



스마트 불쾌지수 조절 시스템

Software Design Specification

2022.05.15.

Introduction to Software Engineering

TEAM 4

Team Leader
Team Member
Team Member
Team Member

고귀환
김민성
오하은
임준우

1. Preface	12
1.1. Readership	12
1.2. Scope	12
1.3. Objective	12
1.4. Document Structure	13
2. Introduction	14
2.1 Objectives	14
2.2 Applied Diagrams	14
2.2.1 UML	14
2.2.2 Use case Diagram	14
2.2.3 Sequence Diagram	15
2.2.4 Class Diagram	15
2.2.5 Context Diagram	15
2.2.6 Entity Relationship Diagram	16
2.3 Applied Tools	16
2.3.1 Microsoft Word	16
2.3.2 Microsoft PowerPoint	16
2.3.3 Google Docs	17
2.3.4 draw.io	17
2.4 Project Scope	17
2.5 References	18
3. System Architecture - Overall	18
3.1. Objectives	18
3.2. System Organization	18
3.2.1 Context Diagram	19
3.2.2 Sequence Diagram	20

3.2.3. Use Case Diagram	21
4. System Architecture - Frontend	23
4.1. Objectives	23
4.2. Subcomponents	23
4.2.1. 로그인 및 회원가입	23
4.2.1.1. Attributes	23
4.2.1.2 Methods	23
4.2.1.3 Class Diagram	24
4.2.1.4 Sequence Diagram	25
4.2.2. 기기 전원 켜기/끄기	25
4.2.2.1. Attributes	25
4.2.2.2 Methods	25
4.2.2.3 Class Diagram	26
4.2.2.4 Sequence Diagram	27
4.2.3. 온도/습도 설정	27
4.2.3.1. Attributes	27
4.2.3.2 Methods	28
4.2.3.3 Class Diagram	28
4.2.3.4 Sequence Diagram	29
4.2.4. 비밀번호 분실	29
4.2.4.1. Attributes	29
4.2.4.2 Methods	30
4.2.4.3 Class Diagram	30
4.2.4.4 Sequence Diagram	31
4.2.5. 기기 및 장소 추가	31
4.2.5.1. Attributes	31
4.2.5.2 Methods	32

4.2.5.3 Class Diagram	32
4.2.5.4 Sequence Diagram	33
4.2.6. 사용자 피드백	34
4.2.6.1. Attributes	34
4.2.6.2 Methods	35
4.2.6.3 Class Diagram	35
4.2.6.4 Sequence Diagram	36
5. System Architecture - Backend	37
5.1. Objectives	37
5.2. Overall Architecture	37
5.2.1. 저장소(Repository)	39
5.2.2. Systems	39
5.3. Subcomponents	40
5.3.1. 로그인 / 회원가입 시스템	40
5.3.2. 스마트 위치를 통한 사용자 상태 데이터 수집 시스템	41
5.3.3. 사물인터넷을 통한 사용자 활동 데이터 수집 시스템	42
5.3.4. 실외 / 실내 환경 수집 시스템	43
5.3.5. 기기 조절 시스템	44
5.3.6. 머신러닝 시스템	44
5.3.7. 피드백 시스템	46
6. Protocol Design	48
6.1. Objectives	48
6.2. HTTP	48
6.3. 로그인 및 회원가입	48
6.3.1. 회원가입	48
6.3.2. 로그인	49

6.4. 비밀번호 찾기	50
6.5. 기기 전원 켜기 / 끄기	51
6.6. 온도/습도 설정	52
6.7. 기기/장소 추가	53
6.7.1. 기기 추가	53
6.7.2. 장소 추가	54
6.8. 목록 불러오기	55
6.8.1. 기기 목록 불러오기	55
6.8.2. 장소 목록 불러오기	56
6.9. 피드백 작성	57
7. Database Design	58
7.1. Objectives	58
7.2. ER Diagram	58
7.2.1 Entities	59
7.2.1.1. User	59
7.2.1.2. Smart watch	60
7.2.1.4. App service	61
7.2.1.5. Measuring instruments	62
7.2.1.6. Meteorological administration API	62
7.2.1.7. Machine learning	63
7.2.1.8. Smart local devices	63
7.2.1.9. Data base	64
7.3. Relational Schema	65
7.4. SQL DDL	66
7.4.1 User	66
7.4.2 Smart_watch	66
7.4.3 IoT	67

7.4.4 App_service	67
7.4.5 Measuring_instruments	68
7.4.6 Meteorological_administration_API	68
7.4.7 Machine_learning	69
7.4.8 Smart_local_device	69
7.4.9 Date_base	69
8. Testing Plan	71
8.1. Objectives	71
8.2. Testing Policy	71
8.2.1. Development Testing	71
8.2.1.1. Performance	71
8.2.1.2. Reliability	72
8.2.1.3. Security	72
8.2.2. Release Testing	72
8.2.3. User Testing	73
8.2.4. Testing Case	73
9. Development Plan	74
9.1. Objectives	74
9.2. Frontend Environment	74
9.2.1. Adobe Photoshop (UI/UX Design)	74
9.2.2. Adobe Xd (UI/UX Design)	74
9.2.3. Flutter	75
9.3. Backend Environment	76
9.3.1. Github	76
9.3.2. Visual Studio Code	76
9.3.3. Tensorflow	77

9.3.4. MySQL Workbench	77
9.3.5. Spring Boot	78
9.4. Constraints	78
9.5. Assumptions and Dependencies	79
10. Supporting Information	79
10.1. Software Design Specification	79
10.2. Document History	80

LIST OF FIGURES

[Figure 1] Overall system architecture	19
[Figure 2] Overall context diagram	19
[Figure 3] 가입 / 로그인 sequence diagram	20
[Figure 4] 유저 피드백 반영 sequence diagram	21
[Figure 5] Use case diagram	22
[Figure 6] Class diagram - Profile	24
[Figure 7] Sequence diagram - Profile	25
[Figure 8] Class diagram - Power control	26
[Figure 9] Sequence diagram - Power control	27
[Figure 10] Class diagram - Air condition	28
[Figure 11] Sequence diagram - Air condition	29
[Figure 12] Class diagram - Password Find	30
[Figure 13] Sequence diagram - Password Find	31
[Figure 14] Class diagram - Add Place/Product	32
[Figure 15] Sequence diagram - Add Place/Product	33
[Figure 16] Class diagram - User Feedback	35
[Figure 17] Sequence diagram - User Feedback	36
[Figure 18] Overall architecture	38
[Figure 19] Class Diagram - Login / Register	40
[Figure 20] Class Diagram - User Condition information	41

[Figure 21] Class Diagram - User Activity information	42
[Figure 22] Class diagram - 실외 / 실내 환경 수집 시스템	43
[Figure 23] Class diagram - 실외 / 실내 환경 수집 시스템	44
[Figure 24] Class diagram - Machine learning	45
[Figure 25] Class diagram - 피드백 시스템	46
[Figure 26] Sequence diagram - 피드백 시스템	47
[Figure 27] ER-Diagram	58
[Figure 28] ER diagram, Entity, User	59
[Figure 29] ER diagram, Entity, Smart watch	60
[Figure 30] ER diagram, Entity, IoT	60
[Figure 31] ER diagram, Entity, App service	61
[Figure 32] ER diagram, Entity, Measuring instruments	62
[Figure 33] ER diagram, Entity, Meteorological administration API	62
[Figure 34] ER diagram, Entity, Machine learning	63
[Figure 35] ER diagram, Entity, Smart local devices	63
[Figure 36] ER diagram, Entity, Data base	64
[Figure 37] Relational Schema	65
[Figure 38] Adobe Photoshop logo	74
[Figure 39] Adobe Xd logo	74
[Figure 40] Flutter logo	75
[Figure 41] Github logo	76

[Figure 42] Visual Studio Code logo	76
[Figure 43] TensorFlow logo	77
[Figure 44] MySQL Workbench logo	77
[Figure 45] Spring Boot logo	78
[Figure 46] Document History	80

LIST OF TABLES

[Table 1] register request	48
[Table 2] register response	48
[Table 3] Log-in request	49
[Table 4] Log-in response	49
[Table 5] Find Password request	50
[Table 6] Find Password response	50
[Table 7] Power control request	51
[Table 8] Power control response	51
[Table 9] Air conditioning request	52
[Table 10] Air conditioning response	52
[Table 11] Add Product request	53
[Table 12] Add Product response	53
[Table 13] Add Place request	54
[Table 14] Add Place response	54
[Table 15] Load Product List request	55
[Table 16] Load Product List response	55
[Table 17] Load Place List request	56
[Table 18] Load Place List response	56
[Table 19] Send Feedback request	57
[Table 20] Send Feedback response	57

1. Preface

이 챕터에서는 “스마트 불쾌지수 조절 시스템” 소프트웨어 디자인 명세서의 독자들을 위한 문서의 간략한 정보, 개요, 목적, 그리고 목차를 소개한다.

1.1. Readership

본 문서는 “스마트 불쾌지수 조절 시스템”의 소프트웨어 디자인 명세서로서 10개의 부분으로 나뉘어진다. 문서의 각 부분은 해당 시스템의 여러 부분들을 목차별로 나누어 상세히 기술한다. 자세한 목차는 1.4에서 확인할 수 있다. 본 문서의 주요 독자들은 소프트웨어공학(AAI3007_01)의 Team 4(고귀환, 김민성, 오하은, 임준우), 교수, 학생, 그리고 “스마트 불쾌지수 조절 시스템”的 사용자이다.

1.2. Scope

본 문서는 “스마트 불쾌지수 조절 시스템”을 개발하기 위한 디자인 정의로써 소프트웨어 공학(Software Engineering)과 소프트웨어 품질 공학(Software Quality Engineering)에 쓰일 예정이다.

1.3. Objective

본 디자인 명세서의 주요 목적은 “스마트 불쾌지수 조절 시스템”과 해당 애플리케이션에 필요한 기술적 디자인 요소들을 서술하는 것이다. 따라서 본 문서는 앞서 작성된 요구사항 명세서의 내용들을 구체화하여 “스마트 불쾌지수 조절 시스템”을 개발하는데 필요한 소프트웨어 아키텍쳐들과 여러 디자인 요소들을 상세하게 기술한다. 해당 내용들을 통해 “스마트 불쾌지수 조절 시스템”과 그 애플리케이션에 대한 대략적인 조망이 가능하다. 본 문서는 시스템 아키텍처의 frontend, backend, 프로토콜 디자인, 데이터베이스 디자인, 시험 계획, 개발과정들을 서술한다.

