - Back end

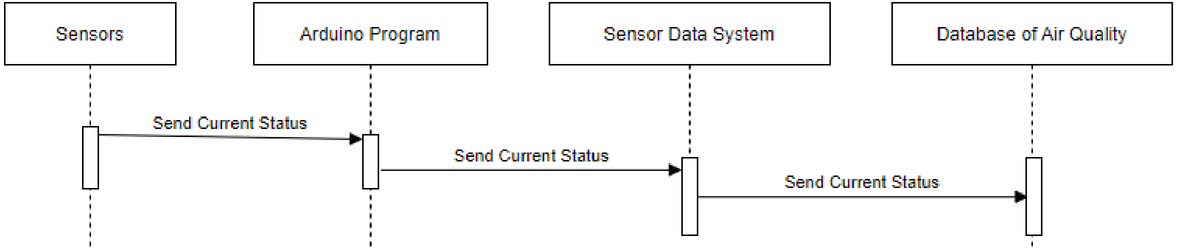
Overall Structure of Backend



Sensor Data collecting System


Arduino Program
receiveSensorData() : void
sendSensorData(double [ ]) : void
Sensor Data System
getSensorData() : void
receiveSensorData() : void
sendSensorData(double [ ]) : void
Air Quality Database
CO2 Sensor
+ level: double
- sendData(double) : void
Fine-dust Sensor
+ level: double
- sendData(double) : void
Humidity Sensor
+ level: double
- sendData(double) : void
Temperature Sensor
+ level: double
- sendData(double) : void

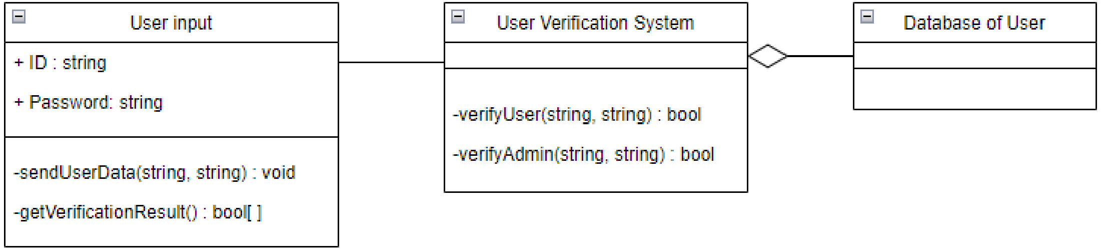
[Class Diagram]



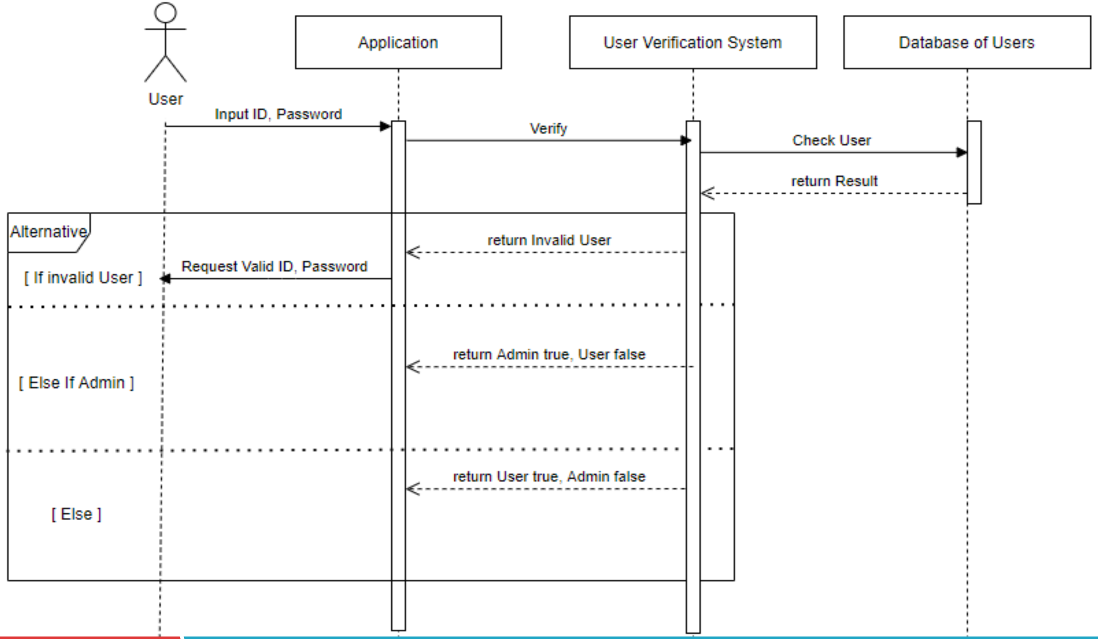
[Sequence Diagram]

