스마트 온습도 조절 시스템

Software Design Specification

2022.05.15

소프트웨어공학

Team6

팀장 서현기

조원 박종찬

조원 신영섭

조원 안예림

조원 황선우

CONTENTS

List of Figures

List of Tables

1. Preface

본 문서는 스마트 온습도 조절기의 개발 후 테스트에 대한 계획서이다. 본 챕터에선 문서의 목적, 범주, 용어들의 정의 및 약어 설명, 참고 문헌, 본 문서의 개괄적인 요약이 담겨있다

1.1 Readership

이 문서는 총 열 개의 항목으로 이루어져 있고 각 항목은 여러 개의 세부 항목으로 나뉘어져 있다. 주요 독자는 이 문서의 작성자인 팀6 조원들이고 교수님, 조교, 다른 학생들도 추가적으로 독자가 될 수 있다

1.2 Scope

이 소프트웨어 디자인 명세서는 소프트웨어 엔지니어링과 소프트웨어 품질 엔지니어링에서 각 가정의 건강을 위해 스마트 온습도 조절기를 구현하는데 필요한 디자인 정의로 사용된다

1.3 Objective

이 소프트웨어 설계 문서의 주요 목적은 스마트 온습도 조절기의 기술적 설계 측면에 대한 설명을 제공하는 것이다. 이 문서는 이 프로젝트의 소프트웨어 아키텍처를 포함하고 있다. 더불어 시스템의 다양한 측면을 표현하기 위해 여러 형태의 다이어그램(Class Diagram, Sequence Diagram 등)과 시스템 구조적 개요를 제공한다.

1.4 Document Structure

Preface: 이 장에서는 독자층, 문서의 범위와 구조, 시스템의 목적에 대해서 설명한다

Introduction: 이 장에서는 문서에 사용된 여러 형태의 다이어그램들에 관한 설명과 문서 작성에 사용된 여러 도구, 이 문서가 쓰인 목적에 대해 설명한다.

System Architecture – Overall: 이 장에서는 context diagram, sequence diagram, use case diagram을 사용해 시스템의 전체적인 구조에 대해서 설명한다.

System Architecture – Frontend: 이 장에서는 class diagram, sequence diagram을 사용해 Frontend 시스템의 구조에 대해서 설명한다.

System Architecture – Backend: 이 장에서는 class diagram, sequence diagram을 사용해 Backend 시스템의 구조에 대해서 설명한다.

Protocol Design: 이 장에서는 클라이언트와 서버 간의 통신에 사용되는 여러 프로토콜 디자인에 대해 설명한다.

Database Design: 이 장에서는 ER diagram 과 SQL DDL을 사용해 데이터베이스 디자인에 대해 설명한다.

Testing Plan: 이 장에서는 이 시스템을 테스트하는 계획에 대해 설명한다.

Developing Plan: 이 장에서는 시스템 개발을 위해 사용할 툴, 제약사항, 가정, 종속성에 대해 설명한다.

Supporting Information: 이 장에서는 이 문서의 기준과 역사에 대해 설명한다

2. Introduction

이 프로젝트는 '스마트 온습도 조절기'를 제작하기 위한 것이다. 이 프로젝트의 목표는 가정의 건강을 지키기 위해 가정의 온습도를 자동으로 설정해주기 위한 것이다. 팬데믹 상황과 더불어 가정의 건강 환경은 최근 들어 급격하게 악화되었다. 이러한 상황에서 온도와 습도를 가장 쾌적의 상태에 맞춤으로써 세균 등과 같은 유해요소로부터 가정을 지킬 수 있다. 또한 이외에도 쾌적 그래프 제공, 개별 온습도 조절, 건강 설문 등과 같은 기능도 제공함으로써 프로그램의 완성도를 높이려 노력하였다. 본 명세서에는 스마트 온습도 조절기에 사용되는 여러 디자인 구조들이 묘사되어 있다. 디자인 구조들은 이전에 작성한 SRS를 기반으로 설계하였다.

2.1 Objectives

이 장에서는 문서에 사용된 여러 형태의 다이어그램들에 관한 설명과 문서 작성에 사용된 여러 도구, 이 문서가 쓰인 목적에 대해 설명한다.

2.2 Applied Diagrams

2.2.1 UML

UML(Unified Modeling Language)은 소프트웨어 개발 및 객체 지향 설계 분야에서 표준화된 시각화 방법으로 설계된 픽토그램 기반의 그래픽 모델링 언어이다. UML은 모델링 소프트웨어 아키텍처의 표준으로 객체 지향 소프트웨어 개발에 필요한 문서의 설계를 용이하게 하기 위한 것이다. 표현할 수 있는 다양한 요소는 개체/소프트웨어의 활동, 배우, 프로세스, 데이터베이스 스키마, 소프트웨어 구성 요소, 구성 요소의 재사용 등이 있다. 또한 생성된 문서에서 코드의 전체 또는 일부(예: Java 언어)를 자동으로 생성할 수도 있습니다

2.2.2 Use case diagram

Use case diagram은 소프트웨어 및 기타 시스템의 구조 및 동작을 모델링하기 위한 언어인 UML(Unified Modeling Language)의 다이어그램 유형 중 하나입니다. 각각의 종속성 및 관계가 있는 사용 사례 및 행위자를 제시합니다. Use case diagram은 프로세스 설명을 나타내지 않고 대신 활동, 시퀀스 또는 협업 다이어그램으로 나타낼 수 있습니다

2.2.3 Sequence diagram

Sequence diagram은 UML(Unified Modeling Language)의 의미에서 상호 작용을 그래픽으로 나타내는 동작 다이어그램이다. 소프트웨어 및 기타 시스템의 모델링 언어인 UML의 맥락에서 Sequence diagram은 네 가지 유형의 상호 작용 다이어그램 중 하나이다. Sequence diagram은 일반적으로 시스템 플로우 내의 의사결정 트리를 통한 경로를 나타낸다. 모든 의사결정 옵션이 포함된 개요를 개발하려면 가능한 각 플로우에 대해 별도의 Sequence diagram을 모델링해야 한다.

2.2.4 Class diagram

Class diagram은 클래스, 인터페이스 및 해당 관계의 그래픽 표현(모델링)을 위한 UML(Unified Modeling Language)의 구조 다이어그램이다. 객체 지향에서 클래스는 객체의 공통 구조 및 공통 동작(분류)에 대한 설명에 대한 추상적인 일반 용어이다. 객체를 추상화하는 역할을 한다. 다른 클래스와 상호 작용하여 객체 지향 분석 및 설계에서 구분된 시스템의 모델링을 가능하게 한다. 클래스 다이어그램은 소프트웨어 및 기타 시스템의 모델링 언어인 UML의 14가지 유형의 다이어그램 중 하나이다

2.2.5 Context diagram

Context diagram은 초기 설계 또는 분석 단계에서 시스템 환경을 모델링하는 데 사

용된다. Context diagram은 데이터 흐름 다이어그램의 가장 높은 계층적 수준을 나타내며 시스템과 환경의 인터페이스가 매핑되는 추상 데이터 흐름 다이어그램이다. 원으로 표시되고 숫자 0이 지정된 정확히 하나의 프로세스로 구성됩니다. 이 프로세스는 점차 더 세부적으로 세분화되고 구성 요소로 나뉜다. 시스템과 상호 작용하는 구성 요소는 직사각형으로 표시되고 데이터 흐름은 프로세스에서 또는 프로세스로 가는 화살표로 표시된다

2.2.6 Entity Relationship diagram

Entity Relationship model은 데이터베이스의 엔티티 표현을 용이하게 하는 데이터 모델 도구로 Peter Chen이 1976년에 정의했습니다. ER 모델(=Entity Relationship model)은 기본적으로 그래픽(ER 다이어그램, 약어 ERD)과 여기에 사용된 요소에 대한 설명으로 구성된다. ER 모델은 사용자와 개발자 간의 이해를 위한 응용 프로그램 개발의 개념적 단계와 구현 단계에서 기본으로 제공된다. 대부분 관계형 데이터베이스의 설계를 위해 데이터 모델에 대한 다른 그래픽 표현이 있음에도 불구하고 ER 모델은 데이터 모델링에 대한 사실상의 표준이다

2.3 Applies Tools

2.3.1 Edrawmax

EdrawMax는 순서도, 조직도, 마인드 맵, 네트워크 다이어그램, 평면도, 워크플로 다이어그램, 비즈니스 차트 및 엔지니어링 다이어그램을 만드는 데 도움이 되는 2D 비즈니스 기술 다이어그램 작성 소프트웨어이다. 최신 버전인 EdrawMax 11.5.0은 2021년 11월 Microsoft Windows, macOS 및 Linux용으로 출시되었다. EdrawMax는 Visio와 유사한 다이어그램 작성 도구이다.

2.3.2 Lucidchart

Lucidchart는 웹 기반 다이어그램 작성 애플리케이션으로 사용자가 차트 및 다이어그램 그리기, 수정 및 공유에 대해 시각적으로 협업하고 프로세스, 시스템 및 조직 구조를 개선할 수 있다. Lucidchart는 Google, GE, NBC Universal 및 Amazon과 같은 회사에서 사용한다. Lucidchart는 HTML5를 지원하는 브라우저에서 실행된다. 즉, Adobe Flash와 같은 타사 소프트웨어의 업데이트가 필요하지 않다. 이 툴은 완성도 높은 ER 다이어그램 제작이 가능하다고 판단하여 이 명세서 작성에 사용되었다

2.4 Project Scope

이 프로젝트는 코로나 19와 기타 등등의 유해 병균으로부터 가정을 보호하기 위해 집안의 온도와 습도를 자동으로 조절해주는 기능을 제공할 것이다. 집안의 환경을 더 쾌적하고 건강하게 하기 위해 자동으로 조절해주는 기능에 더해 온습도 관련 각 기기를 조절하는 기능, 현재 상태 설문 기능 등을 포함하여 종합적으로 관리해줄 수 있도록 하였다. 이 프로젝트로 적정 수치의 온습도에 의하여 건강한 가정을 유지하는 역할을 기대한다

2.5 References

- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library. https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc

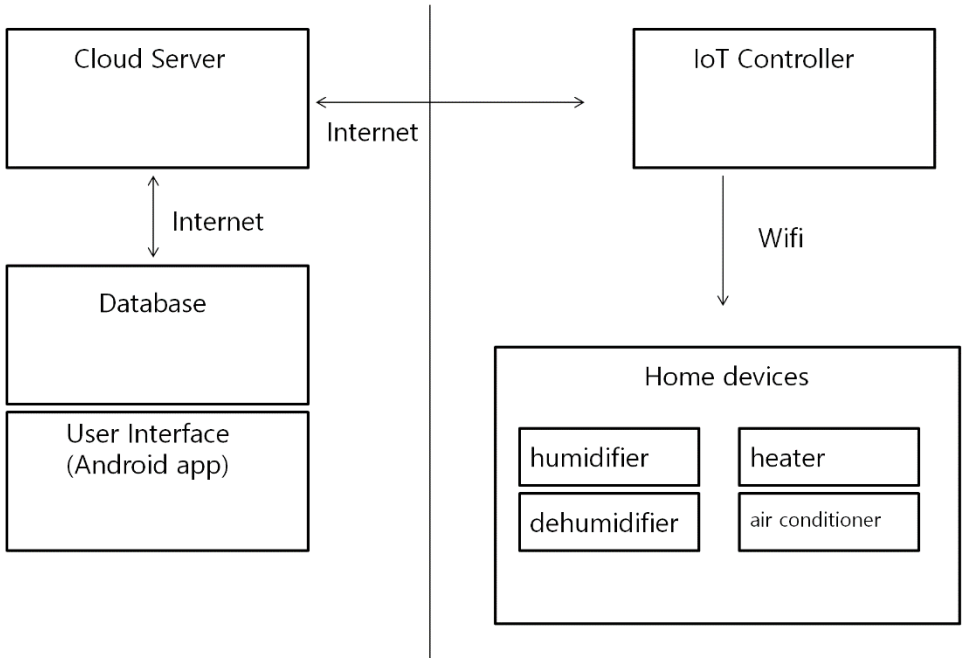
- Team 7, 2021 Spring, Software Design Document, SKKU

3. System Architecture – Overall

3.1 Objectives

이 챕터에는 시스템의 기능과 기능별 작업 처리 과정이 서술되어 있다.