1.4. Document Structure

- 1. Preface: 이 챕터는 문서의 소개, 용도, 그리고 문서의 작성 목적과 목차를 기술한다.
- 2. Introduction: 이 챕터는 시스템을 소개하고, 본 문서를 작성하는 데 사용된 도표들과 도구들에 대한 설명을 기술한다.
- 3. Overall System Architecture: 이 챕터는 시스템의 전체적인 구조를 보여주는 여러 아키텍쳐들과 도표들을 기술한다.
- 4. System Architecture - Frontend: 이 챕터는 본 시스템의 Frontend 부분을 기술한다.
- 5. System Architecture - Backend: 이 챕터는 본 시스템의 Backend 부분을 기술한다.
- 6. Protocol Design: 이 챕터는 사용자와 서버 간의 통신에 사용되는 여러 프로토콜들을 기술한다.
- 7. Database Design: 이 챕터는 본 시스템에 사용된 데이터베이스에 대해 기술한다.
- 8. Testing Plan: 이 챕터는 본 시스템의 시험 계획을 기술한다.
- 9. Development Plan: 이 챕터는 본 시스템과 애플리케이션 개발에 필요한 여러 도구들과 개발 계획을 기술한다.
- 10. Supporting Information: 이 챕터는 본 디자인 명세서를 작성하는데 참고한 양식과 문서 작성 기록을 기술한다.

2. Introduction

스마트 불쾌지수 조절 시스템은 Team4에서 기획한 스마트 홈 시스템이다. 본 시스템은 유저의 별다른 개입 없이 자동적으로 유저의 피드백을 인식하고 이를 통해 실내 환경을 개선할 수 있도록 설계되었다. 본 서비스는 유저의 몸 상태, 실내 활동 상태 등을 실시간으로 수집하여 다이나믹한 유저의 활동에 따라 유동적인 실내 환경 조성을 도와준다. 유저는 앱 서비스를 통해서 목표 환경 수치를 조정할 수 있으며, 본 시스템은 이러한 피드백을 비중있게 고려해서 서버 내의 환경 조절 모델을 재 학습하는데 사용한다. 본 디자인 명세서에는 스마트 불쾌지수 조절 시스템에 직/간접적으로 사용되는 여러 디자인 구조가 묘사되어 있다. 묘사된 디자인 구조는 요구사항 명세서에서 명시된 요구사항들에 따라 설계되어 있다.

2.1 Objectives

이 챕터에서는 스마트 불쾌지수 조절 시스템의 디자인 명세서에서 사용한 툴 및 다이어그램을 설명하였다.

2.2 Applied Diagrams

2.2.1 UML

UML 이란 Unified Modeling Language의 약자로 1997년 OMG(Object Management Group)에서 표준으로 채택한 통합모델링 언어이다. 즉, 모델을 만드는 표준언어이다. 프로그램 설계를 표현하기 위해 사용하는 주로 그림으로 된 표기법을 의미한다. 객체지향 언어와 밀접한 관련이 있기에 객체지향 모델링 언어라고도 불린다.

2.2.2 Use case Diagram

Use Case Diagram은 본 서비스의 사용자의 요구사항에 따른 사용 예시에 대한 Diagram이다. 시스템의 설계에 있어 요구사항이 가장 기초에 있는 것처럼 Use Case

Diagram은 시스템의 주요한 사용 상황에 대한 요약이라고 할 수 있다. Use Case의 Scope는 본 시스템이 제공하는 기능의 범위를 나타낼 때 쓰이고, Scope 안의 Use Case들은 시스템이 제공해주는 서비스와 기능을 나타내며, 사용자의 요구사항을 구조화한 것이다.

2.2.3 Sequence Diagram

Sequence Diagram은 시스템의 아키텍처의 발생 순서와 상황을 더욱 직접적으로 보여주는 Diagram이다. 이는 시간에 따른 순차적인 진행 과정을 보이는 모든 산업에서 중요하게 쓰이는 Diagram이다. Sequence Diagram을 통해서 장황하지 않고 일관성 있는 표준화된 작업 과정을 보여줄 수 있다. 본 서비스에 있어서 Sequence Diagram은 비교적 복잡할 수 있는 유저 피드백 수렴과 모델 재훈련의 과정을 시간 순서로 순차적으로 표현해 줄 수 있는 주요한 도구가 될 것이다. 또한 이는 사용자, 시스템, 저장소 간 어떤 상호 작용을 하는지 보여줄 것이다.

2.2.4 Class Diagram

Class Diagram은 본 시스템의 아키텍처를 더욱 직접적으로 보여주는 Diagram이다. Class Diagram에서는 시스템을 구성하는 객체와, 그 객체의 속성, 함수, 그리고 객체 간의 관계를 보여준다. 시스템은 각 컴포넌트로 이루어져 있고 현대의 소프트웨어는 객체 지향 프로그래밍을 지향하기에 Class Diagram을 통해서 대략적인 시스템 구성을 볼 수 있다. Class Diagram에서 하나의 Class는 3개의 행과 1개의 열을 갖는 사각형으로 묘사된다. 맨 윗줄에는 Class의 이름이, 가운데 줄에는 Class의 attribute, 맨 아랫줄에는 Class의 method가 표시된다.

2.2.5 Context Diagram

Context Diagram은 시스템의 데이터 플로우를 표현하는 하이 레벨의 Diagram이다. 이 Diagram 예선 전체 시스템을 대표하고 시스템의 맥락과 범위를 만드는 가장 큰

범위 만이 표현된다. 즉, 시스템과 사용자 간의 데이터의 큰 흐름만 표현하는 것이다. Context Diagram은 이미 요구사항 명세서에서 기술하였고, 이해를 돋기 위해 본 디자인 명세서에도 일관된 Context Diagram을 제시할 것이다. 이 Diagram은 모든 이해관계자가 이해할 수 있도록 전문용어가 아닌 누구라도 이해할 수 있는 용어로 기술되어야 한다. 시스템 전체를 하나의 프로세스로 보여준다는 점이 다른 Diagram과의 차이점이라고 할 수 있다.

2.2.6 Entity Relationship Diagram

데이터베이스 구조를 모델링 할 때 entity relationship diagram은 entity들의 속성을 보여줄 수 있는 diagram이다. entity relationship diagram에서 entity는 사각형으로 표현되고, 속성은 타원으로 표현된다. entity들과 속성들은 선으로 연결되고, 관계 집합들은 마름모로 표기된다.

2.3 Applied Tools

2.3.1 Microsoft Word

이 툴은 본 프로젝트의 요구사항 명세서, 디자인 명세서를 작성하는 데에 사용되었으며, 팀원들 간 용이하게 작업을 공유 및 편집하기 위해 채택하였다. 팀원들 간의 서로 다른 문서 편집환경에서 제일 공통된 툴이며, 작성된 파일에 대한 다양한 편집 및 수정이 가능하므로 Team 4는 본 툴을 기본 편집 툴로 사용하기로 하였다.

2.3.2 Microsoft PowerPoint

이 툴은 프로젝트 발표에 주로 사용되었다. 시각적으로 효과적으로 발표할 수 있는 가장 대중적인 툴이며, 가장 널리 사용되고 여러 기기와 호환이 되어서 Team 4는 본 툴을 사용하기로 하였다.

t of Microsoft Office, has become the most widely used presentation program in the world.

2.3.3 Google Docs

팀원 간의 진행상황을 효과적으로 공유할 수 있는 툴이다. 기본적으로 2.3.1 과 형태가 비슷하지만 편집할 수 있는 범위에 제약이 있다. 이 툴은 팀원 간 진행상황을 공유하고 공통적인 부분을 서로 참고하고 일관성을 유지할 수 있도록 사용 되었다.

2.3.4 draw.io

Draw.io 는 플로우 차트를 작성하는 툴로, 자체적으로 UML을 통한 Diagram 작성을 지원하기 때문에 본 프로젝트에서 사용하는 수많은 Diagram의 작성에 직접적인 큰 도움을 줄 수 있다. 또한 웹 브라우저에서도 작성할 수 때문에 별도의 설치가 필요 없고 네트워크가 연결된 환경이라면 어디서든 Diagram의 작성할 수 있게 해준다. 따라서 Team 4는 UML 작성에 본 툴을 사용하기로 하였다.

2.4 Project Scope

스마트 불쾌지수 조절 시스템은 그 이름에 따라, 가장 “스마트”하게 유저에게 최상의 실내환경을 조성할 수 있도록 설계된 프로젝트이다. 기본적으로 본 프로젝트는 사용자의 암시적인 피드백을 비중있게 받아들여 반자동적으로 사용자에게 최적환경을 제공할 수 있도록 설계 되었다. 시스템을 보조할 수 있도록 리모콘 컨트롤러 컨셉의 앱이 제작도리 것이고, 이러한 앱을 통한 실내 환경 조절은 피드백으로 서버에 저장되어 모델 재훈련에 사용된다. 스마트 홈 디바이스가 점점 늘어나는 추세이고, 더 다양한 기기와의 호환을 염두해두면서 전체 시스템을 설계하였다.

2.5 References

- Team 1, 2020 Spring, Software Design Document, SKKU.