User verification System

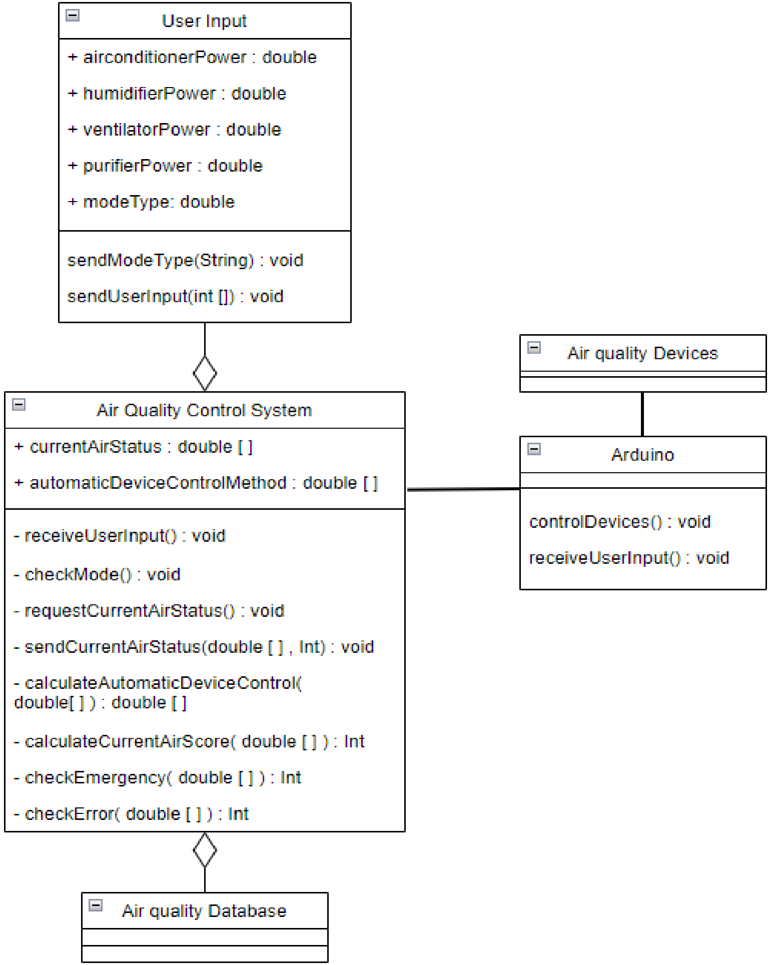
[Class Diagram]


User input
+ ID : string
+ Password: string
-sendUserData(string, string) : void
-getVerificationResult() : bool[ ]
User Verification System
-verifyUser(string, string) : bool
-verifyAdmin(string, string) : bool
Database of User

[Sequence Diagram]



Air Quality Control System



User Input

+ airconditionerPower : double
+ humidifierPower : double
+ ventilatorPower : double
+ purifierPower : double
+ modeType: double

sendModeType(String) : void
sendUserInput(int []) : void

Air Quality Control System

+ currentAirStatus : double [ ]
+ automaticDeviceControlMethod : double [ ]

- receiveUserInput() : void
- checkMode() : void
- requestCurrentAirStatus() : void
- sendCurrentAirStatus(double [ ] , Int) : void
- calculateAutomaticDeviceControl(double[ ]) : double [ ]
- calculateCurrentAirScore(double [ ]) : Int
- checkEmergency(double [ ]) : Int
- checkError(double [ ]) : Int