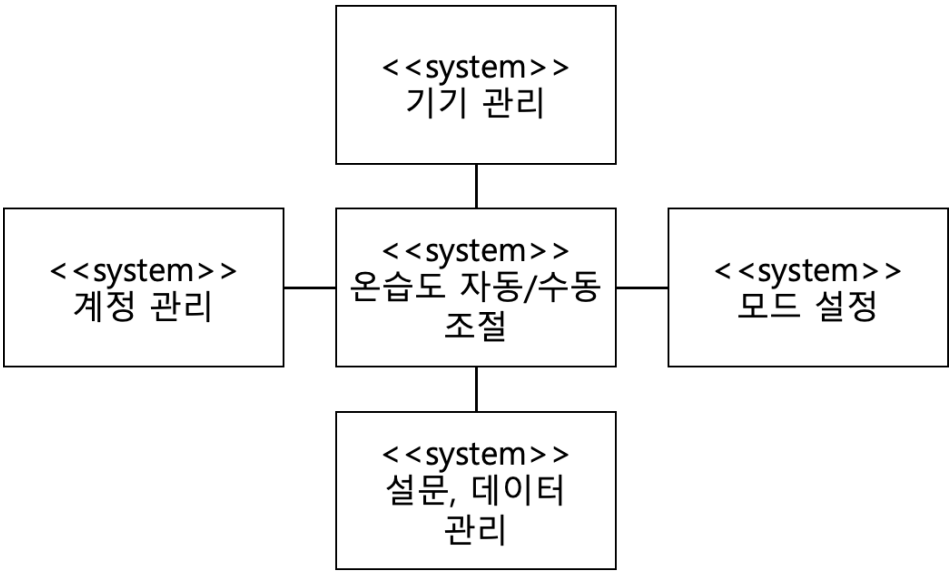
3.2 System Organization



[Figure 1] System Architecture

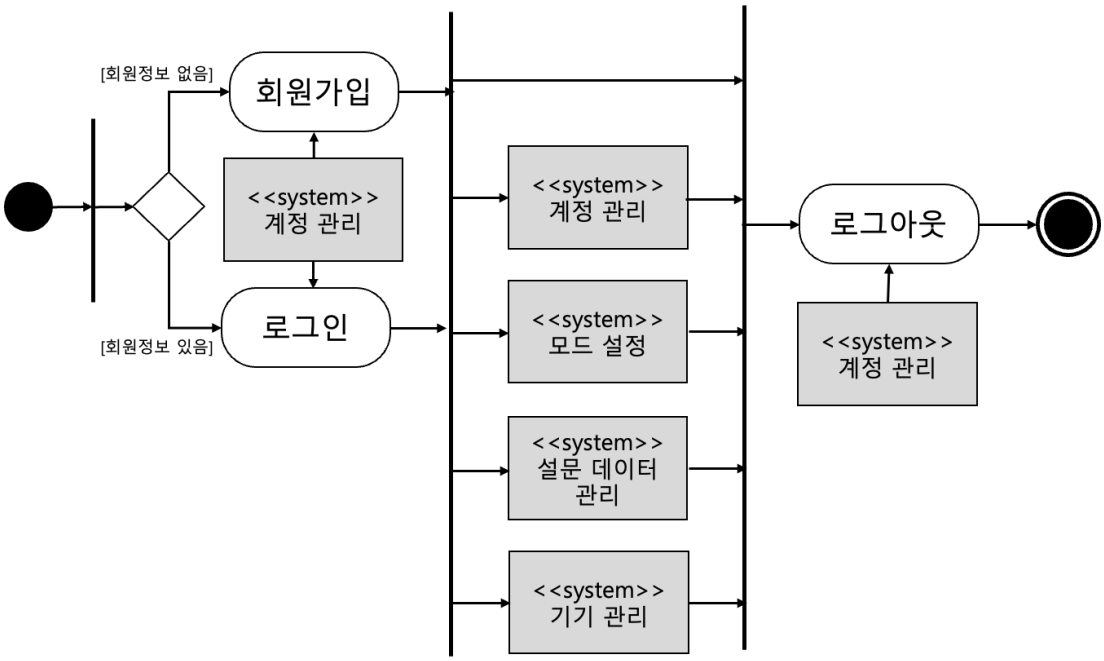
본 프로그램은 AWS IoT 클라우드 서버 및 Android Studio로 개발된 모바일 애플리케이션을 활용하여 온도와 습도를 자동으로 조절하는 스마트홈 기능을 구현하고자 한다. 사용자가 모바일 기기를 사용하여 AWS IoT 클라우드 서버에 등록된 계정으로 로그인을 하면, 해당 계정에 등록된 Raspberry Pi가 내장된 IoT 디바이스에 대한 관리 권한을 얻고 조작할 수 있다. 사용자는 등록된 IoT 디바이스를 자동 또는 수동 조작하여 온습도를 측정하거나 제어할 수 있다. 자동 조작 시에는 클라우드 서버로부터 현재 온습도 정보를 받아 온도와 습도를 규칙 관리 시스템에 명시된 범위 내로 조절한다. 또한 프로그램은 매일 사용자로부터 설문을 받아 서버의 데이터베이스에 이를 저장하고, 데이터 관리 시스템에서 해당 데이터를 통해 쾌적 값을 도출할 수 있다.