3. System Architecture - Overall

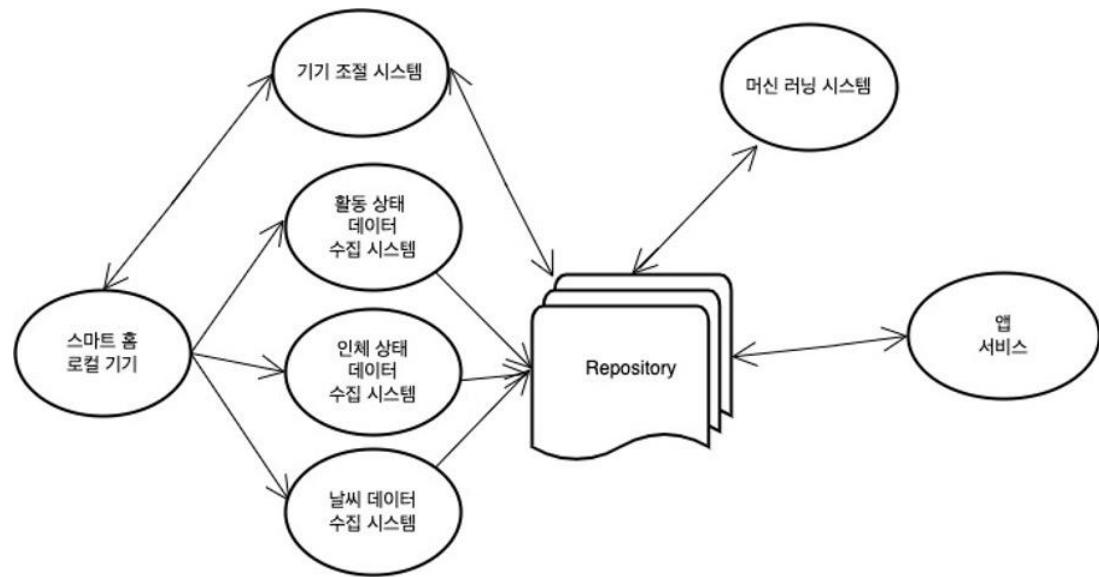
3.1. Objectives

이 챕터에서는 시스템의 프론트엔드부터 백엔드까지의 디자인 구성을 구체적으로 묘사하였다.

3.2. System Organization

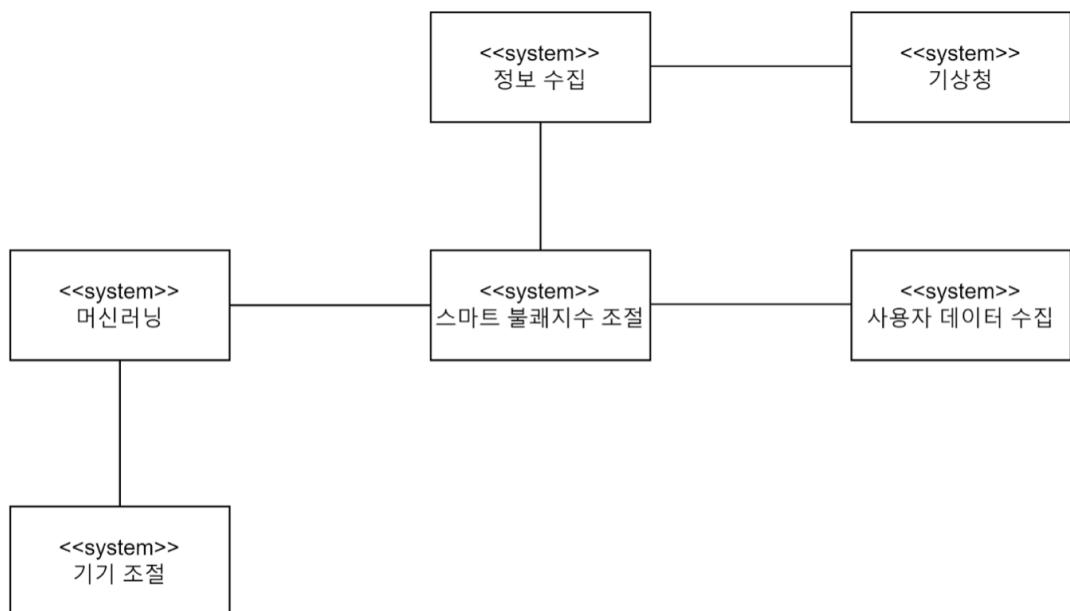
스마트 불쾌지수 조절 시스템은 유저의 별다른 개입없이 자동적으로 유저의 피드백을 인식하고 이를 통해 실내 환경을 개선할 수 있도록 설계되었다. 본 서비스는 유저의 몸 상태, 실내 활동 상태 등을 실시간으로 수집하여 다이나믹한 유저의 활동에 따라 유동적인 실내 환경 조성을 도와준다. 유저는 앱 서비스를 통해서 목표 환경 수치를 조정할 수 있으며, 본 시스템은 이러한 피드백을 비중있게 고려해서 서버 내의 환경 조절 모델을 재 학습하는데 사용한다. 본 시스템의 주요한 목적과, 스마트 홈 생태계의 기기 다양화 추세를 모두 고려하여서 본 시스템의 아키텍처는 저장소 아키텍처를 채택하였다. 이를 통해 추후 다양한 스마트 기기들을 서버에 추가하여서 더 다양하고 사용자 친화적인 서비스를 구축하려 하였다. 사용자는 다양한 기기를 앱 서비스를 통해서 등록할 수 있고, 다양한 기기는 형식화된 데이터 형태로 저장소와 소통할 수 있다.