Air quality Devices

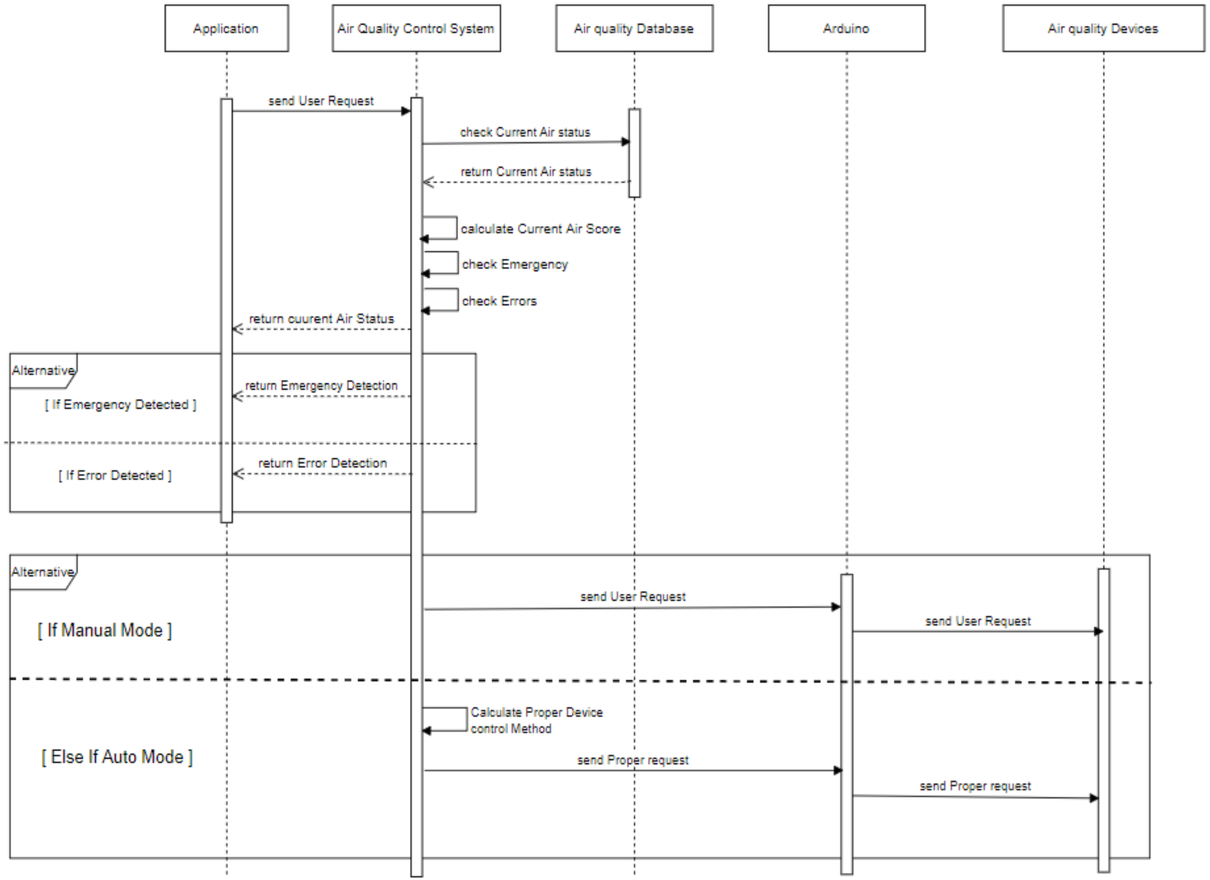
Arduino

controlDevices() : void
receiveUserInput() : void

Air quality Database

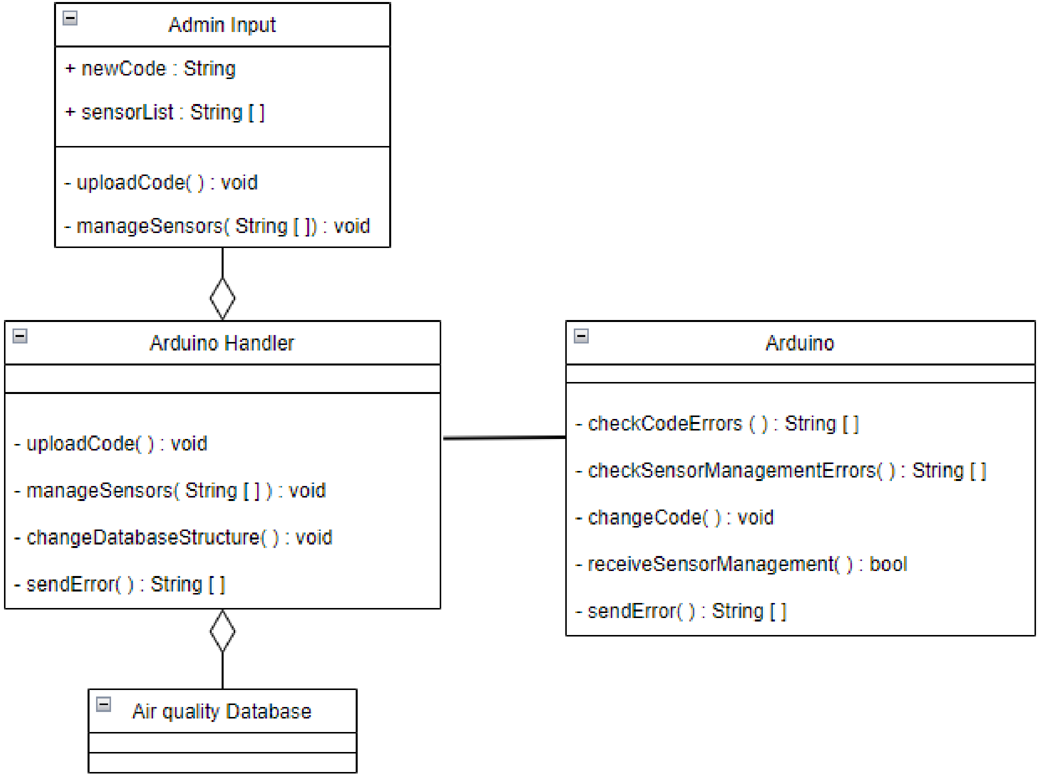
[Class Diagram]