3.2.1 Context Diagram

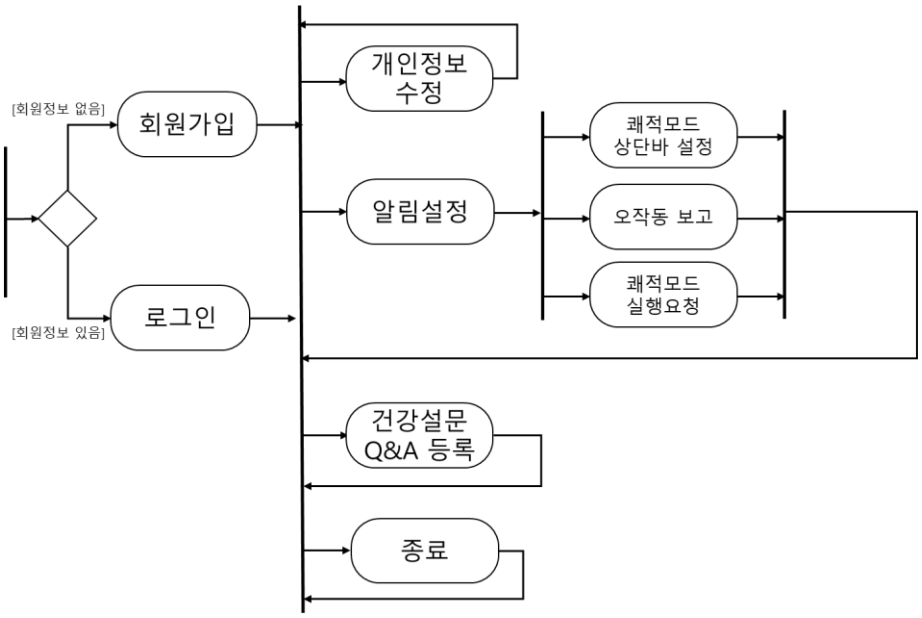


[Figure 2] Context Model

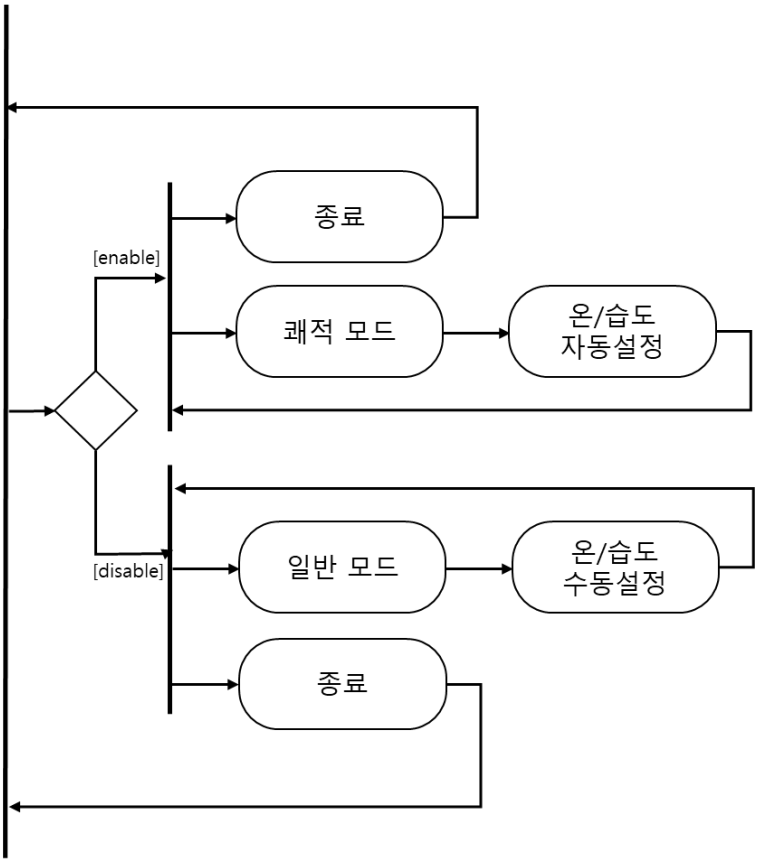
3.2.2 Sequence Diagram



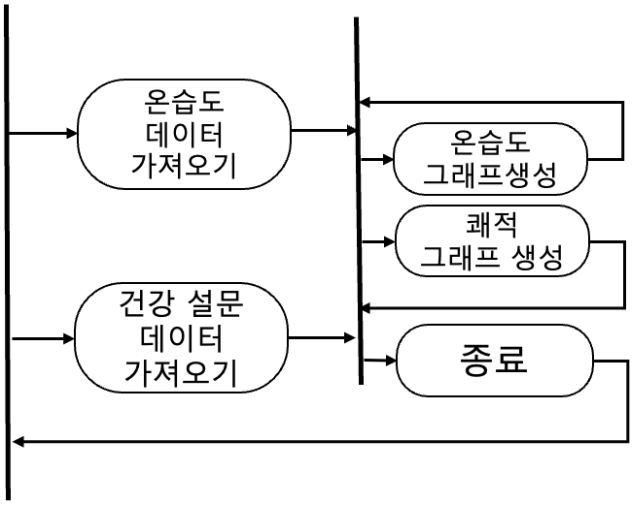
[Figure 3] 총괄



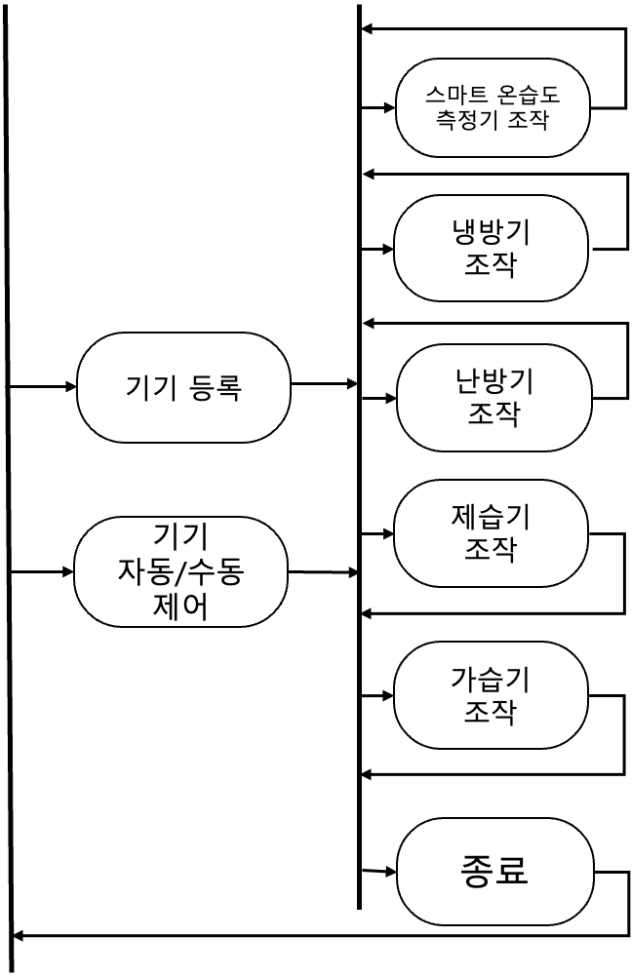
[Figure 4] 계정 관리



[Figure 5] 모드 설정

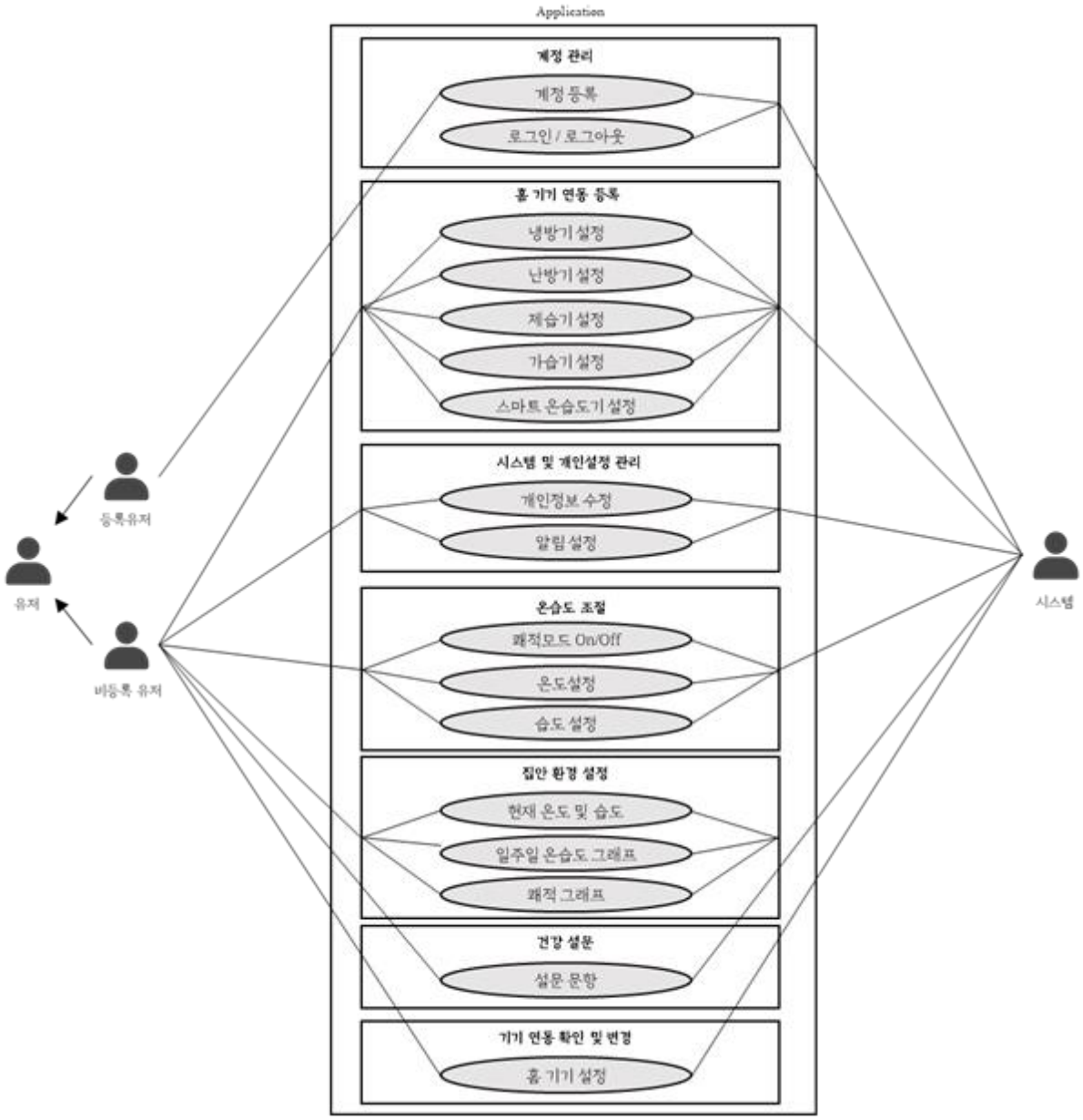


[Figure 6] 설문, 데이터 관리



[Figure 7] 기기 관리

3.2.3 Use Case Diagram



[Figure 8] Use Case Diagram

4. System Architecture – Frontend

4.1 Objectives

이 챕터에서는 Frontend에 해당하는 인터페이스 각각의 Attributes, Method, Class diagram, 그리고 Sequence diagram을 설명하고자 한다.

4.2 Subcomponents

4.2.1 로그인

로그인 Class에서는 전체 사용자의 로그인 세션과 계정을 관리한다.

(1) Attributes

로그인에서의 Attributes은 다음과 같다.

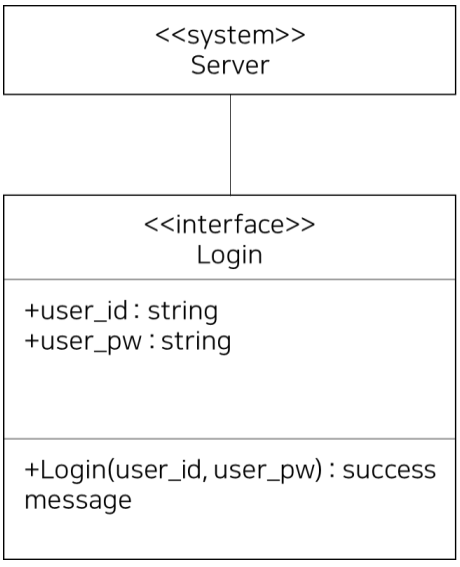
- user_id : 로그인 시 사용되는 사용자의 ID
- user_pw : 로그인 시 사용되는 사용자의 Password

(2) Method

로그인에서의 Method는 다음과 같다.

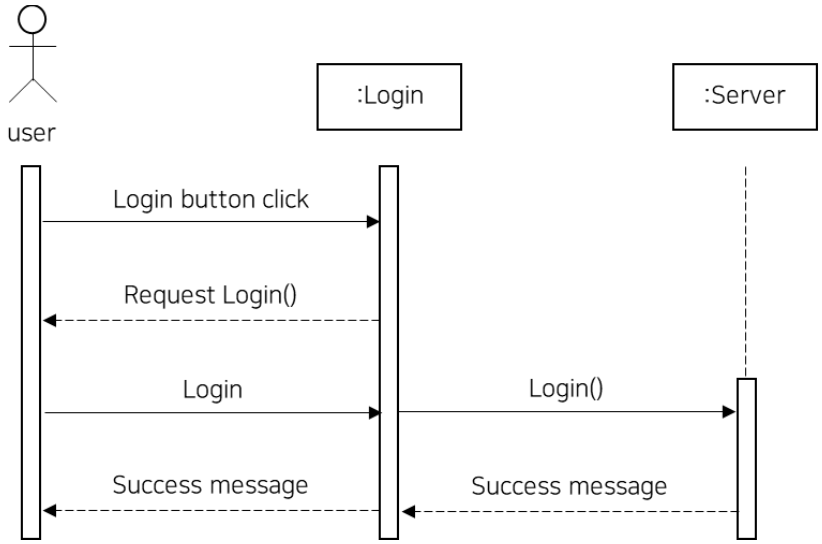
- Login()

(3) Class diagram



[Figure 9] 로그인 Class Diagram

(4) Sequence diagram



[Figure 10] 로그인 Sequence diagram

4.2.2 초기 홈기기 연동

초기 홈기기 연동 Class에서는 각 사용자의 홈기기 연동 설정 내용을 관리한다. 총 5개의 설정이 있으며, 냉방기, 난방기, 제습기, 가습기, 스마트 온습도 측정기 설정이다.

(1) Attributes

초기 홈기기 연동에서의 Attributes은 다음과 같다.

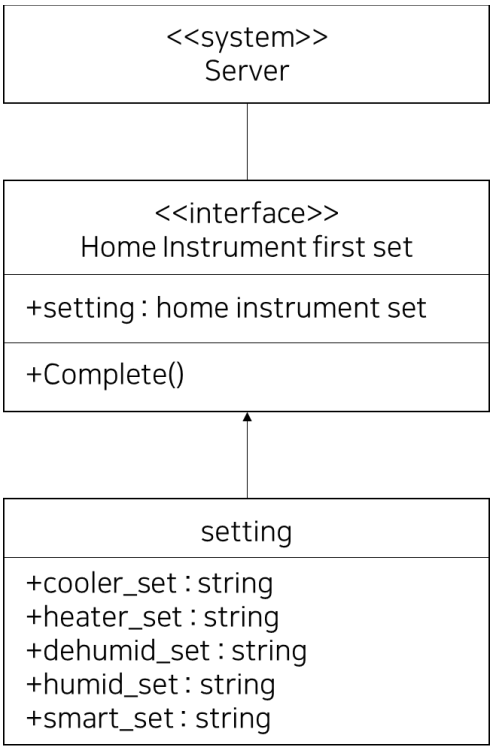
- setting: 초기 홈 기기 5가지 설정

(2) Method

Method는 다음과 같다. 홈기기 연동을 처음 설정하는 것이고, 한 번 설정하고 나면 이 화면은 나타나지 않기 때문에 set method만 있고, get method는 있지 않다. 모든 설정을 완료해주었을 때의 저장 버튼을 포함하고 있다.

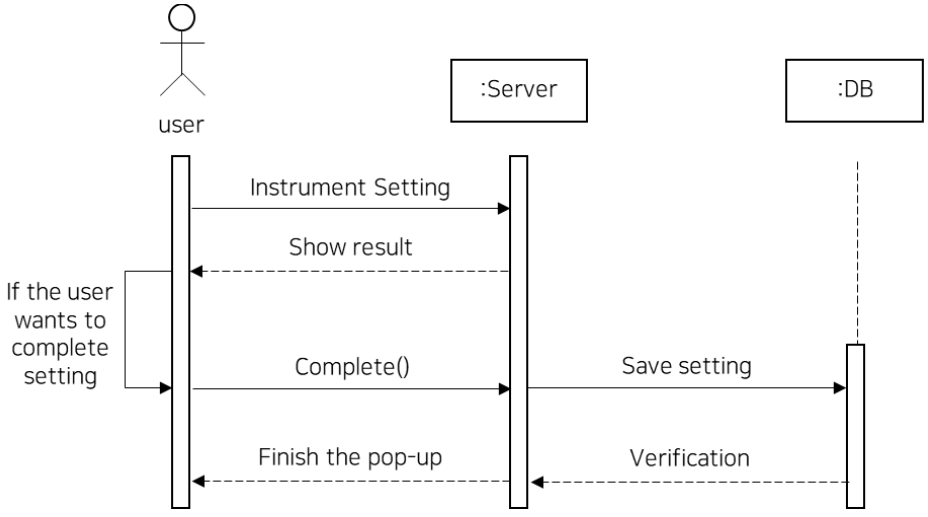
- Complete()

(3) Class diagram



[Figure 11] 초기 홈기기 연동 Class diagram

(4) Sequence diagram



[Figure 12] 초기 홈기기 연동 Sequence diagram

4.2.3 온습도 조절

온습도 조절 Class에서는 사용자의 설정에 따라 온습도를 조절하고 현재의 온습도를 확인한다. 사용자는 조절 장치를 자동/수동으로 맞춤 설정하여 쾌적모드 On/off 여부를 결정할 수 있다.

(1) Attributes

온습도 조절에서 사용되는 Attributes는 다음과 같다.

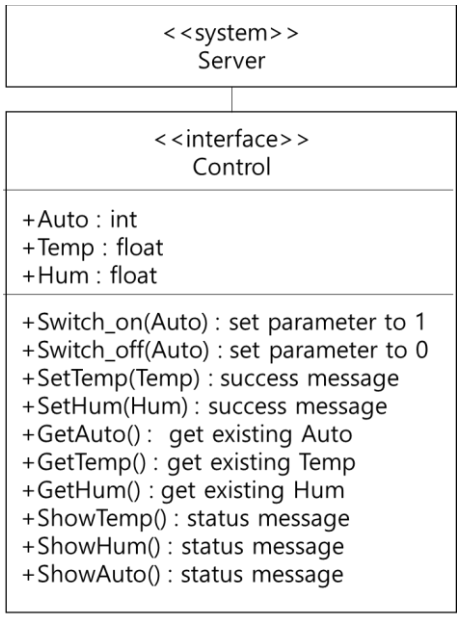
- Auto: 쾌적그래프 On,Off 설정값
- Temp: 온도 설정값
- Hum: 습도 설정값

(2) Methods

온습도 조절에서 사용되는 Methods는 다음과 같다.

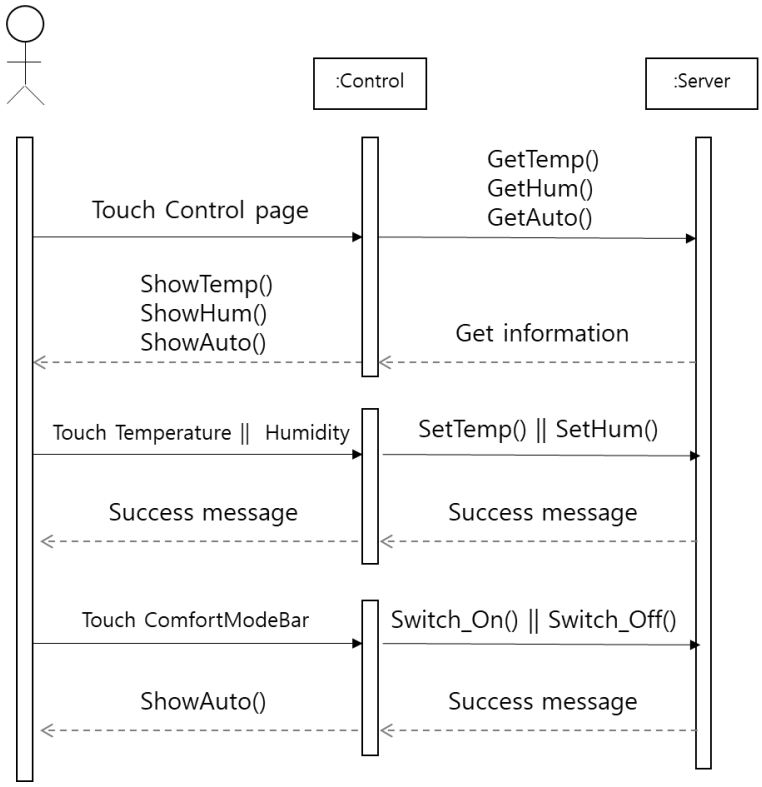
- Switch_On()
- Switch_Off()
- SetTemp()
- SetHum()
- ShowTemp()
- ShowHum()
- ShowAuto()
- GetTemp()
- GetHum()
- GetAuto()

(3) Class Diagram



[Figure 13] 온습도 조절 Class diagram

(4) Sequence Diagram



[Figure 14] 온습도 조절 Sequence diagram

4.2.4 집안 환경 설정

집안 환경 설정 Class에서는 사용자의 집안 상태를 전체적으로 확인할 수 있으며 최근 일주일 내 온습도 그래프와 쾌적그래프, 사용자가 업데이트한 건강 설문 데이터를 확인할 수 있다.