[Figure 1] Overall system architecture

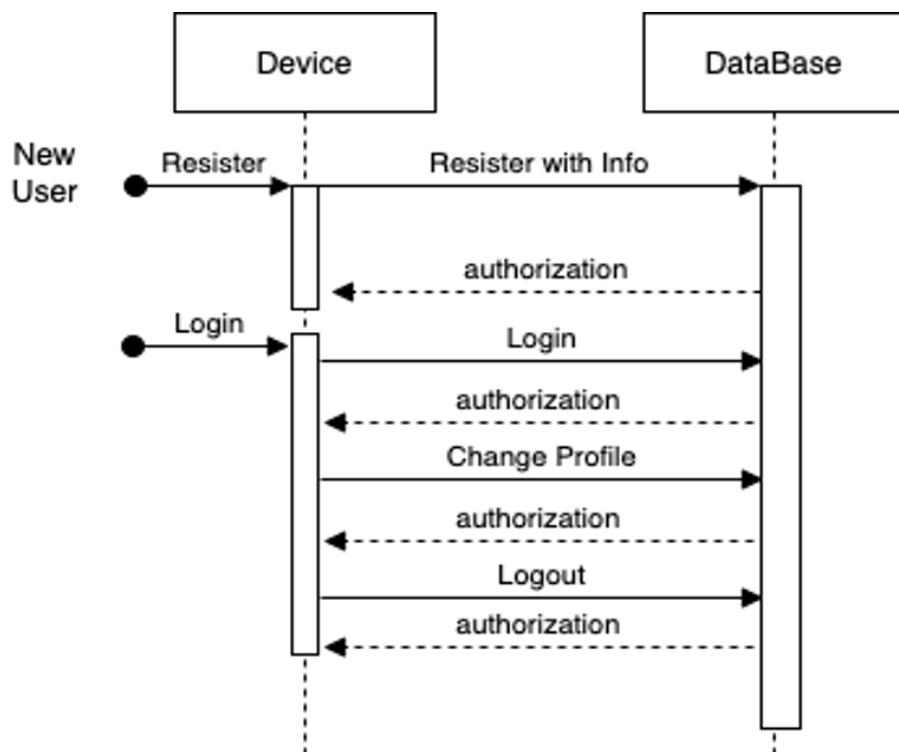


3.2.1 Context Diagram

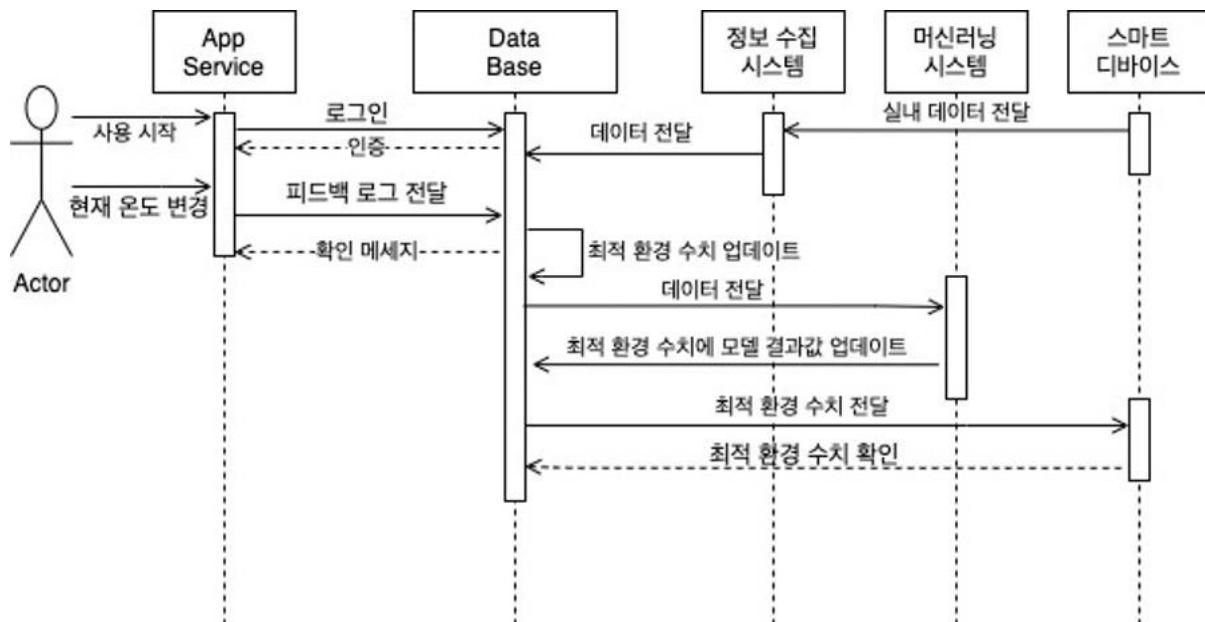
[Figure 2] Overall context diagram



3.2.2 Sequence Diagram

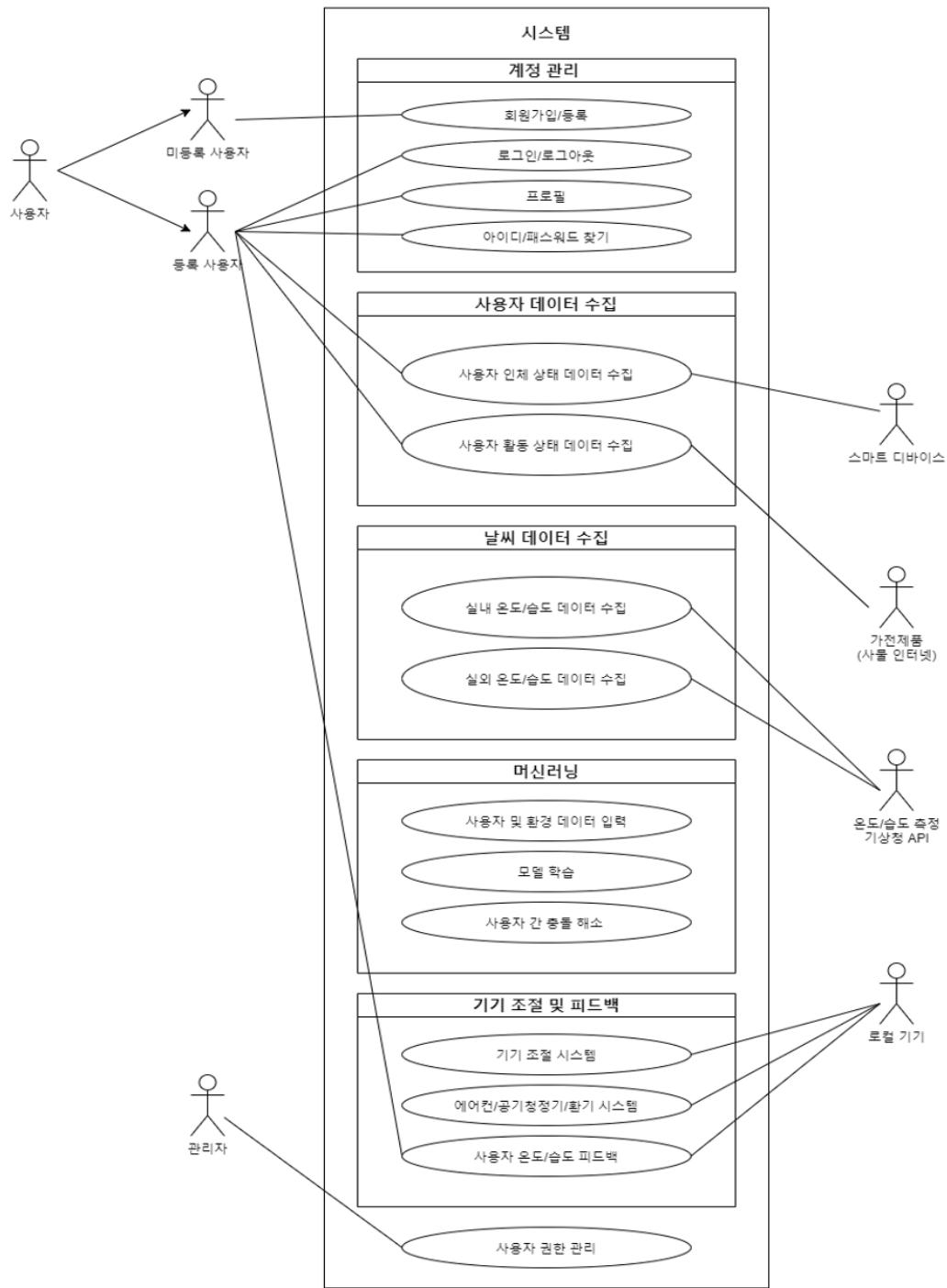


[Figure 3] 가입 / 로그인 sequence diagram



[Figure 4] 유저 피드백 반영 sequence diagram

3.2.3. Use Case Diagram



[Figure 5] Use case diagram

4. System Architecture - Frontend

4.1. Objectives

이 챕터에서는 System Architecture에서 사용자 인터페이스에 해당하는 Frontend 시스템을 이루는 Component들의 구성을 Class Diagram과 Sequence Diagram으로 도식화 및 설명한다.

4.2. Subcomponents

4.2.1. 로그인 및 회원가입

4.2.1.1. Attributes

해당 profile object가 가진 attribute는 다음과 같다.

- user_id : 로그인 시 사용되는 사용자의 id
- name : 사용자의 이름
- age : 사용자의 연령
- gender : 사용자의 성별

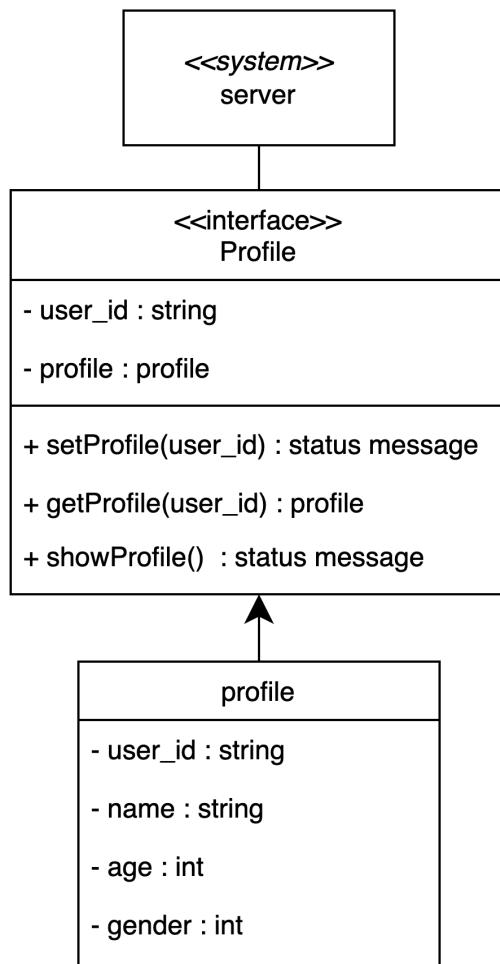
4.2.1.2 Methods

해당 profile class가 가지는 method는 다음과 같다.

- setProfile()
- getProfile()
- showProfile()

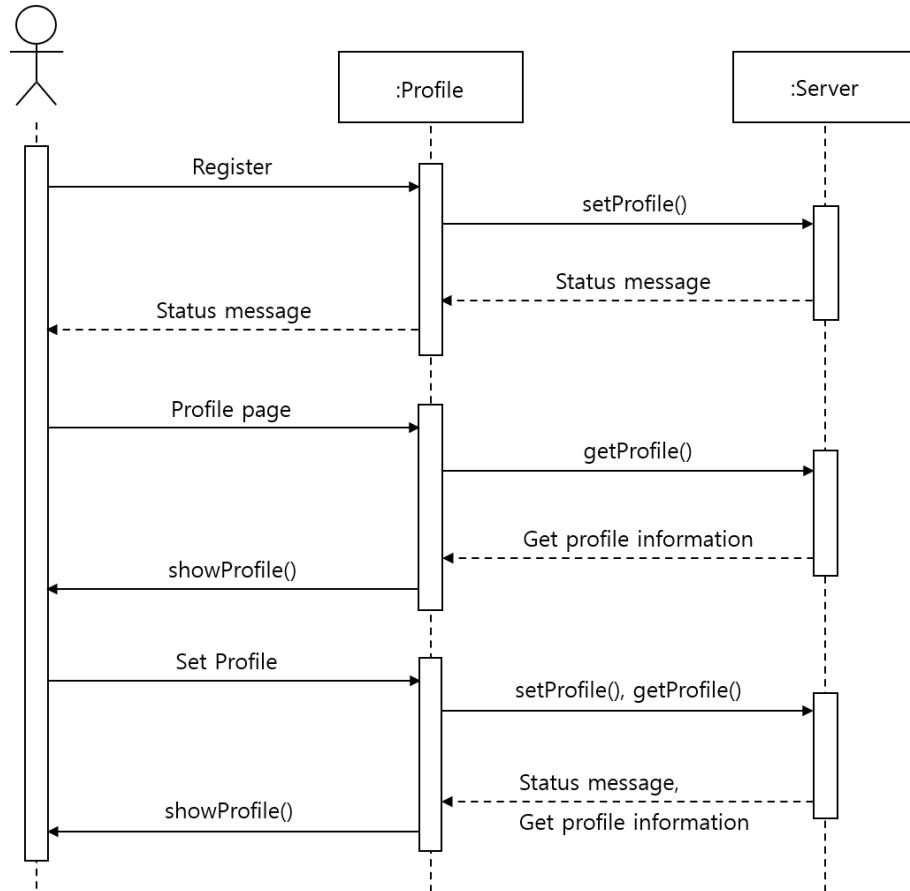
4.2.1.3 Class Diagram

[Figure 6] Class diagram - Profile



4.2.1.4 Sequence Diagram

[Figure 7] Sequence diagram - Profile



4.2.2. 기기 전원 켜기/끄기

4.2.2.1. Attributes

해당 power object가 가진 attribute는 다음과 같다.

- power : 기기 전원의 on/off 상태 저장

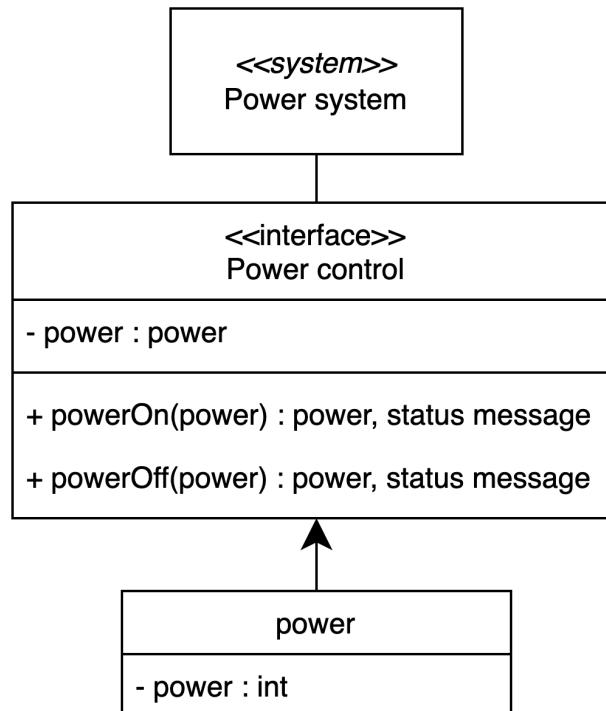
4.2.2.2 Methods

해당 power class가 가진 method는 다음과 같다.