Air Quality Control System



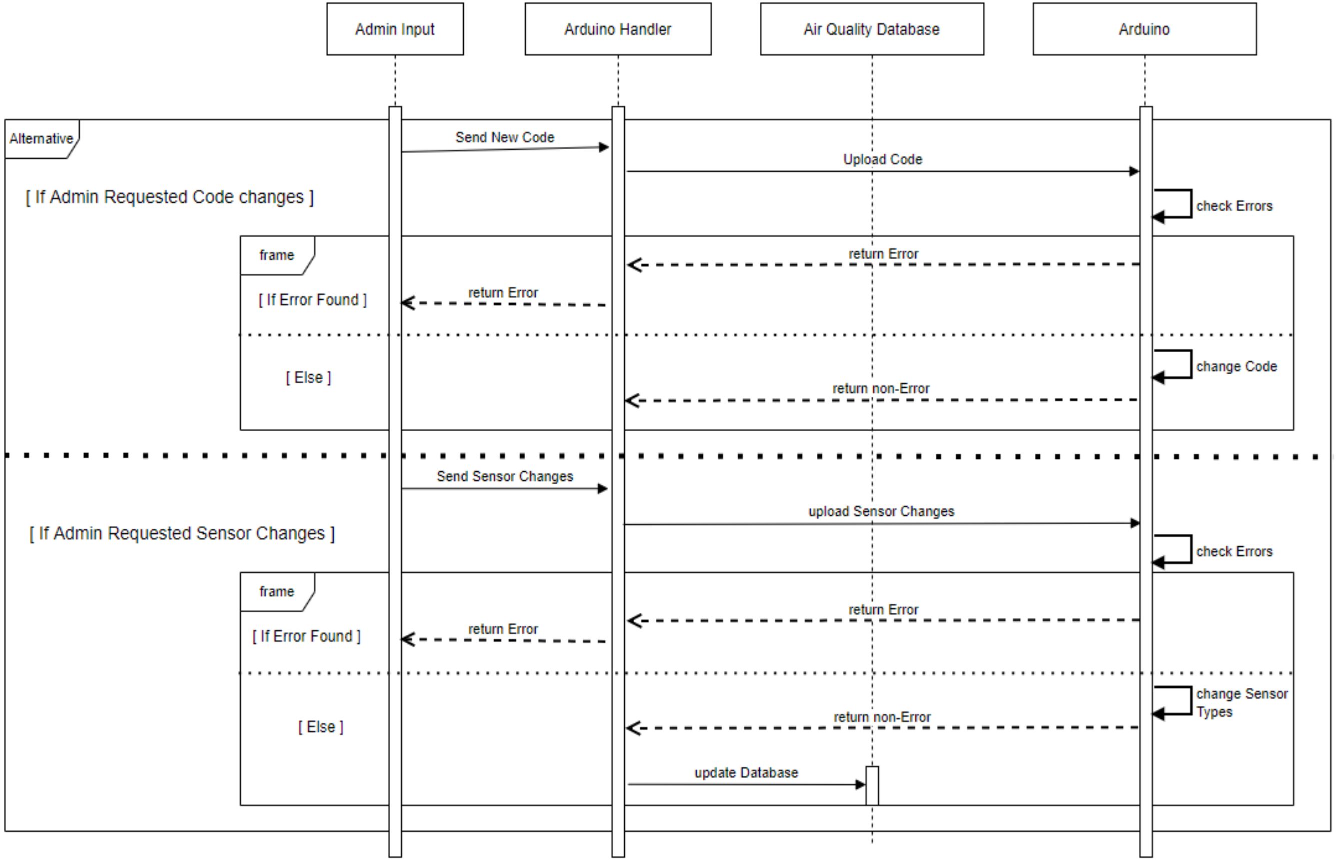
[Sequence Diagram]

Arduino Handler


Admin Input
+ newCode : String
+ sensorList : String [ ]
- uploadCode( ) : void
- manageSensors(String [ ]) : void
Arduino Handler
- uploadCode( ) : void
- manageSensors(String [ ]) : void
- changeDatabaseStructure( ) : void
- sendError( ) : String [ ]
Arduino
- checkCodeErrors ( ) : String [ ]
- checkSensorManagementErrors( ) : String [ ]
- changeCode( ) : void
- receiveSensorManagement( ) : bool
- sendError( ) : String [ ]
Air quality Database

[Class Diagram]

Arduino Handler



[Sequence Diagram]