(1) Attributes

집안 환경설정에서 사용되는 Attributes는 다음과 같다.

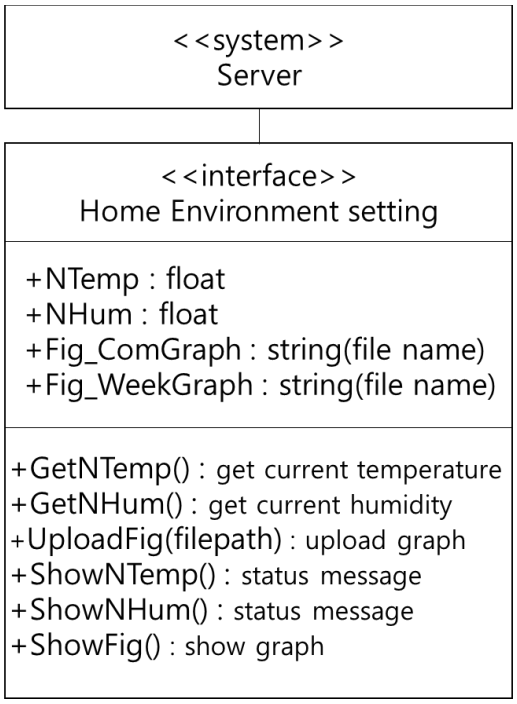
- NTemp: 현재 온도값
- NHum: 현재 습도값
- Fig_ComGraph: 쾌적그래프
- Fig_WeekGraph: 일주일 온습도 그래프

(2) Methods

집안 환경설정에서 사용되는 Methods는 다음과 같다.

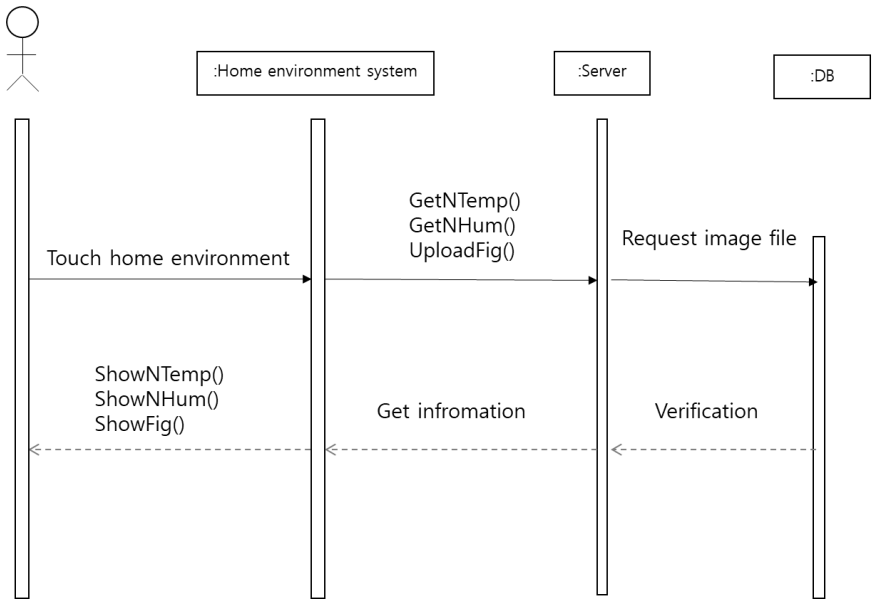
- GetNTemp()
- GetNHum()
- UploadFig()
- ShowNTemp()
- ShowNHum()
- ShowFig()

(3) Class Diagram



[Figure 15] 집안 환경 설정 Class diagram

(4) Sequence Diagram



[Figure 16] 집안 환경 설정 Sequence diagram

4.2.5 건강 설문

건강 설문 Class에서는 사용자가 설문을 통해 현재 건강 상태 등을 업로드 할 수 있다.

(1) Attributes

건강 설문에서 사용되는 Attributes는 다음과 같다.

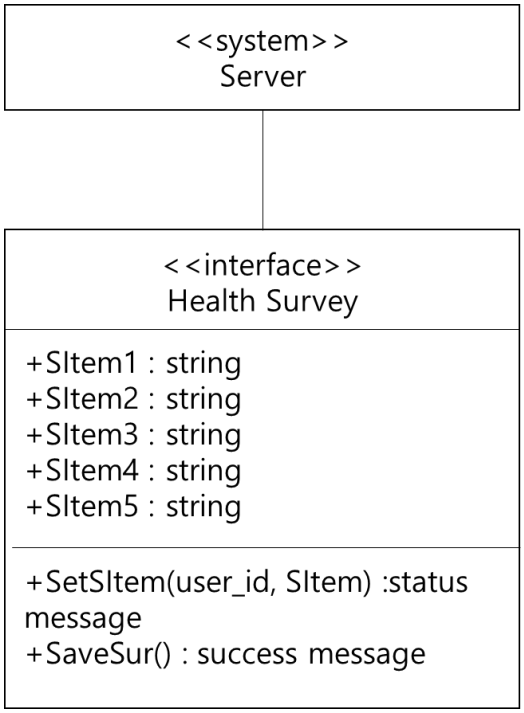
- SItem1: 문항 1번 응답값
- SItem2: 문항 2번 응답값
- SItem3: 문항 3번 응답값
- SItem4: 문항 4번 응답값
- SItem5: 문항 5번 응답값

(2) Methods

건강 설문에서 사용되는 Methods는 다음과 같다.

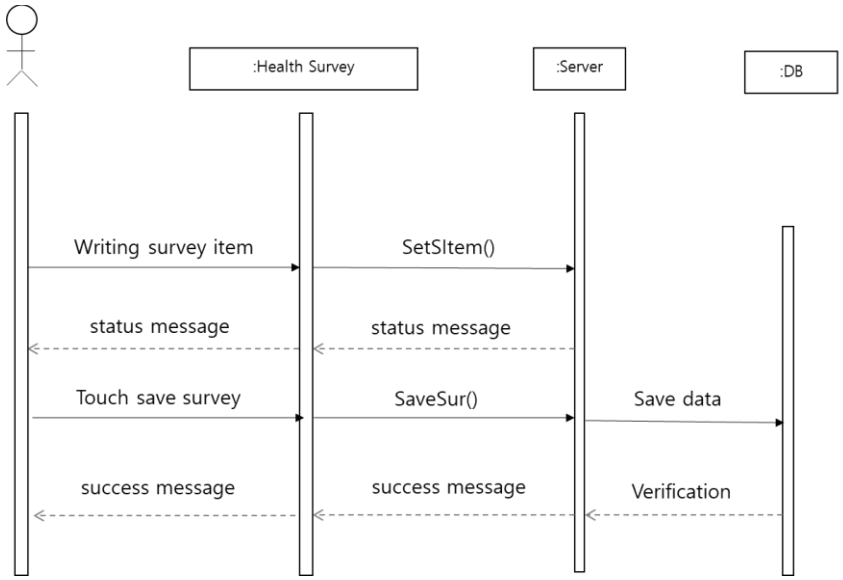
- SetSItem()
- SaveSur()

(3) Class Diagram



[Figure 17] 건강 설문 Class diagram

(4) Sequence Diagram



[Figure 18] 건강 설문 Sequence diagram

4.2.6 홈 기기 연동 확인 및 변경

홈 기기 연동 확인 Class에서는 초기 설정한 홈기기 연동 설정을 확인 및 변경할 수 있다.

(1) Attributes

홈 기기 연동 확인 및 변경에서 사용되는 Attributes는 다음과 같다.

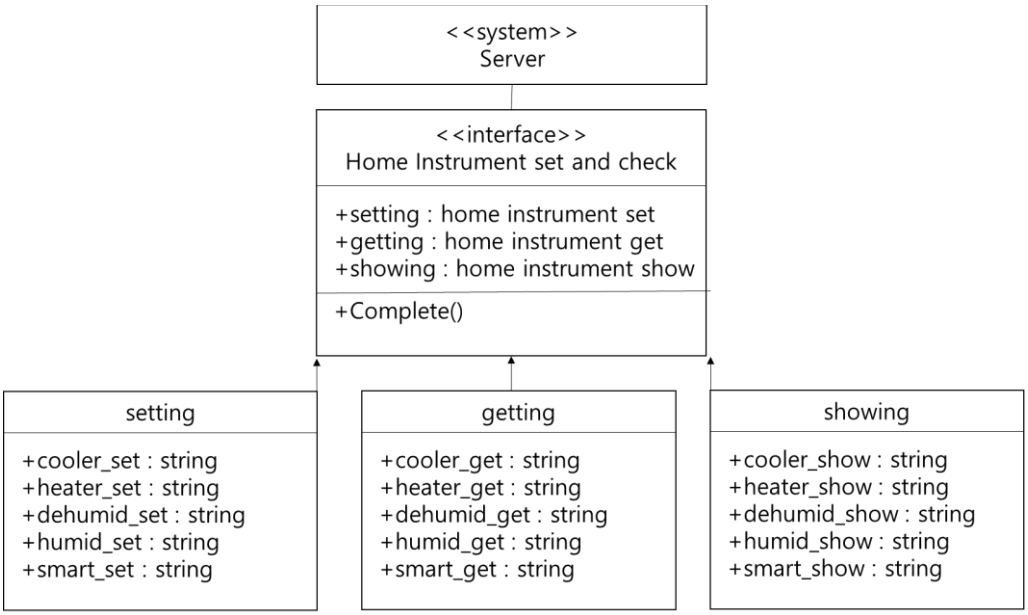
- getting: 현재 홈 기기 5가지 값 반환
- setting: 홈 기기 5가지 설정
- showing: 홈 기기 5가지 값 나열

(2) Methods

홈 기기 연동 확인 및 변경에서 사용되는 Methods는 다음과 같다.
같은 인터페이스와 기능을 사용하기 때문에 Attributes 와 Methods는 초기 홈기기 연동 설정 때 사용하였던 것과 동일하지만 초기화면에서 기존 설정값들을 보여주기 위해 반환과 보여주기 methods가 들어간다.

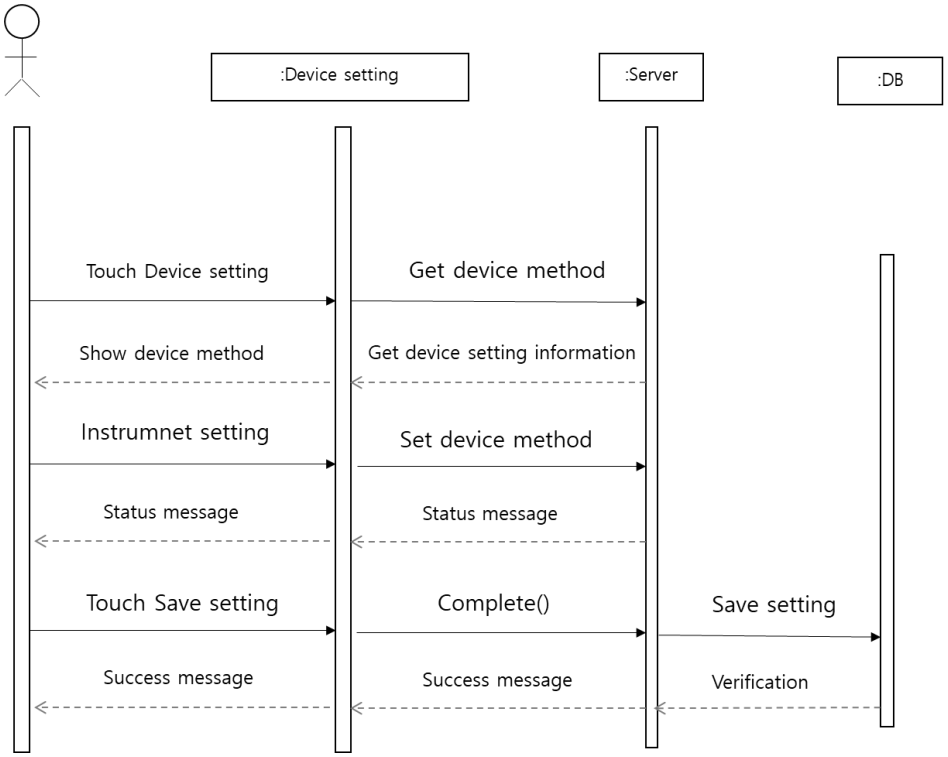
- set method
- get method
- show method
- Complete()

(3) Class Diagram



[Figure 19] 홈 기기 연동 확인 및 변경 Class diagram

(4) Sequence Diagram



[Figure 20] 홈 기기 연동 확인 및 변경 Sequence diagram

4.2.7 관리

관리 Class에서는 개인정보 및 개인 알림 설정을 관리할 수 있다.