- power_on()
- power_off()

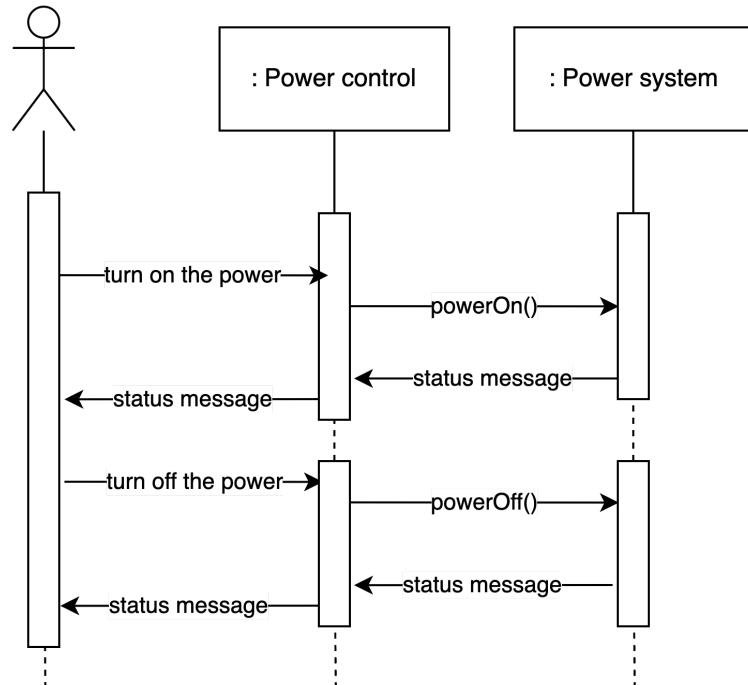
4.2.2.3 Class Diagram

[Figure 8] Class diagram - Power control



4.2.2.4 Sequence Diagram

[Figure 9] Sequence diagram - Power control



4.2.3. 온도/습도 설정

4.2.3.1. Attributes

해당 Air Condition class가 가진 attribute는 다음과 같다.

- room_temperature : 현재 방의 온도
- room_humidity : 현재 방의 습도
- temperature_input : 사용자가 입력한 희망 온도
- humidity_input : 사용자가 입력한 희망 습도

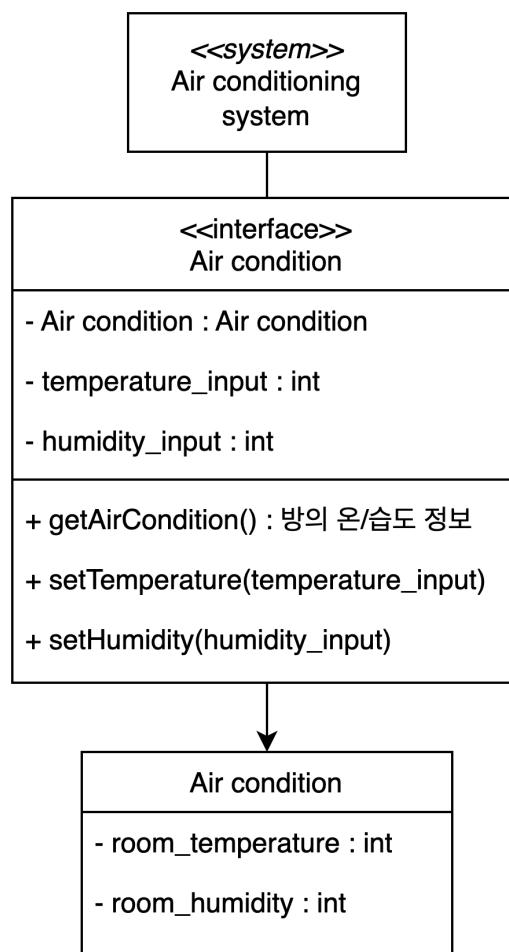
4.2.3.2 Methods

These are the methods that the search result class has.

- `getAirCondition()`
- `setTemperature(temperature_input)`
- `setHumidity(humidity_input)`

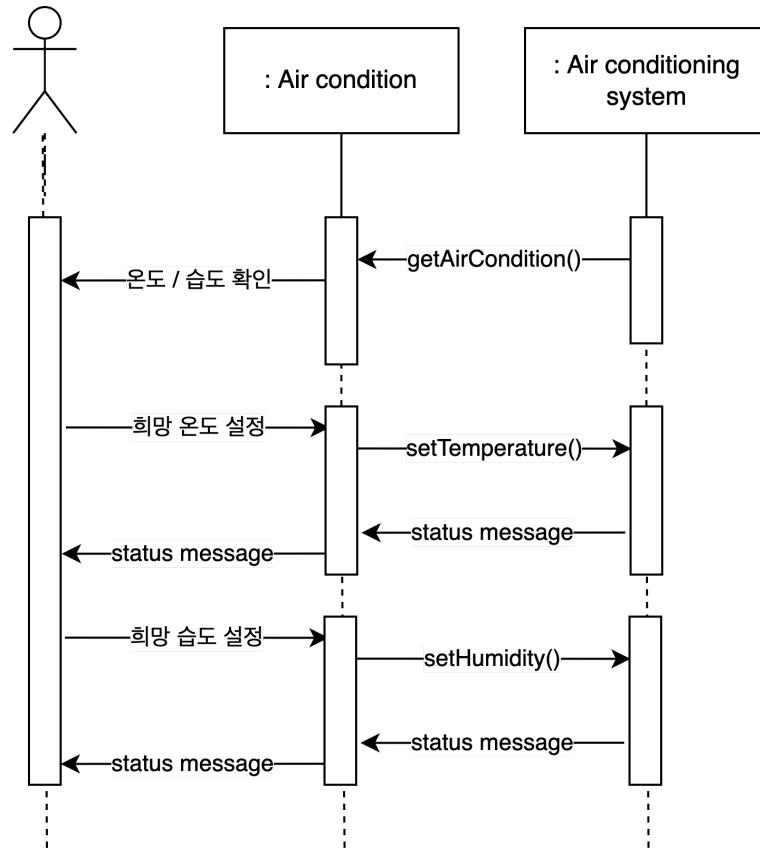
4.2.3.3 Class Diagram

[Figure 10] Class diagram - Air condition



4.2.3.4 Sequence Diagram

[Figure 11] Sequence diagram - Air condition



4.2.4. 비밀번호 분실

4.2.4.1. Attributes

해당 password object가 가진 attribute는 다음과 같다.

- user_id : 로그인 시 사용되는 사용자의 id
- user_pw : 사용자가 설정한 password
- name : 사용자의 이름

- tel_num : 사용자의 전화번호

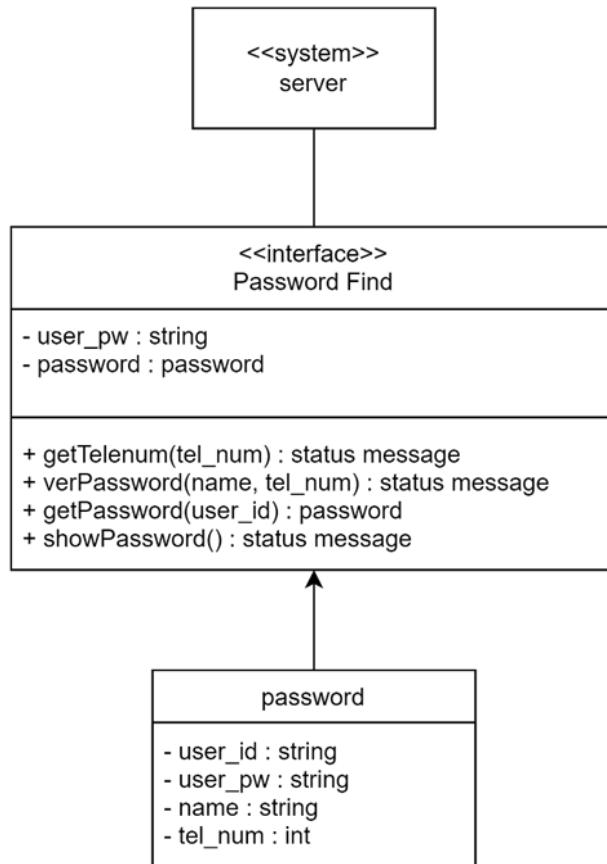
4.2.4.2 Methods

해당 password find class가 가진 method는 다음과 같다.

- getTelenum()
- verPassword()
- getPassword()
- showPassword()