Protocol Design

Use HTTP protocol

User-Backend Communication

- User to Backend
- Backend to User

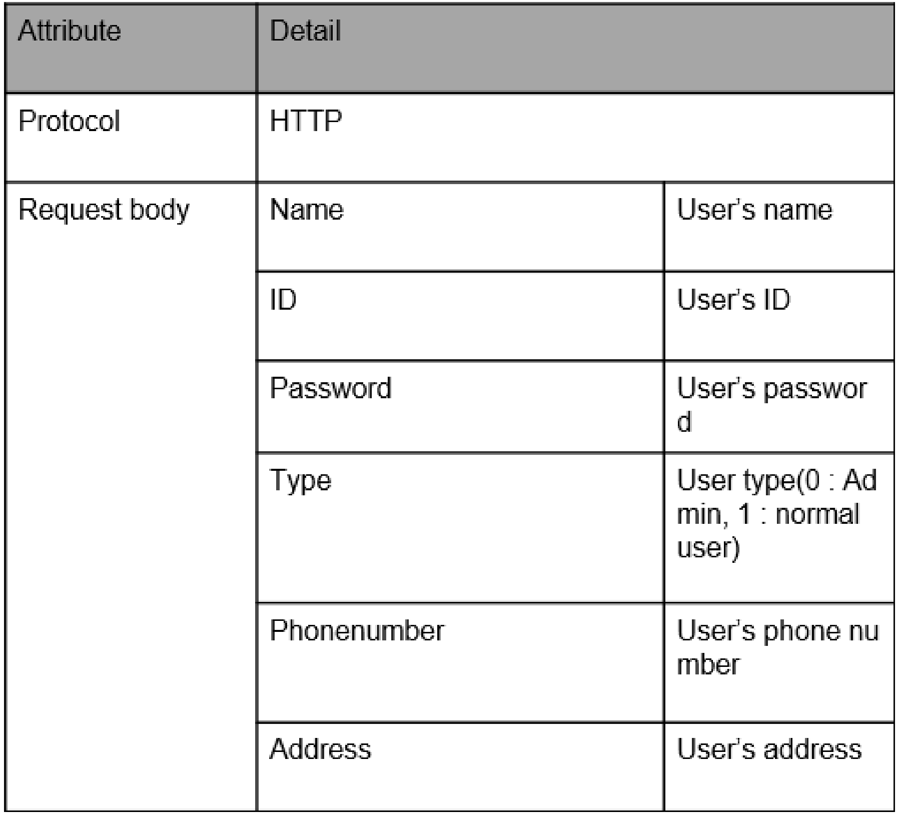
SKIP

Backend-DB Communication

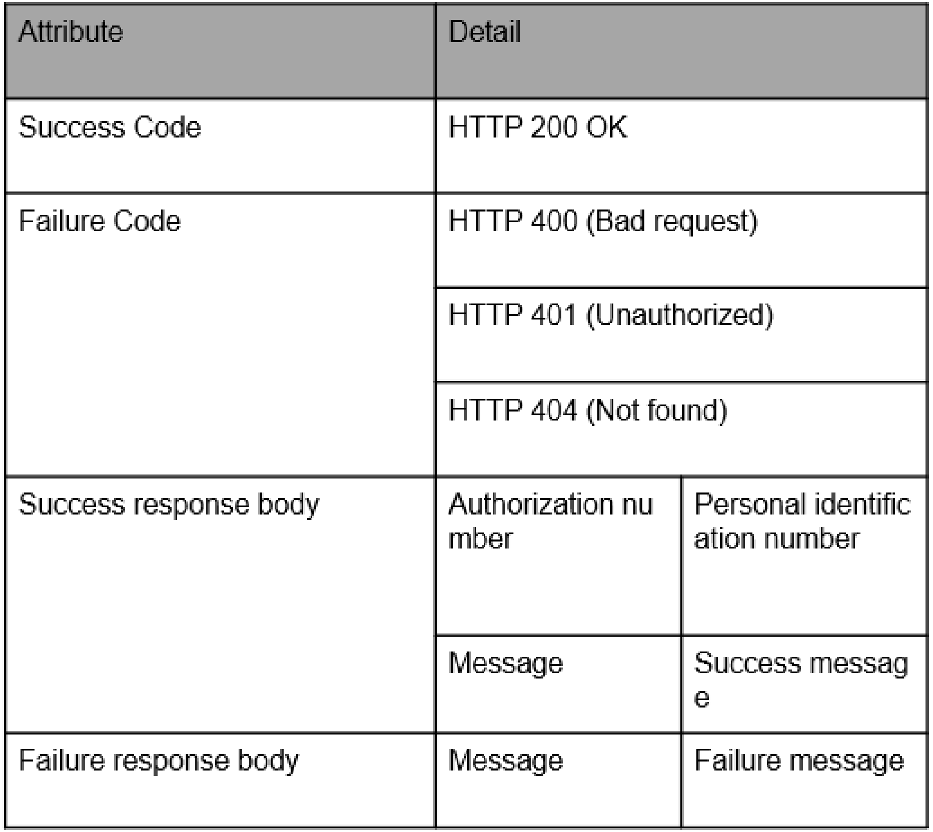
Backend-Arduino communication

- Backend to Arduino
- Arduino to Backend

User to Backend - Registration Attempt

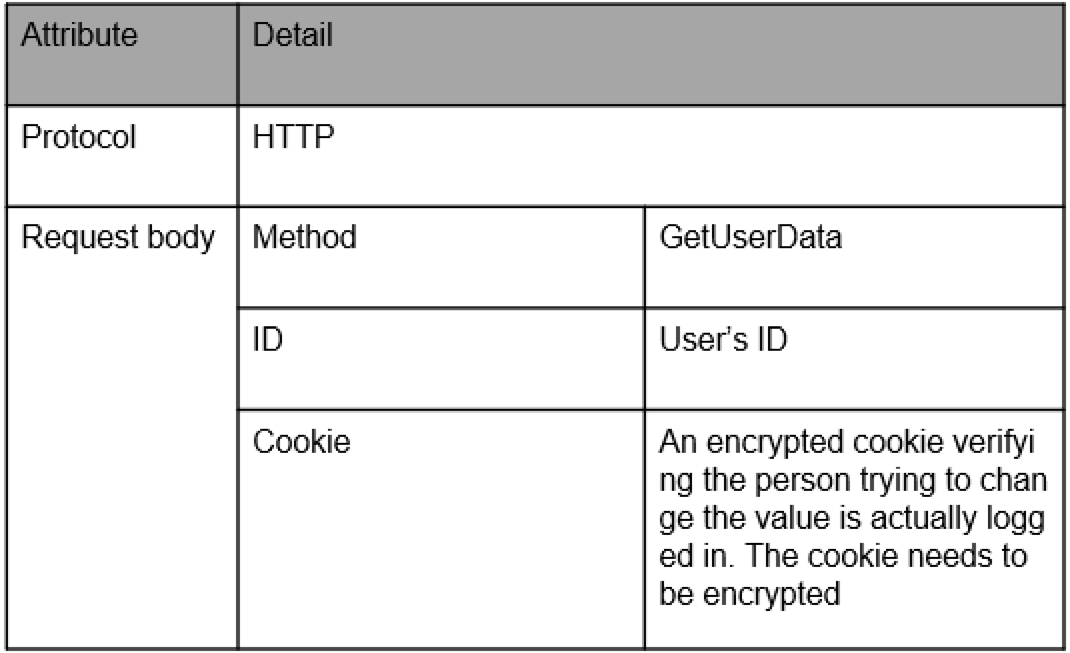


Attribute	Detail	
Protocol	HTTP	