(1) Attributes

관리에서 사용되는 Attributes는 다음과 같다.

- profile: 개인정보
- Top_bar: 상단바 on/off 설정값
- ErrorRequest: 오작동보고 on/off 설정값
- Auto: 쾌적모드 on/off 설정값

profile object에서 사용되는 Attributes는 다음과 같다.

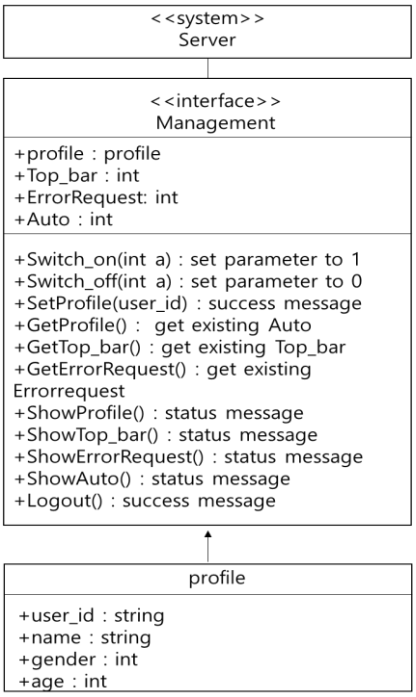
- user_id
- name
- gender
- age

(2) Methods

관리에서 사용되는 Methods는 다음과 같다.

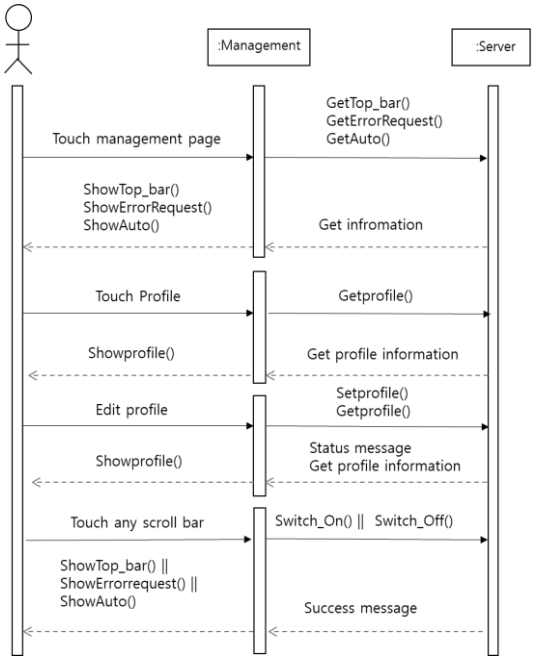
- Switch_On()
- Switch_Off()
- SetProfile()
- GetProfile()
- GetTop_bar()
- GetErrorRequest()
- GetAuto()
- ShowProfile()
- ShowTop_bar()
- ShowErrorRequest()
- ShowAuto()
- Logout()

(3) Class Diagram



[Figure 21] 관리 Class diagram

(4) Sequence Diagram



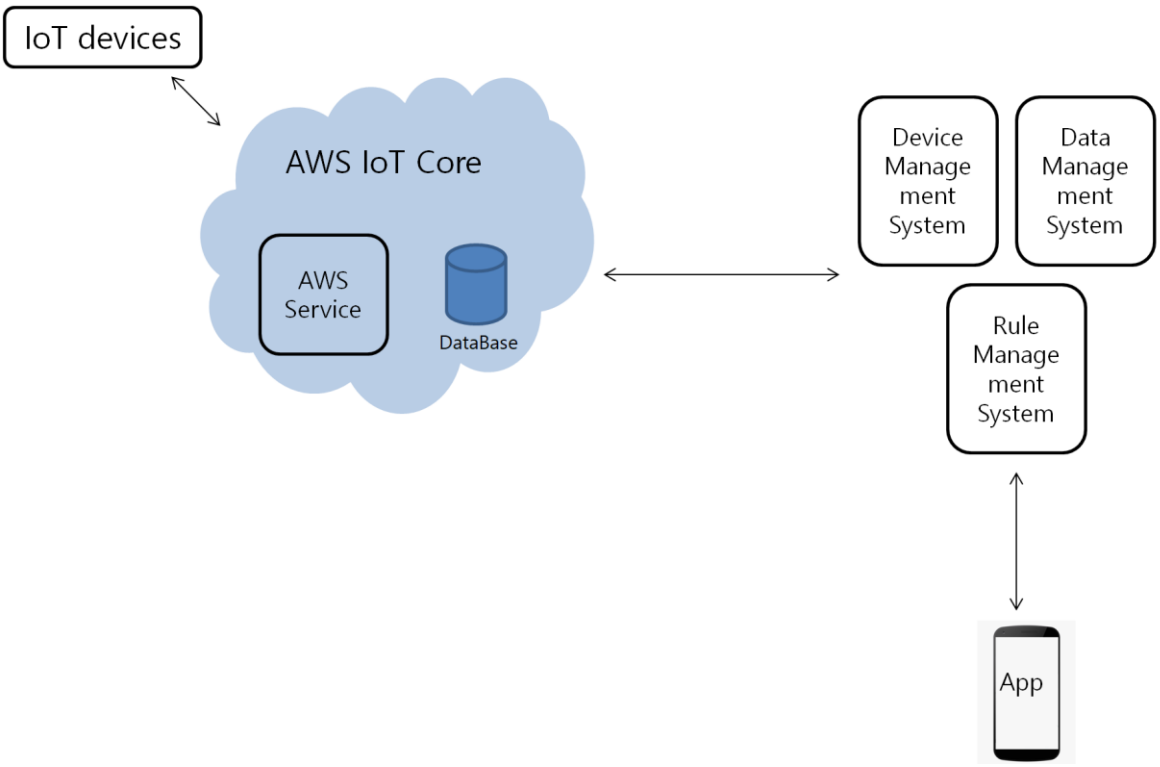
[Figure 22] 관리 Sequence diagram

5. System Architecture – Backend

5.1 Objectives

이 장에서는 클라우드 통신과 데이터베이스를 포함한 백엔드 시스템에 대한 구조를 서술한다.

5.2 Overall Architecture



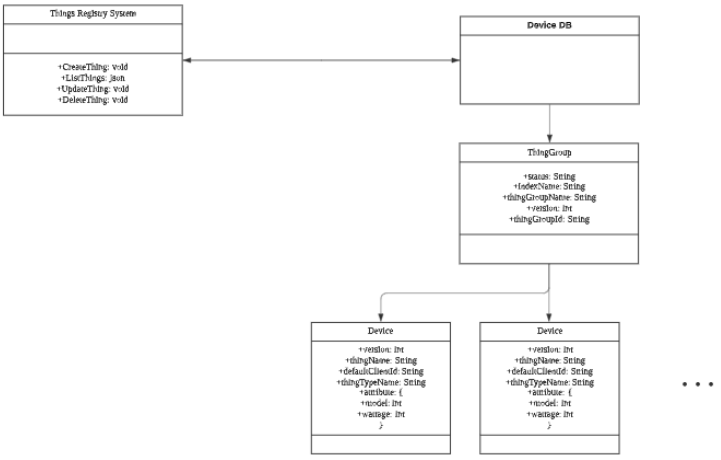
[Figure 23] Overall Architecture

전반적인 백엔드 아키텍처는 다음과 같다. 사용자는 앱을 통해 시스템 상으로 AWS 서비스에 요청을 보내거나 DB 상에서 데이터를 불러올 수 있다. AWS IOT Core는 클라우드 서비스로 IoT 디바이스와 애플리케이션 간 원활한 상호작용을 가능하게 한다. 디바이스 관리 시스템은 사용자가 IoT 디바이스를 등록, 수정, 또는 삭제할 수 있도록 한다. 데이터 관리 시스템은 일정 주기마다 사용자와 디바이스로부터 데이터를 받아 DB에 저장하고, 쾌적 지수를 도출하여 애플리케이션으로 데이터를 송신한다. 규칙 관리 시스템은 사용자가 AWS Lambda 서비스에 규칙을 추가하여 온습도 자동 모드 개인화 및 수동 조작이 가능하도록 한다.

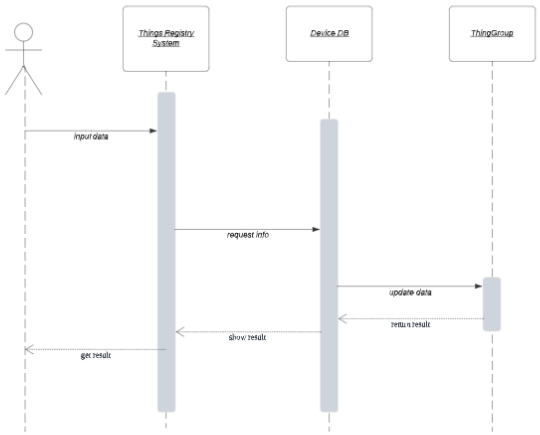
5.3 Subcomponents

5.3.1 Device Management System

사용자가 사물 레지스트리를 통하여 디바이스를 확인, 추가 및 삭제할 수 있다. ListThings 메소드를 통하여 계정에 존재하는 모든 디바이스를 불러올 수 있다. CreateThing, UpdateThing, DeleteThing을 통하여 각각 디바이스 추가, 갱신, 삭제 작업을 할 수 있다.



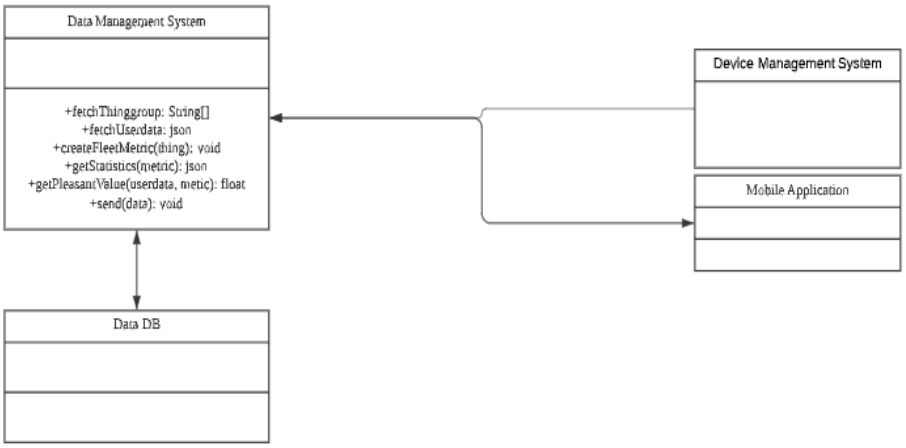
[Figure 24] Class Diagram - Device Management System



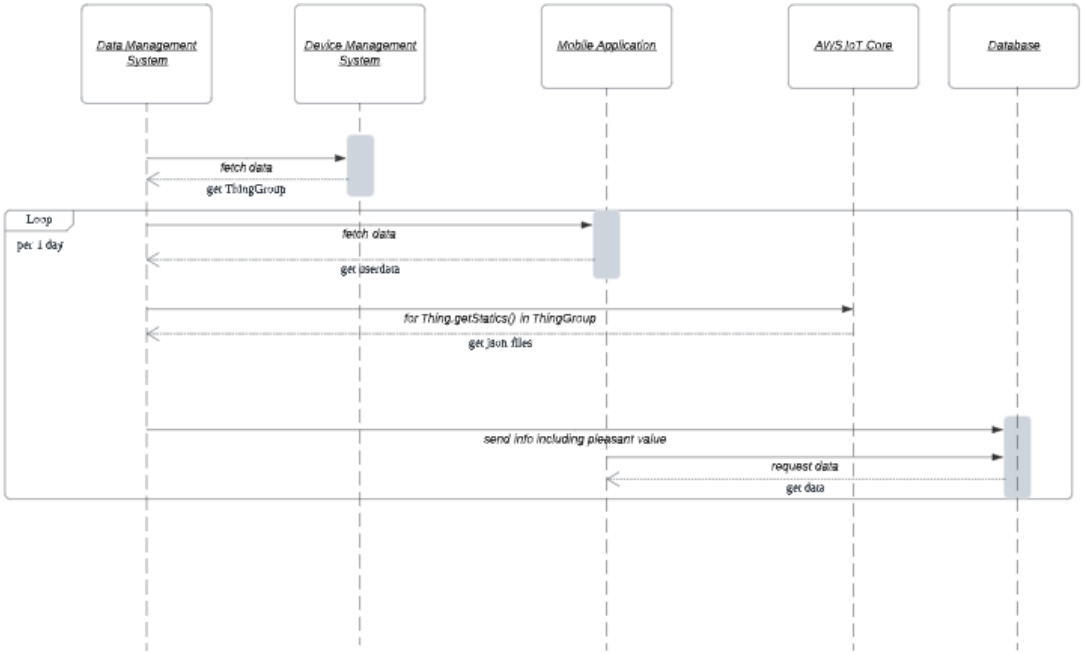
[Figure 25] Sequence Diagram - Device Management System

5.3.2 Data Management System

데이터 관리 시스템은 디바이스 관리 시스템으로부터 디바이스 리스트를 불러와 최초에 AWS IoT에서 제공하는 플릿 지표를 생성한다. 그 후 24시간 주기로 애플리케이션에서 사용자 데이터를, getStatics 메소드를 통하여 코어로부터 집계 데이터를 반환받는다. 반환받은 두 종류의 데이터를 통해 쾌적 지수를 도출하고, 데이터를 DB에 저장한다. 사용자는 애플리케이션을 통해 요청한 데이터를 받아올 수 있다.