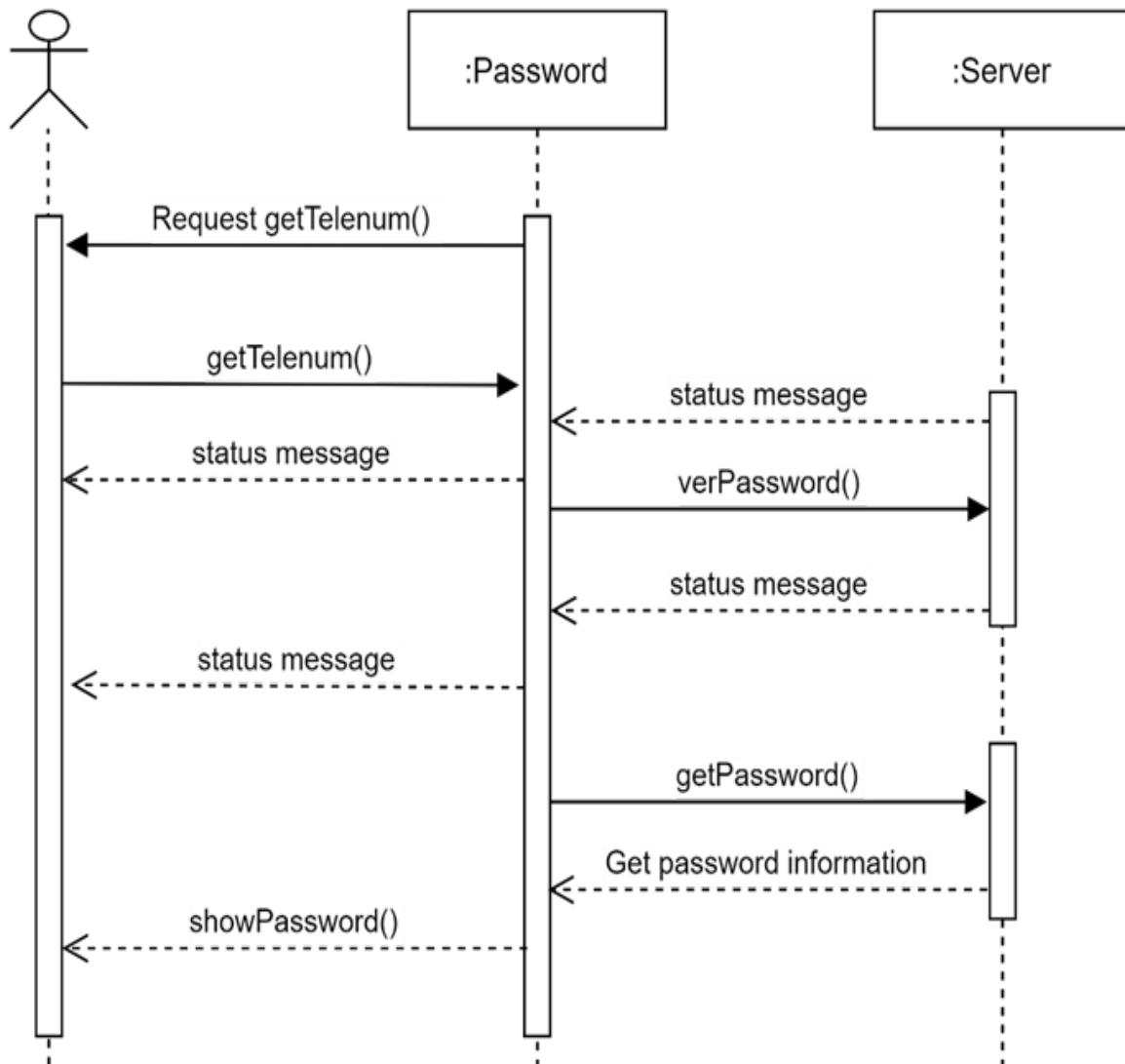
4.2.4.3 Class Diagram

[Figure 12] Class diagram – Password Find



4.2.4.4 Sequence Diagram

[Figure 13] Sequence diagram – Password Find



4.2.5. 기기 및 장소 추가

4.2.5.1. Attributes

해당 Add Place/Product class가 가진 attribute는 다음과 같다.

- place_list : 사용자가 추가한 장소들의 목록
- product_list : 사용자가 추가한 기기들의 목록
- place_name : 사용자가 추가한 장소들의 이름
- product_name : 사용자가 추가한 기기들의 이름

해당 place_list object가 가진 attribute는 다음과 같다.

- place_name : 사용자가 추가한 장소들의 이름
- pl_list : 사용자가 추가한 장소들의 리스트

해당 product_list object가 가진 attribute는 다음과 같다.

- product_name : 사용자가 추가한 기기들의 이름
- pd_list : 사용자가 추가한 기기들의 리스트

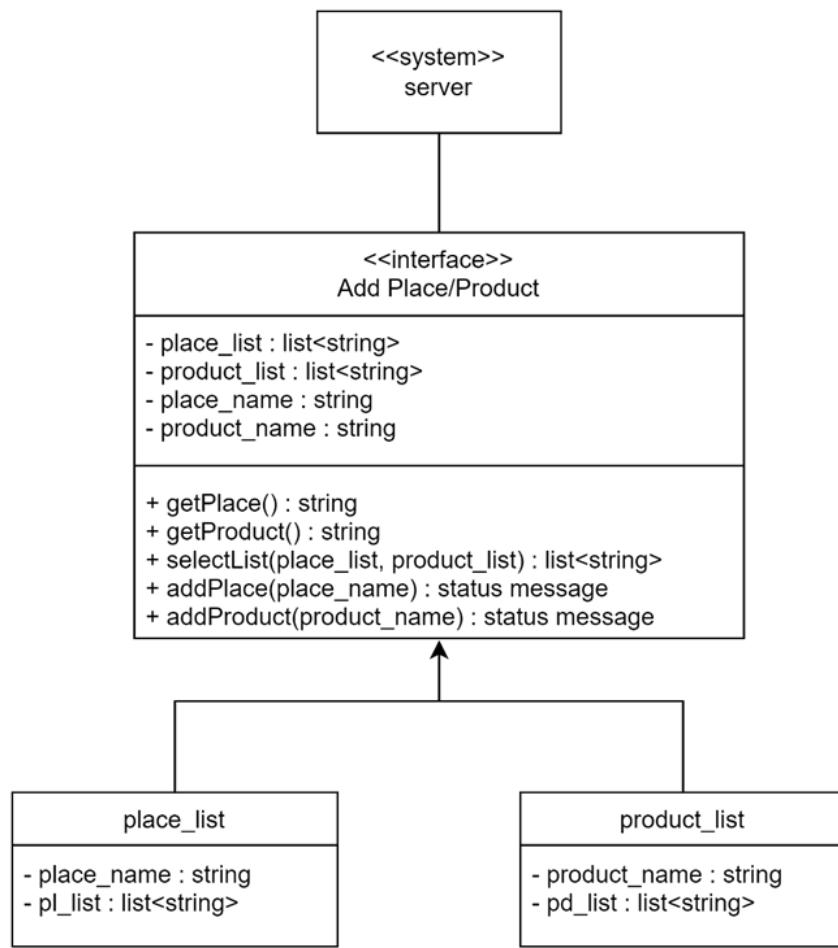
4.2.5.2 Methods

해당 Add Place/Product class가 가진 method는 다음과 같다.

- getPlace()
- getProduct()
- selectList()
- addPlace()
- addProduct()

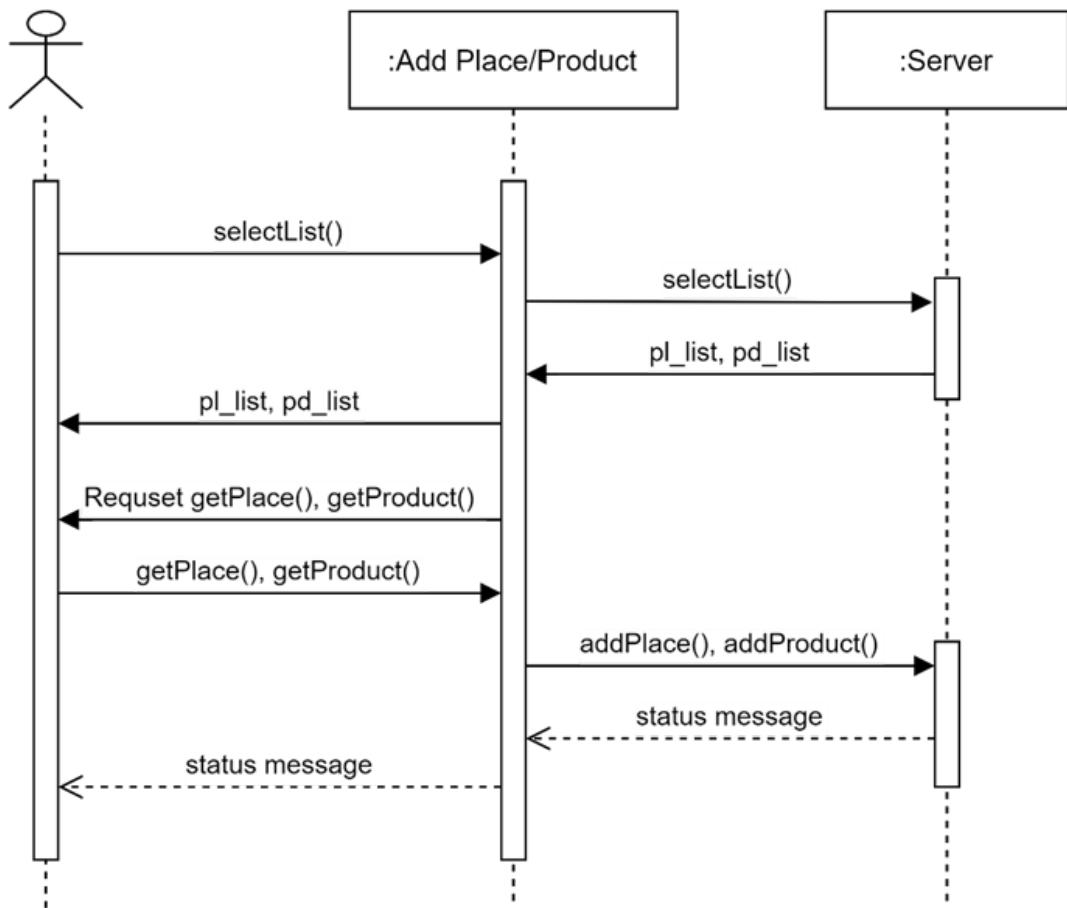
4.2.5.3 Class Diagram

[Figure 14] Class diagram – Add Place/Product



4.2.5.4 Sequence Diagram

[Figure 15] Sequence diagram – Add Place/Product



4.2.6. 사용자 피드백

4.2.6.1. Attributes

해당 User Feedback class가 가진 attribute는 다음과 같다.

- feedback : 시스템 기능에 대한 사용자의 피드백 정보

해당 feedback object 가 가진 attribute는 다음과 같다.