Request body	Name	User’s name
	ID	User’s ID
	Password	User’s password
	Type	User type(0 : Admin, 1 : normal user)
	Phonenumber	User’s phone number
	Address	User’s address

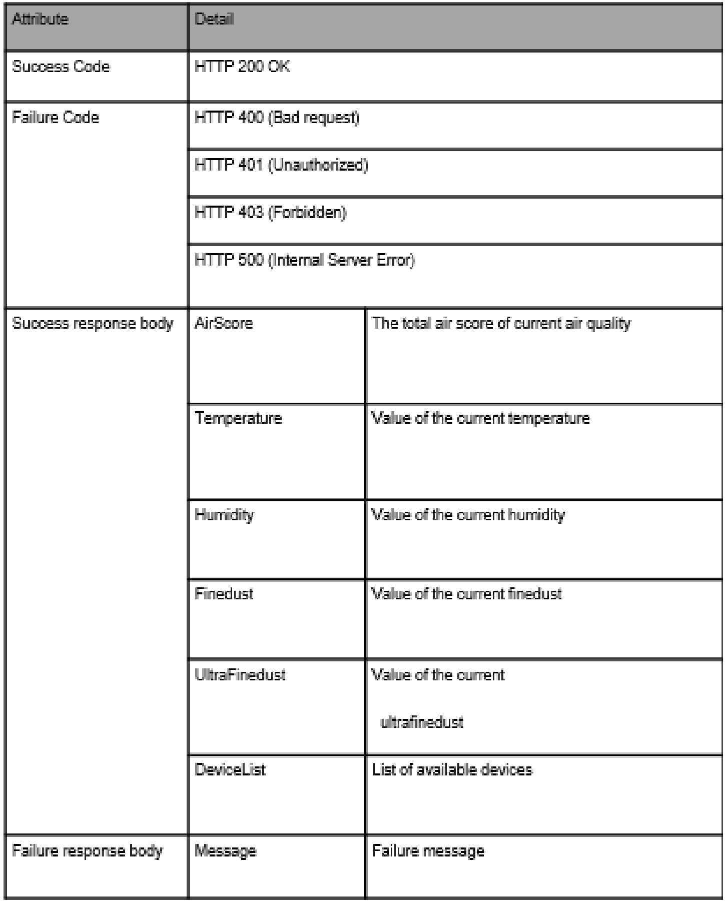


Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 404 (Not found)	
Success response body	Authorization number	Personal identification number
	Message	Success message
Failure response body	Message	Failure message