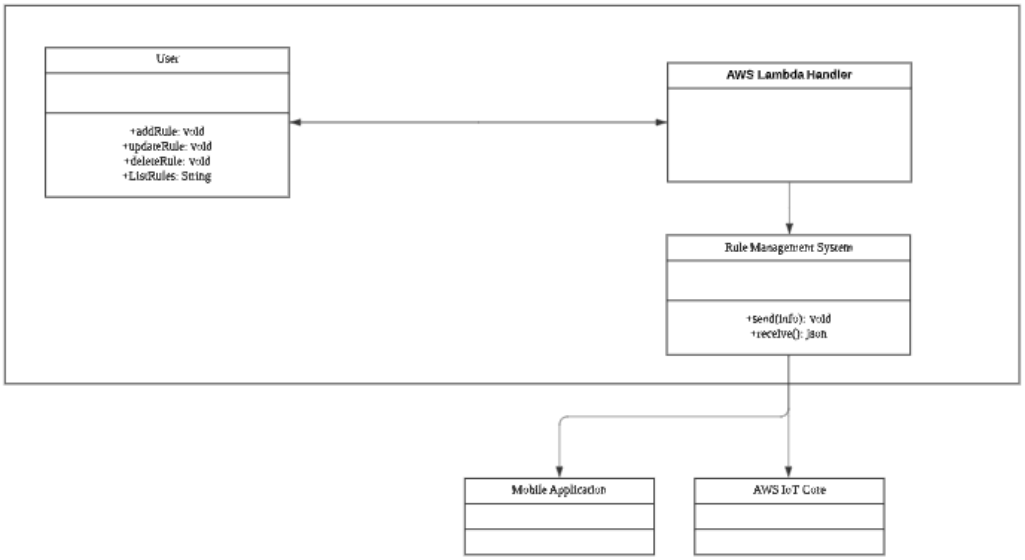
[Figure 26] Class Diagram - Data Management System



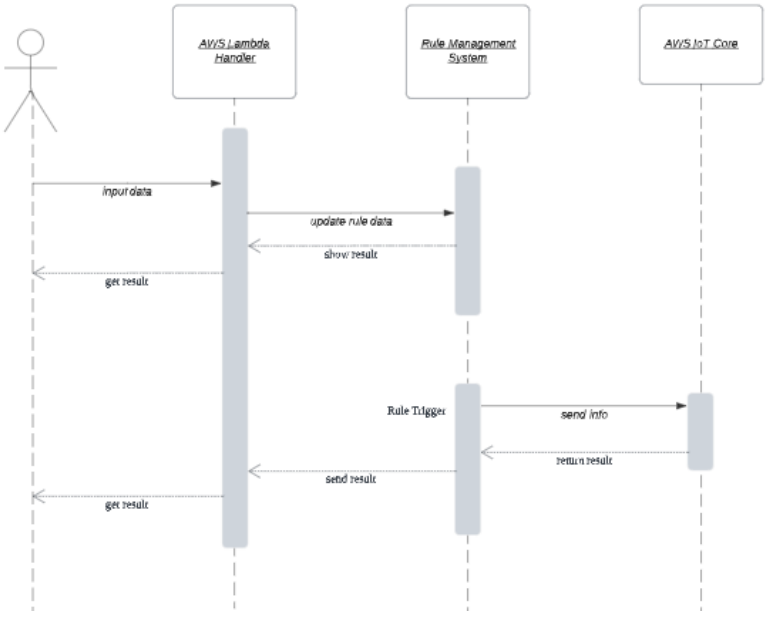
[Figure 27] Sequence Diagram - Data Management System

5.3.3 Rule management System

사용자는 각 디바이스의 작동 규칙을 AWS Lambda를 통해 갱신하거나 삭제할 수 있다. 디바이스의 작동 규칙은 초기에 애플리케이션에서 설정한 default 값으로 설정되어 있다. 사용자는 설정한 규칙을 애플리케이션을 통해 추가, 수정, 확인, 삭제할 수 있다. 설정한 규칙이 트리거된다면 코어로 데이터를 전송하여 디바이스가 규칙에 맞게 작동하도록 한다.



[Figure 28] Class Diagram - Rule Management System



[Figure 29] Sequence Diagram - Rule Management System

6. Protocol Design

6.1 Objectives

이 장에서는 클라이언트와 서버 간의 통신에 사용되는 여러 프로토콜 디자인에 대해 설명한다.

6.2 Rest API

RESTful API는 HTTP 요청을 사용하여 데이터에 액세스하고 사용하는 API(응용 프로그램 인터페이스)의 아키텍처 스타일이다. 해당 데이터는 리소스와 관련된 작업의 읽기, 업데이트, 생성 및 삭제를 나타내는 GET, PUT, POST 및 DELETE 데이터 유형에 사용할 수 있다.

REST는 클라우드 애플리케이션에서 유용하다. 상태 비저장 구성 요소는 문제가 발생하면 자유롭게 재배포할 수 있으며 부하 변경을 수용하도록 확장할 수 있다. 이는 모든 요청이 구성 요소의 모든 인스턴스로 전달될 수 있기 때문이다. 다음 트랜잭션에서 기억해야 하는 것은 저장될 수 없다. 따라서 웹 사용에 REST를 선호한다. API를 통해 서비스에 바인딩하는 것은 URL이 디코딩되는 방식을 제어하는 문제이기 때문에 RESTful 모델은 클라우드 서비스에서도 유용하다. 클라우드 컴퓨팅과 마이크로서비스는 미래에 RESTful API 설계를 규칙으로 만들 것이 거의 확실하다.

6.3 Details

6.3.1 회원가입

-Request

[Table 1] 회원가입 요청

Method	POST	
URL	/user/registration	
Parameter	ID	사용자 아이디
	Password	아이디 비밀번호
	Address	사용자의 주소
	Name	사용자의 이름
	Age	사용자의 나이

-Response

[Table 2] 회원가입 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: "Access Success"
Failure Response Body	Message	Message: "Access Fail"

6.3.2 로그인

-Request

[Table 3] 로그인 요청

Method	POST	
URL	/user/login	
Parameter	ID	사용자 아이디
	Password	아이디 비밀번호

-Response

[Table 4] 로그인 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: "Access Success"
Failure Response Body	Message	Message: "Access Fail"

6.3.3 로그아웃

-Request

[Table 5] 로그아웃 요청

Method	DELETE	
URL	/user/logout	
Parameter	ID	사용자 아이디
	Token	로그인 시 부여 받은 고유 키
	Time	로그아웃 시간

-Response

[Table 6] 로그아웃 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: "Access Success"
Failure Response Body	Message	Message: "Access Fail"

6.3.4 홈기기 연동

-Request

[Table 7] 홈기기 연동 요청

Method	POST	
URL	/device/homedeviceregistration	
Parameter	home_device_no	홈기기 고유 번호
	home_device_name	홈기기 이름
	ID	사용자 아이디

-Response

[Table 8] 홈기기 연동 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: "Access Success"
Failure Response Body	Message	Message: "Access Fail"

6.3.5 홈기기 해제

-Request

[Table 9] 홈기기 해제 요청

Method	POST	
URL	/device/ homedevicedelete	
Parameter	home_device_no	홈기기 고유 번호
	home_device_name	홈기기 이름
	ID	사용자 아이디

-Response

[Table 10] 홈기기 해제 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.6 온습도 기기 연결

-Request

[Table 11] 온습도 기기 연결 요청

Method	POST	
URL	/device/deviceregistration	
Parameter	device_no	기기 고유 번호
	device_name	기기 이름
	device_category	기기가 해당하는 종류(냉방기,난방기, 가습기, 제습기)
	ID	사용자 아이디

-Response

[Table 12] 온습도 기기 연결 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.7 온습도 기기 해제

-Request

[Table 13] 온습도 기기 해제 요청

Method	POST	
URL	/device/devicedelete	
Parameter	device_no	기기 고유 번호
	device_name	기기 이름
	device_category	기기가 해당하는 종류(냉방기,난방기, 가습기, 제습기)
	ID	사용자 아이디

-Response

[Table 14] 온습도 기기 해제 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.8 개별 온습도 기기 조절

-Request

[Table 15] 개별 온습도 기기 조절 요청

Method	POST	
URL	/selectivecontrol	
Parameter	device_category	기기가 해당하는 종류(냉방기,난방기, 가습기, 제습기)
	figure	설정한 온습도의 수치

-Response

[Table 16] 개별 온습도 기기 조절 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.9 자동 온습도 조절(쾌적 모드)

-Request

[Table 17] 자동 온습도 조절(쾌적 모드) 요청

Method	POST
URL	/autocontrol
Parameter	-

-Response

[Table 18] 자동 온습도 조절(쾌적 모드) 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.10 건강 설문 작성

-Request

[Table 19] 건강 설문 작성 요청

Method	POST	
URL	/user/currenthealth	
Parameter	Eye	사용자의 눈 건강 수치
	Neck	사용자의 목 건강 수치
	ID	사용자의 아이디

-Response

[Table 20] 건강 설문 작성 응답

Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: “Access Success”
Failure Response Body	Message	Message: “Access Fail”

6.3.11 쾌적 그래프

-Request

[Table 21] 쾌적 그래프 요청

Method	GET	
URL	/temhumgraph	
Parameter	ID	사용자의 아이디
	Time	설정한 기간

-Response

[Table 22] 쾌적 그래프 응답

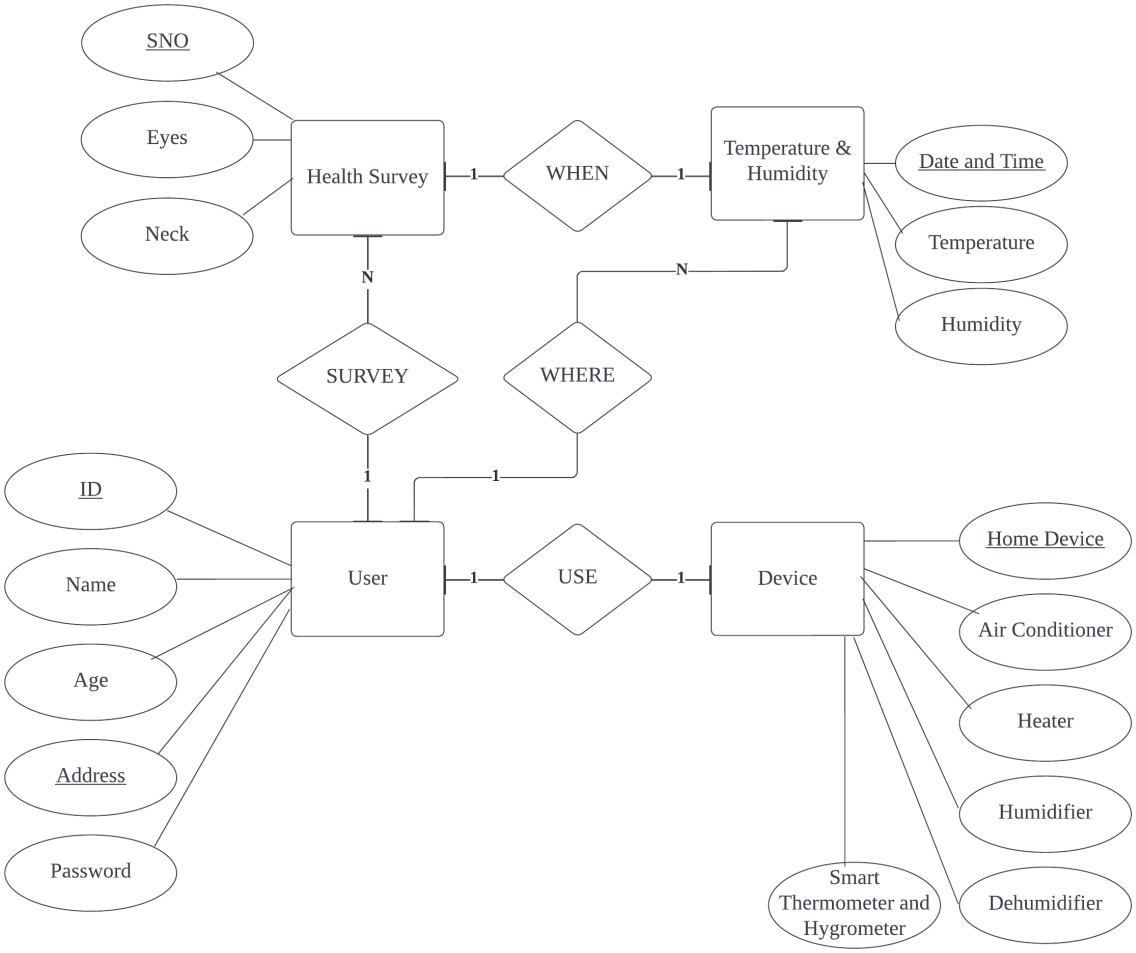
Success Code	200 OK	
Failure Code	400 Bad Request	
Success Response Body	Access Token	Token for access
	Message	Message: "Access Success"
	URL	쾌적 그래프 정보 URL
Failure Response Body	Message	Message: "Access Fail"

7. Database Design

7.1 Objectives

이 장에서는 ER diagram 과 SQL DDL을 사용해 데이터베이스 디자인에 대해 설명한다.