- user_satisfaction : 사용자의 만족도 수치
- curr_state : 현재 설정되어 있는 기기의 수치 정보

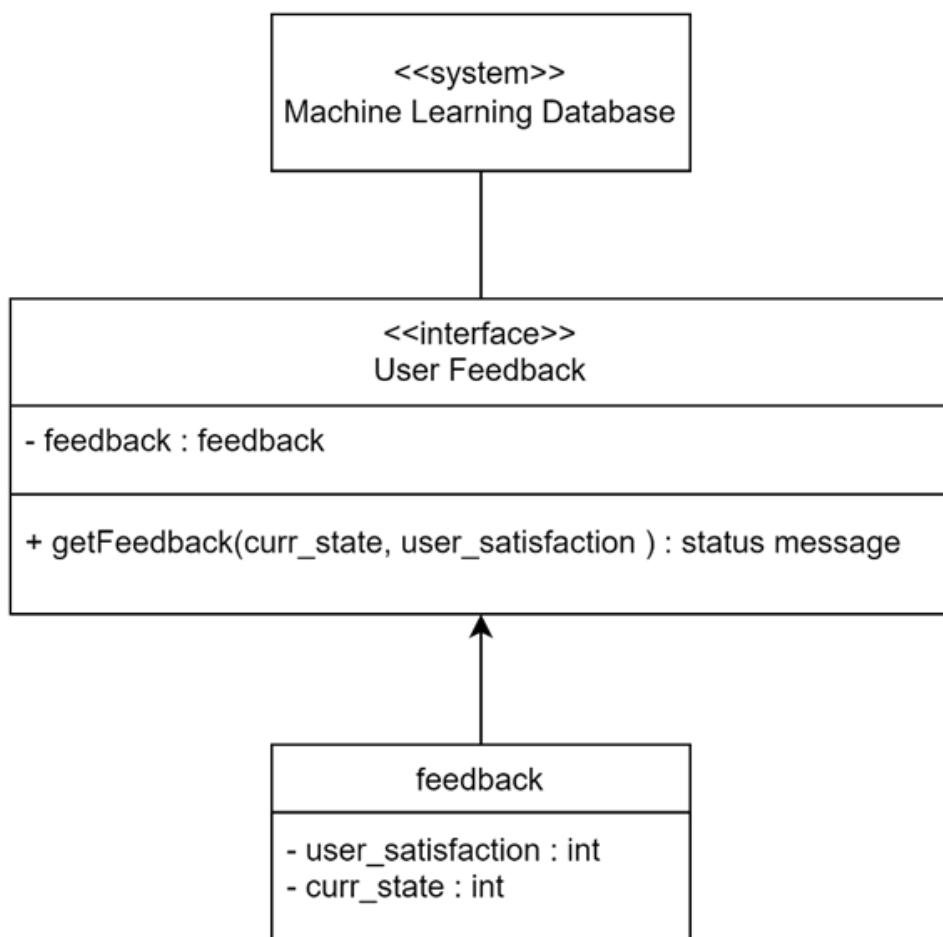
4.2.6.2 Methods

해당 User Feedback class가 가진 method는 다음과 같다.

- getFeedback()
- showMLresult()

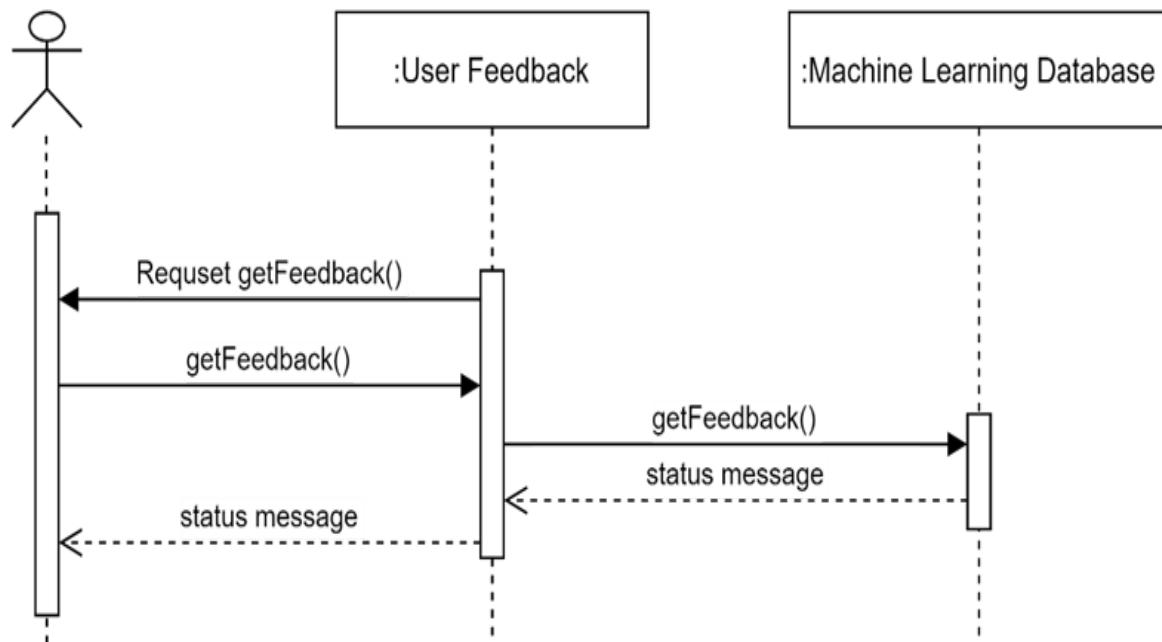
4.2.6.3 Class Diagram

[Figure 16] Class diagram – User Feedback



4.2.6.4 Sequence Diagram

[Figure 17] Sequence diagram – User Feedback



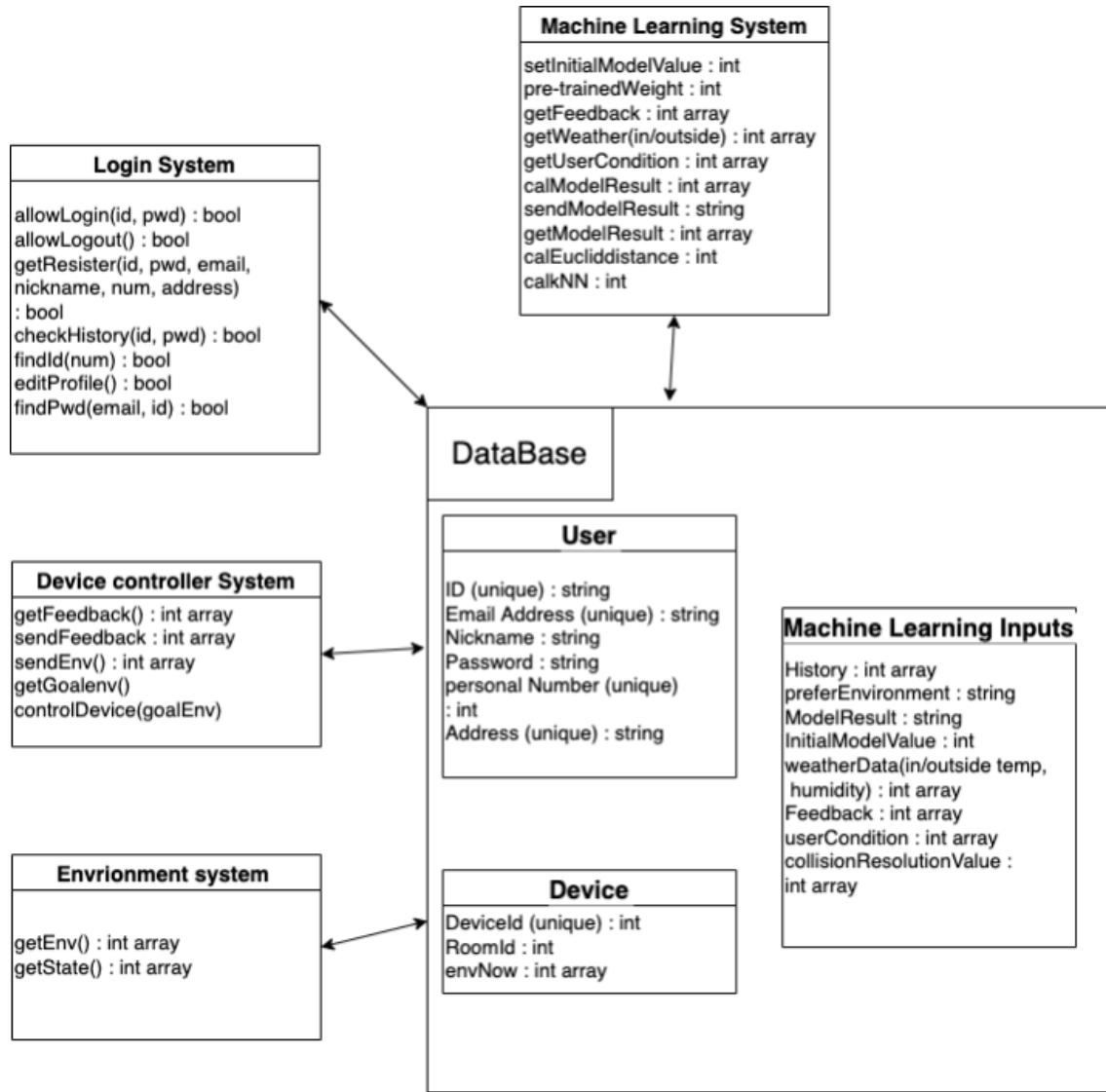
5. System Architecture - Backend

5.1. Objectives

이 챕터에서는 **System Architecture** 중에서 백엔드 시스템의 구조와 기능에 대해서 설명한다.

5.2. Overall Architecture

스마트 불쾌지수 조절 시스템은 다양화 되어가는 스마트 홈 기기의 추세를 지속적으로 반영할 수 있도록 Repository Architecture를 채택하였다. 이를 통해, 형식화된 데이터 형식을 통해서 다양한 기기의 연결이 가능하도록 하였다. 본 디자인 명세서에서는 스마트 홈 기기 중 대표적으로 스마트 워치, 에어컨, 공기청정기, 정수기, 운동기기 등을 예로 들었다. 다양한 스마트 홈 기기가 추가 가능하며, 이 때는 데이터 베이스의 History 부분과 형식을 맞출 필요가 있다.