User to Backend - Get Data

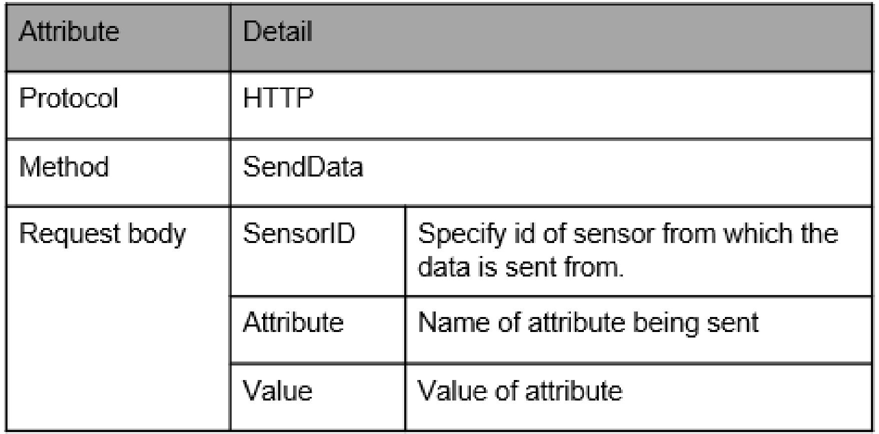


Attribute	Detail	
Protocol	HTTP	
Request body	Method	GetUserData
	ID	User’s ID
	Cookie	An encrypted cookie verifying the person trying to change the value is actually logged in. The cookie needs to be encrypted

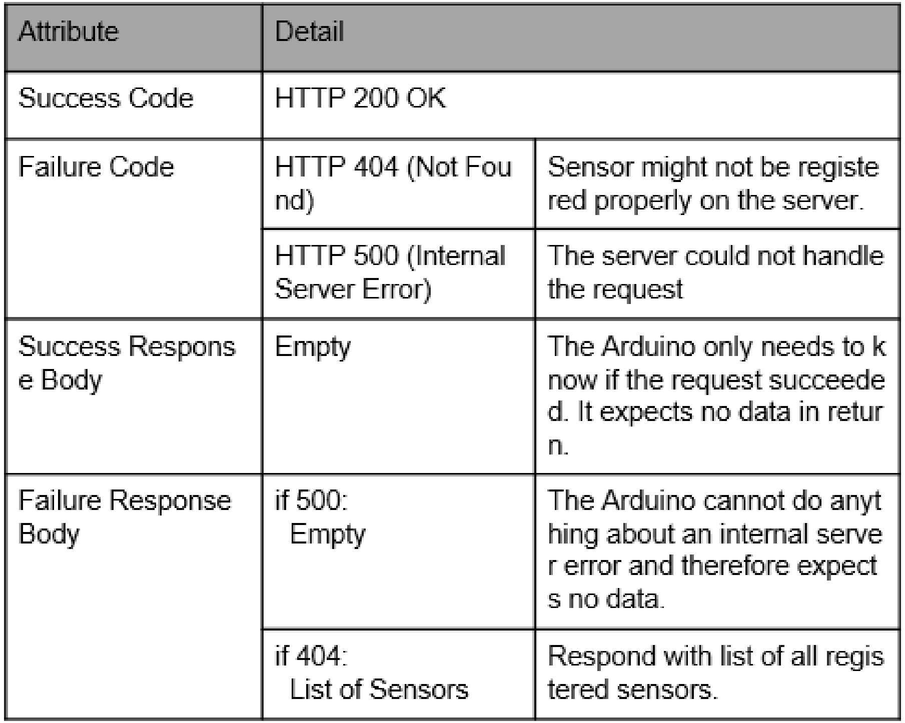


Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 (Bad request)	
	HTTP 401 (Unauthorized)	
	HTTP 403 (Forbidden)	
	HTTP 500 (Internal Server Error)	
Success response body	AirScore	The total air score of current air quality
	Temperature	Value of the current temperature
	Humidity	Value 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