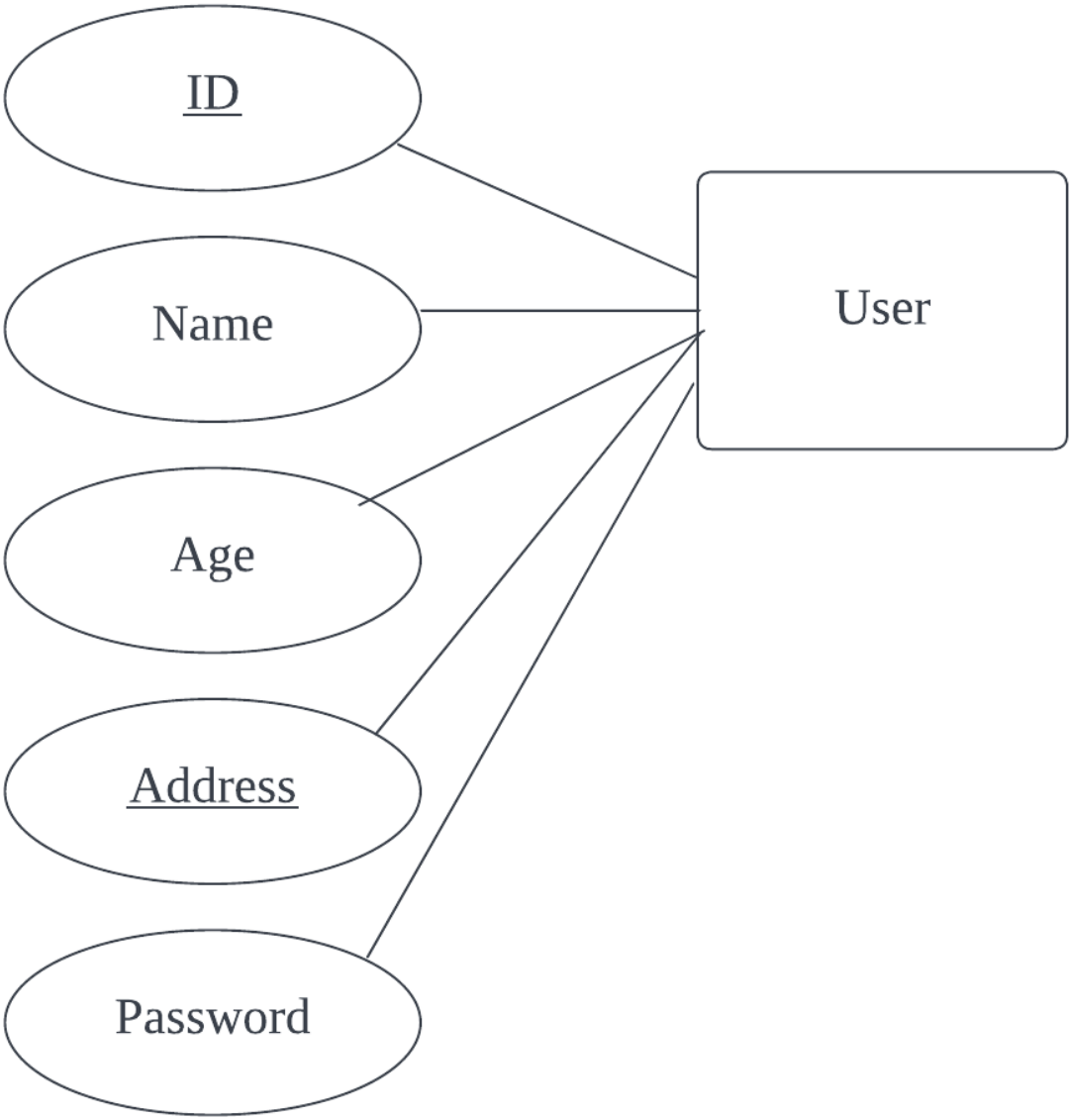
7.2 ER Diagram



[Figure 30] ER Diagram

본 시스템에서는 총 4개의 DB Entity가 존재한다. 각각 User, Device, Temperature & Humidity, Health Survey로 이루어져 있으며 각 Entity 사이의 관계는 아래의 ER 다이어그램으로 확인할 수 있다.

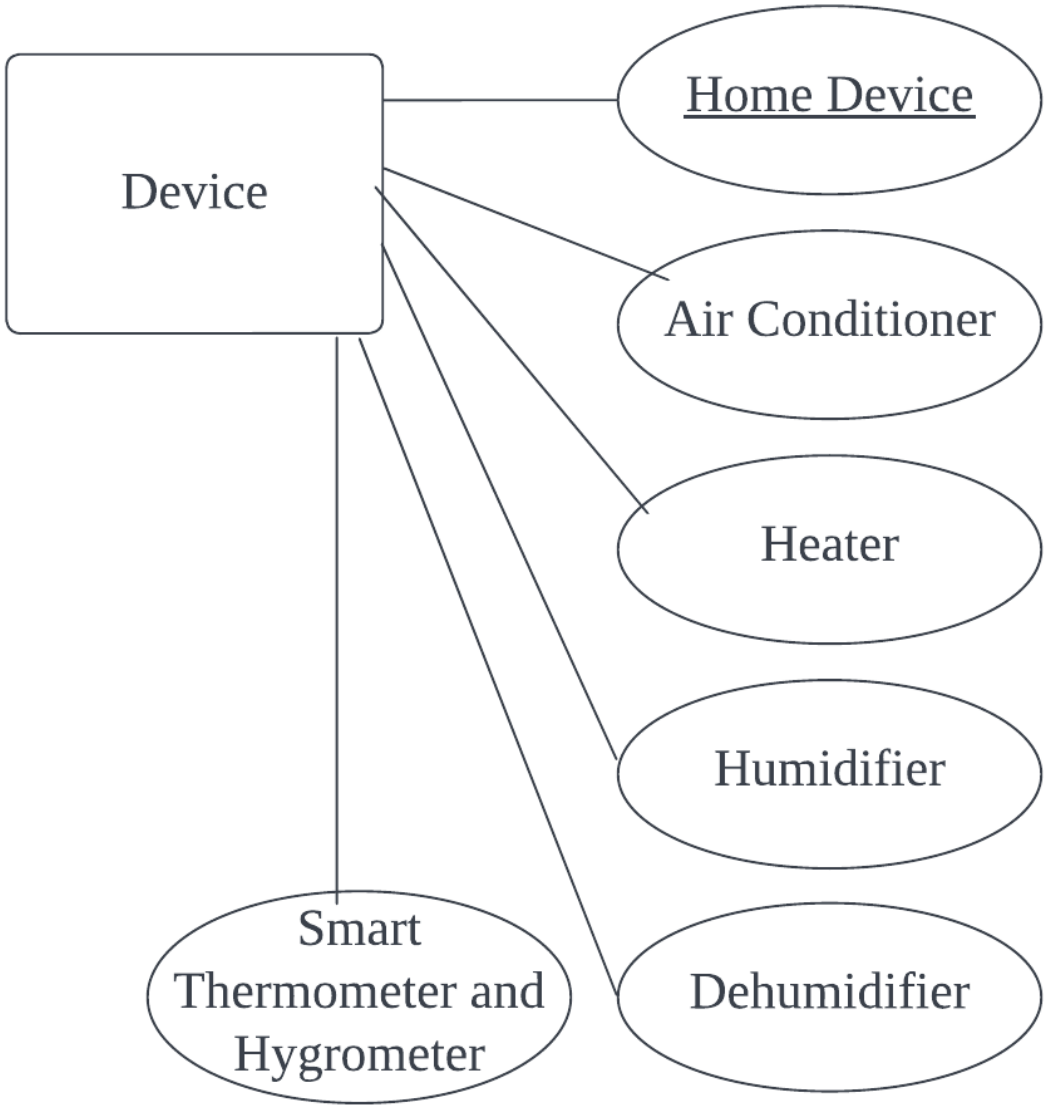
7.2.1 User



[Figure 31] User Entity

User Entity는 총 5개의 항목으로 구성되어 있다. ID, Name, Age, Address, Password가 그것이며 여기서 ID와 Address는 기본키이다.

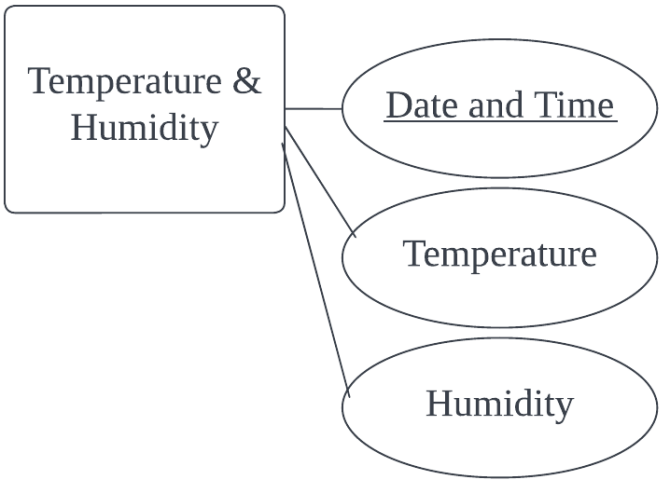
7.2.2 Device



[Figure 32] Device Entity

Device Entity는 프로그램에 등록되어 있는 기기들의 정보를 담고 있는 곳으로 홈기기, 냉방기, 난방기, 가습기, 제습기, 스마트 온습도 측정기로 이루어져 있다. 가장 기본적인 홈기기의 정보는 기본키 역할을 한다.

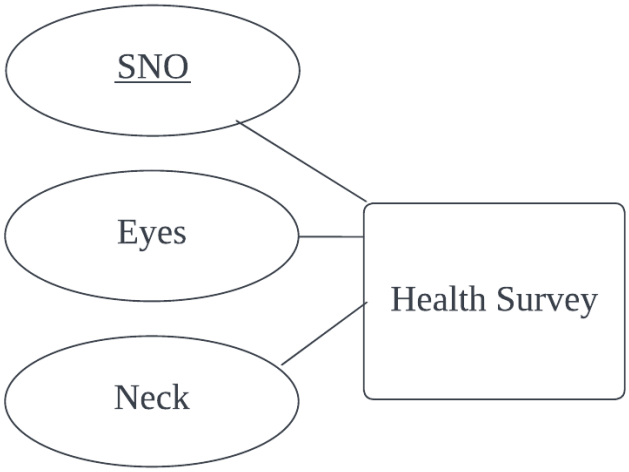
7.2.3 Temperature & Humidity



[Figure 33] Temperature & Humidity Entity

Temperature & Humidity Entity는 지난 시간 동안의 집안의 온도와 습도의 정보를 담고 있다. 날짜와 시간, 온도, 습도의 정보로 이루어져 있다. 여기서 Date and Time(YYYYMMDDHHMM) 즉 날짜와 시간은 기본키 역할을 할 것이다.

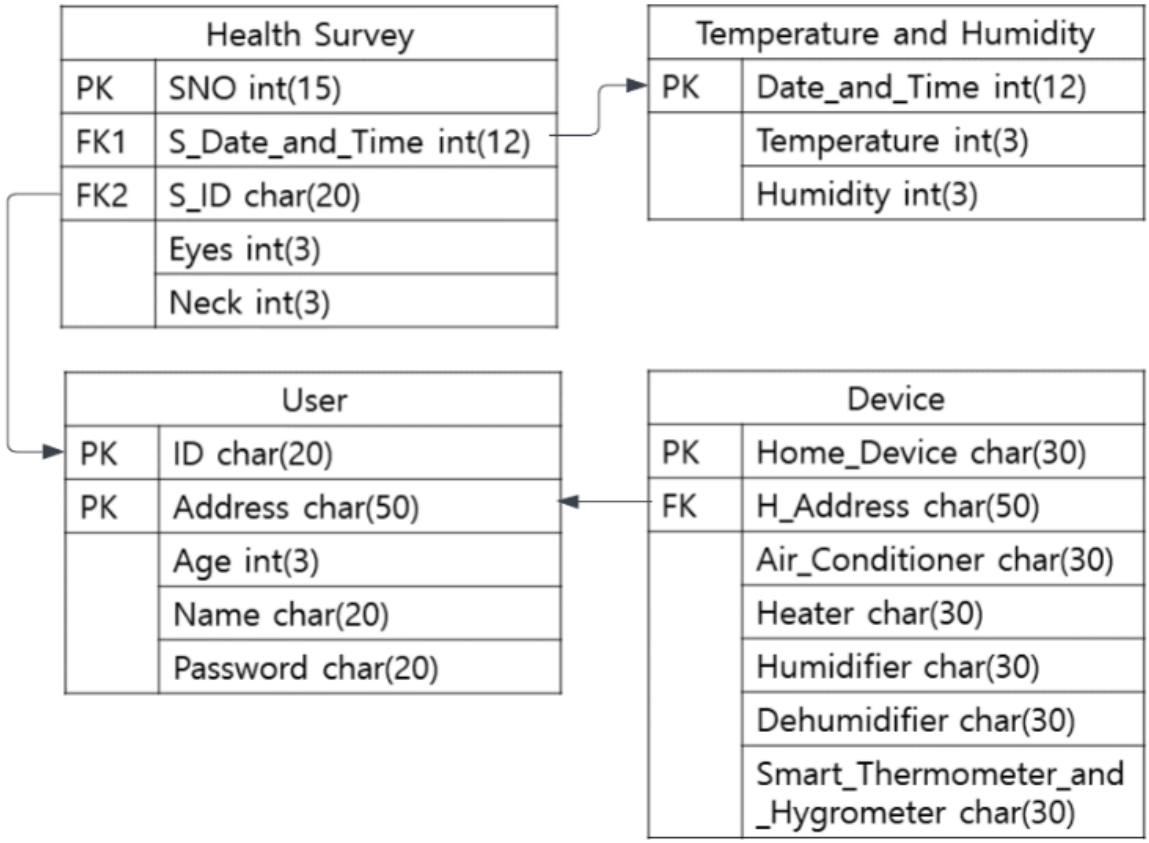
7.2.4 Health Survey



[Figure 34] Health Survey Entity

Health Survey Entity는 사용자로부터 건강 설문을 받은 정보를 담고 있다. 설문번호(SNO), 눈 건강 수치, 목 건강 수치의 정보를 담고 있다. 여기서 설문 번호는 기본키 역할을 할 것이다.

7.3 Relational Schema



[Figure 35] Relational Schema

7.4 SQL DDL

7.4.1 User

```
CREATE TABLE User
(
        ID char(20) NOT NULL,
        Address char(50) NOT NULL,
        Age int(3),
        Name char(20) NOT NULL,
        Password char(20) NOT NULL,

        PRIMARY KEY(ID)
);
```

7.4.2 Device

```
CREATE TABLE Device
(
        Home_Device char(30) NOT NULL,
        H_Address char(50) NOT NULL,
        Air_Conditioner char(30) NOT NULL,
        Heater char(30) NOT NULL,
        Humidifier char(30) NOT NULL,
        Dehumidifier char(30) NOT NULL,
        Smart_Thermometer_and_Hygrometer char(30) NOT NULL,

        PRIMARY KEY(Home_Device),
        FOREIGN KEY(H_Address) REFERENCES User(Address)
);
```

7.4.3 Temperature & Humidity

```
CREATE TABLE Temperature_and_Humidity
(
        Date_and_Time int(12) NOT NULL,
        Temperature int(3) NOT NULL,
        Humidity int(3) NOT NULL,

        PRIMARY KEY(Date_and_Time),
);
```

7.4.4 Health Survey

```
CREATE TABLE Health Survey
(
    SNO int(15) NOT NULL,
    S_Date_and_Time int(12) NOT NULL,
    S_ID char(20) NOT NULL,
    Eyes int(3),
    Neck int(3),

    PRIMARY KEY(SNO),
    FOREIGN KEY(S_Date_and_Time) REFERENCES Temperature_and_Humidity (Date_and_Time),
    FOREIGN KEY(S_ID) REFERENCES User(ID)
);
```

8. Testing Plan

8.1 Objectives

이 장에서는 이 시스템을 테스트하는 계획에 대해 설명한다.

8.2 Testing Policies

8.2.1 Development testing

개발 테스트는 소프트웨어 개발 위험, 시간 및 비용을 줄이기 위해 광범위한 결함 예방 및 감지 전략을 동기화하여 적용하는 소프트웨어 개발 프로세스이다. 소프트웨어 개발에 대한 조직의 기대에 따라 개발 테스트에는 정적 코드 분석, 데이터 흐름 분석, 메트릭 분석, 피어 코드 검토, 단위 테스트, 코드 적용 분석, 추적 가능성 및 기타 소프트웨어 검증 관행이 포함될 수 있다. 개발 테스트는 소프트웨어 개발 라이프사이클의 구축 단계에서 소프트웨어 개발자 또는 엔지니어가 수행합니다. 검증하는 분야는 세 가지인데 1) 성능 2) 안정성 3) 보안 이렇게 이루어져 있다

8.2.1.1 Performance

성능 테스트는 소프트웨어 테스팅의 한 항목으로써 시스템이 어떻게 작동하는지를 중점으로 테스트하는 것이다. 이 프로그램에서 가장 중요한 것은 자동 온습도 조절 기능(쾌적 모드)이다. 자동 온습도 조절 기능에서 확인해야 할 요소들은 다음과 같다. 1) 각 기기들이 정상적인 범위내에서 잘 가동하는지 확인 2)작동이 잘못되어 비효율적으로 운용되고 있지는 않은 지 확인 3)모든 기기들이 작동하는데 10초를 넘기지 않아야 한다. 기타로 스마트폰 프로그램을 실행하는데 10초 이상 걸려서는 안 된다. 개별 기기의 조작 명령을 실행하는데 5초 이상 걸려서는 안 된다

8.2.1.2 Reliability

안정성은 소프트웨어가 특정 환경에서 얼마나 오류 없이 실행이 가능한지를 보여주는 항목이다. 스마트 온습도 조절기는 스마트폰만 있는 것이 아닌 여러 기기들이 서로 연관성을 가지며 실행되는 프로그램이다. 그렇기에 이 프로그램이 오류가 나지 않기 위해서는 각 기기들이 서로 완벽하게 연결되어 있는 환경을 만들어주어야 할 것이다. 이와 마찬가지로 프로그램을 구성하는 여러 하위 구성요소와 서브 시스템이 잘 연결되어야 전체적인 안정성이 유지될 수 있다

8.2.1.3 Security

소프트웨어 보안은 각종 취약 요소로부터 프로그램을 평가하고 보호할 수 있는 기능을 말하는 것이다. 이 기능으로 프로그램은 외부로부터의 악성 공격으로부터 보호받을 수 있다. 스마트 온습도 조절기는 한 가정의 건강한 환경을 책임지는 프로그램이다. 그래서 집안 환경 조절 권한에 무단으로 접근하지 않도록 하는 것이 중요하다. 매 버전으로 업데이트 할 때마다 프로그램의 보안 요소를 확인하여 취약점이 존재하는지를 검증함으로써 보안성을 견고히 유지할 수 있다

8.2.2 Release Testing

Release Testing은 시스템의 특정 버전을 개발팀 외부에서 사용해 보도록 미리 릴리즈하는 과정이다. 이 과정의 목표는 사용자로 하여금 이 프로그램을 사용했을 때 믿고 사용할 수 있는 신뢰가 생기도록 하는 것이다. 테스트는 보통 모든 적용이 완료된 첫 버전을 최초로 사용하며 이를 통해 사용자의 반응을 확인한다. 첫번째 버전과 두 번째 버전을 비교하게 하여 약점을 보완하고 강점을 강조하도록 발전시킨다.

8.2.3 User Testing

User testing이란 사용자의 기능적 요구사항과 비기능적 요구사항을 충족하였는지 확인하는 것이다. 단, 본 시스템의 개발에 있어서 실제 유저의 테스트를 적용시키기는 어렵다. 필수적인 사용자 테스트를 진행할 수 있는 가능한 시나리오와 현실적인 상황을 설정해야 한다. 총 20 명의 프로그램 사용자가 있다고 가정한다. 그리고 스마트 온습도 조절기 베타 버전을 배포하고 사용자의 후기를 모은다

8.2.4 Testing Case

Testing Case는 성능, 안정성, 보안 이 세 가지를 검증하기 위해 설정할 것이다. 각 측면에 따라 5개의 테스트 케이스를 준비하고 전체 프로그램을 테스트한 다음, 평가지를 준비할 것이다

9. Developing Plan

9.1 Objectives

이 장에서는 시스템 개발을 위해 사용할 툴, 제약사항, 가정, 종속성에 대해 설명한다.

9.2 Frontend Environment

9.2.1 Android Studio

[Figure 36] Android Studio Logo

Android Studio는 안드로이드 및 안드로이드 전용 어플 제작을 위해 사용하는 통합 개발 환경으로, Java, C++, Kotlin을 사용하여 만들 수 있다. 본 프로젝트의 경우에는 Kotlin을 사용하여 개발한다. Android Studio를 사용하여 사용자가 직접 확인하는 UI 레이아웃과 애플리케이션 구조를 제작한다.

9.2.2 Adobe Photoshop

[Figure 37] Adobe Photoshop Logo

Adobe Photoshop은 레스터 그래픽 편집기로, 픽셀을 기본 단위로 사용하는 비트맵 방식의 툴이다. 애플리케이션의 로고 모양과 디자인을 제작한다.

9.3 Backend Environment

9.3.1. Github(오픈소스)

[Figure 38] Github Logo

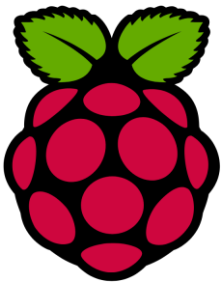
Github는 다수의 사람들이 함께 작업할 수 있는 원격 저장소를 지원해주는 호스팅 서비스이다. Github를 사용하여 어플리케이션 개발자들은 동시에 공동 작업할 수 있으며, 보다 효율적으로 개발에 대한 피드백을 공유할 수 있다. 또한 필요한 기능을 구현하기 위해 비슷한 기능을 제공하는 다른 어플리케이션 또는 웹을 개발하는 코드를 찾고 해당 앱 개발에 참고할 수 있다.

9.3.2. AWS IoT(DB, 서버)

[Figure 39] AWS IoT Logo

AWS IoT는 IoT 기반 애플리케이션을 지원하는 다양한 클라우드 서비스를 제공한다. AWS IoT Core는 디바이스가 클라우드 애플리케이션과 안전하게 상호작용할 수 있게 해주는 클라우드 서비스이다. AWS IoT Device SDK라는 개발 도구를 활용하여 AWS IoT와 Raspberry Pi가 내장된 디바이스를 연결하고 메시지를 주고 받을 수 있다. 이 개발 도구는 인터넷이 연결된 Raspberry Pi OS에 설치될 수 있다.

9.3.3 Raspberry Pi OS 32비트 이상



[Figure 40] Raspberry Pi OS Logo

Raspberry Pi OS는 초소형 컴퓨터인 Raspberry Pi의 운영체제 중 하나이다. 운영체제가 설치된 Raspberry Pi에는 Python으로 프로그래밍을 할 수 있다. Raspberry Pi는 프로그램에 따라 온습도 센서인 DHT11로 실내의 온도와 습도를 측정하고, 수집한 정보를 AWS IoT에 전달할 수 있다. 또한 Raspberry Pi는 냉난방기, 제습기, 가습기에 내장되어 기온조절 명령을 수행할 수 있다. 이를 통하여 사용자는 현재 집안의 기온 및 습도 상태를 알 수 있으며, 온습도 조절 및 쾌적 모드 실행할 수 있다.

9.4 Constraints

온습도 조절 시스템은 이 문서에 나타난 내용에 따라 디자인되고 만들어질 것이다. 자세한 디테일은 만들어지는 사람에 의해 결정되겠지만 아래의 내용들을 바탕으로 만들어질 것이다.

- 사용자가 보기 쉽고 편리하게 사용할 수 있도록 개발한다.
- 시스템이 사용자가 설정한 내용을 오차 없이 작동하도록 한다.
- 작동 시간을 줄이고, 코드가 최적화되도록 개발한다.
- 소스 코드에 주석을 달아 변경이 용이하도록 한다.
- 증명된 기술을 사용하고자 한다.
- 성능이 높아질 수 있는 방향을 고안한다.
- 시스템을 만드는데 사용되는 비용과 최대 효율을 높일 수 있도록 한다.

9.5 Assumptions and Dependencies

이 문서의 모든 시스템은 Raspberry Pi OS 및 AWS IoT 클라우드 서비스를 기반으로 디자인 및 구현되었다. 타 시스템에서 이 개발 디자인을 적용할 시 정상적으로 작동하지 않을 가능성이 있다.

10. Supporting Information

10.1 Software Design Specification

이 요구사항 명세서는 IEEE Recommendation (IEEE Recommended Practice for Software Requirements Specifications, IEEE-Std-830). 서식을 따라 제작되었다.

10.2 Document History

[Table 23] Document History

날짜	내용	이름
05/06	문서 작성 시작	서현기
05/10	5-3	신영섭
5/10	1, 2, 8	서현기
5/10	4	안예림
5/10	4	박종찬
5/10	3, 9-3	황선우
5/11	8.2.1	서현기
5/13	4	박종찬
5/14	9-3	황선우
5/14	6, 7	서현기
5/14	9-2, 9-4	안예림
5/14	5, 9-5	신영섭
5/14	4	박종찬
5/15	명세서 다듬기	서현기