Backend to Arduino - Send Sensor Data

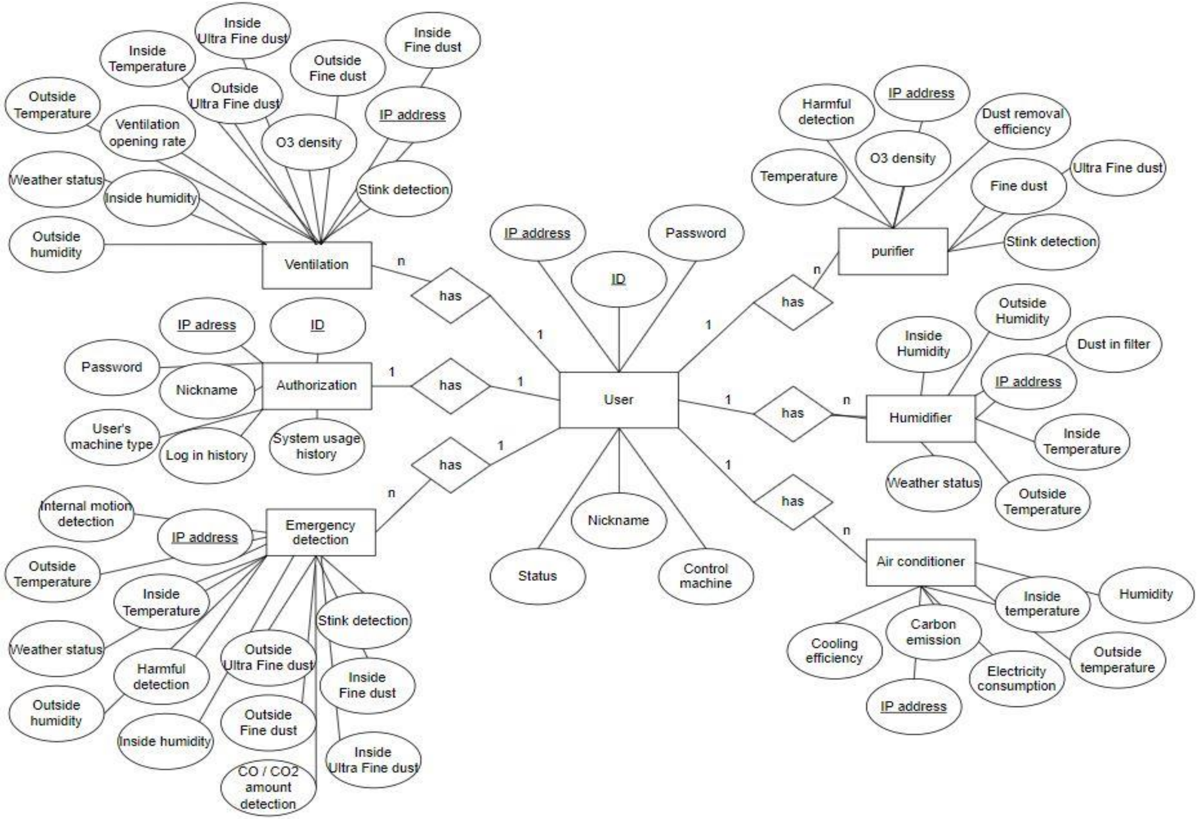


Attribute	Detail	
Protocol	HTTP	
Method	SendData	
Request body	SensorID	Specify id of sensor from which the data is sent from.
	Attribute	Name of attribute being sent
	Value	Value of attribute



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.

Database Design



Performance

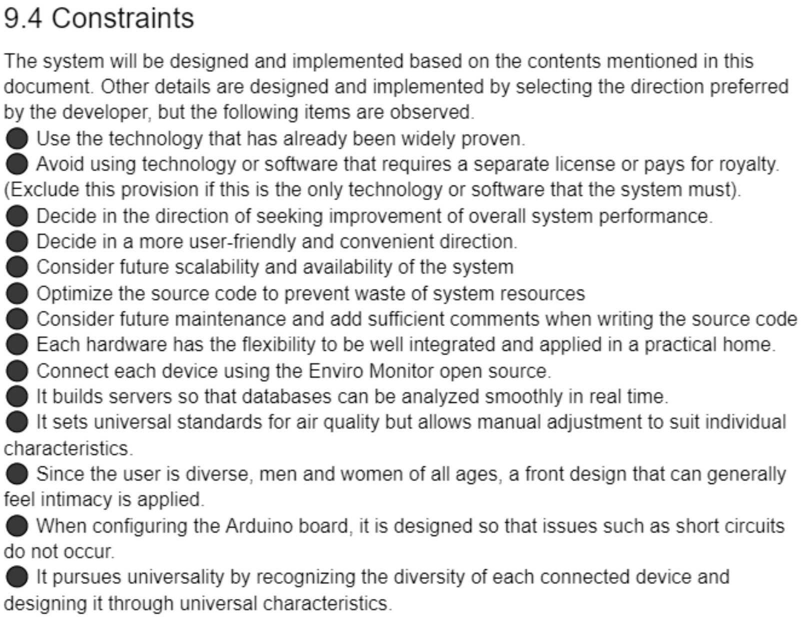
Reliability

Security

Beta Test

Test Case

Development Plan



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.

Thank You