[Figure 18] Overall architecture

5.2.1. 저장소(Repository)

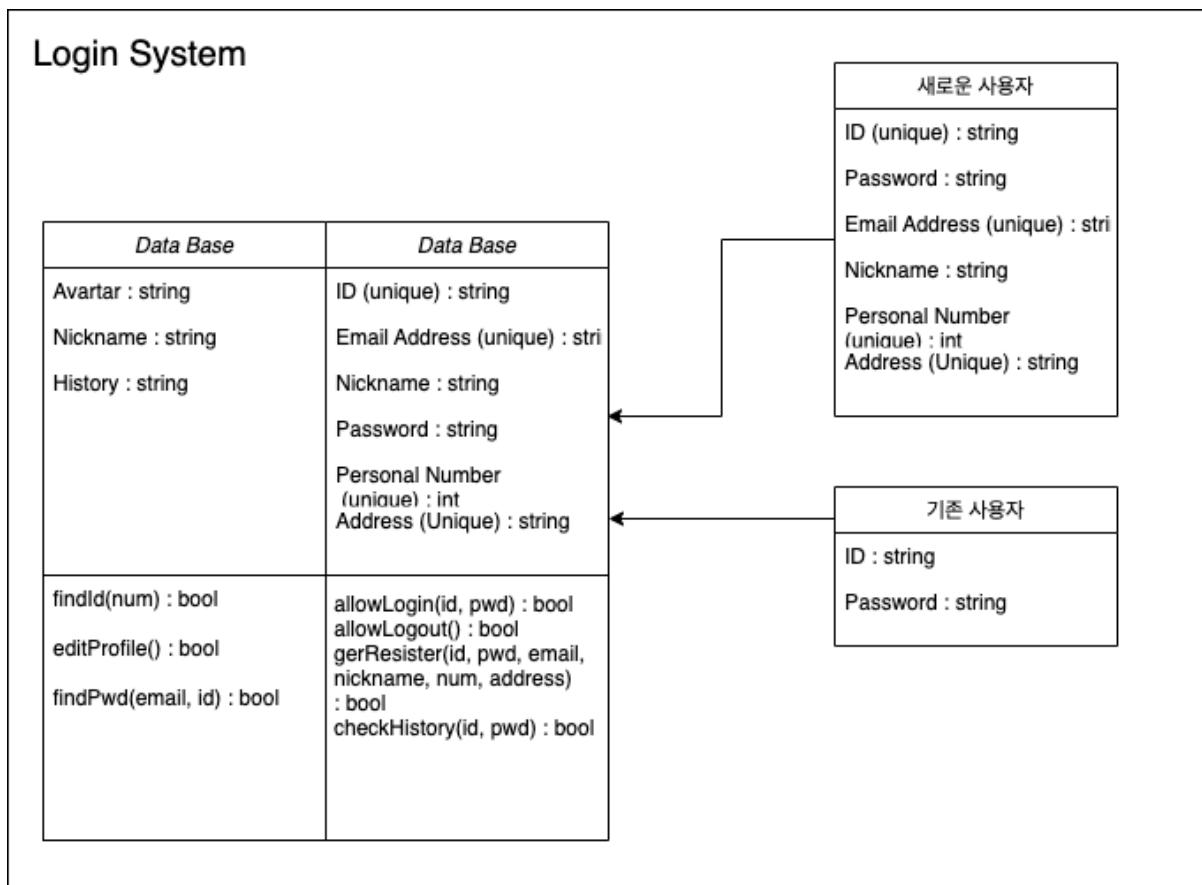
저장소 아키텍쳐는 데이터 생성과 사용의 기기가 다를 때 주로 사용된다. 저장소 아키텍쳐는 형식화된 데이터 규칙만 지킨다면 기존 오브젝트와 상관없이 새로운 오브젝트를 자유롭게 생성 가능한 것이 장점이다. 이러한 장점이 스마트 홈 디바이스가 점점 늘어나는 현재 추세에 따라 본 시스템을 발전시키기 용이하게 만들어 준다. 단, 모든 서비스가 저장소에 의지하고, 머신러닝의 특성상 데이터가 매우 중요하기 때문에 저장소의 보안 유지가 중요하다.

5.2.2. Systems

스마트 불쾌지수 조절 시스템은 저장소와 연결된 주요한 시스템들이 존재한다. 이는 주로, 정보 수집 시스템, 환경 수집 시스템, 기기 조절 시스템, 그리고 머신러닝 시스템이 있다. 각 시스템에 대한 자세한 설명은 5.3에서 다룰 것이다.

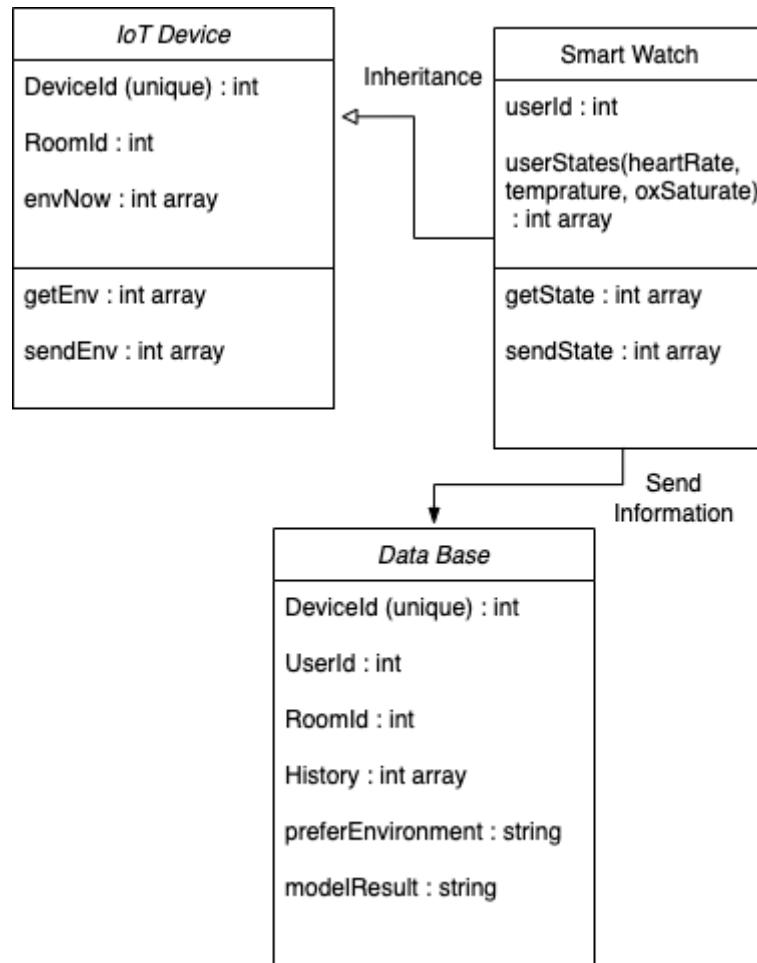
5.3. Subcomponents

5.3.1. 로그인 / 회원가입 시스템



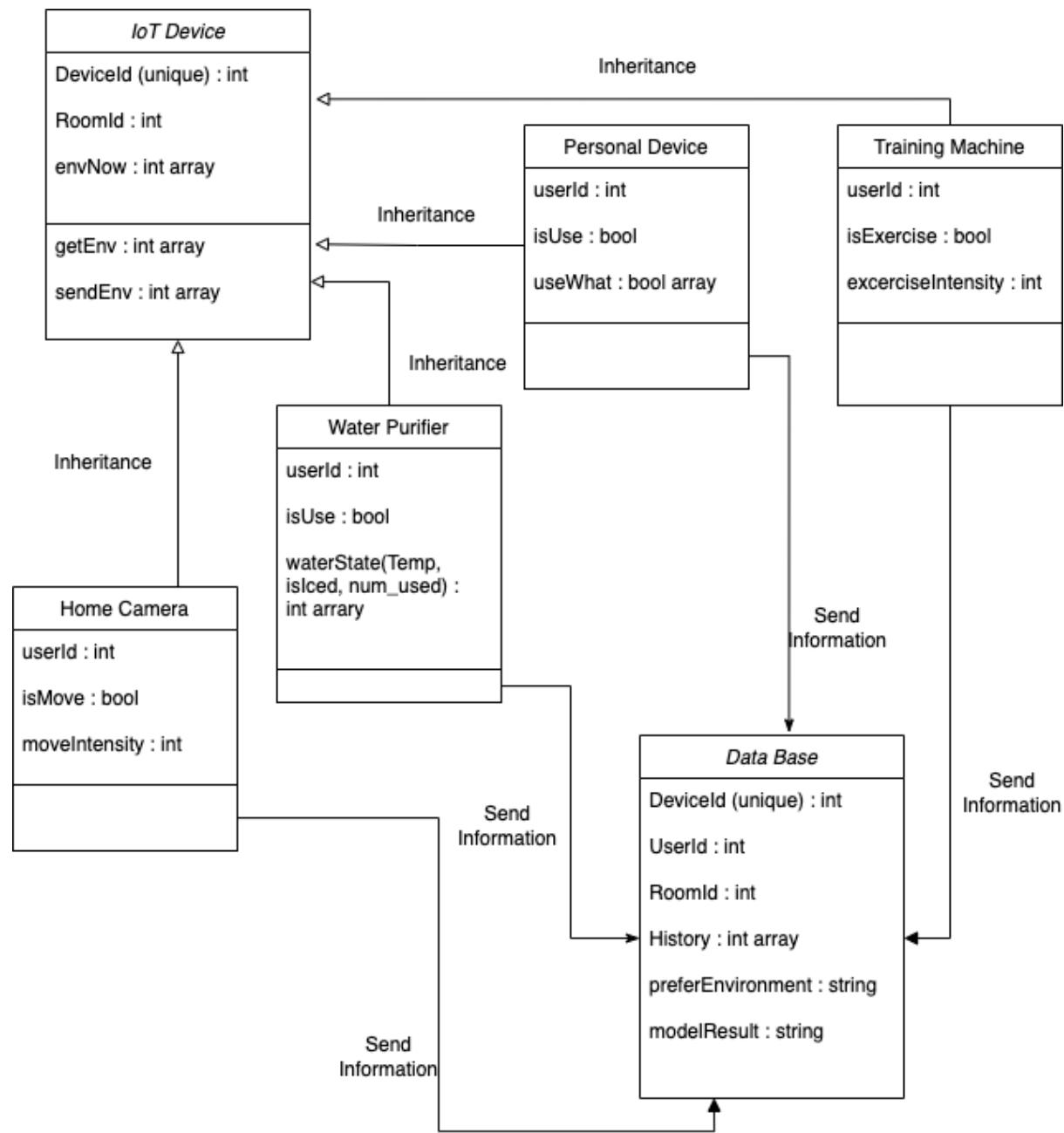
[Figure 19] Class Diagram - Login / Register

5.3.2. 스마트 위치를 통한 사용자 상태 데이터 수집 시스템



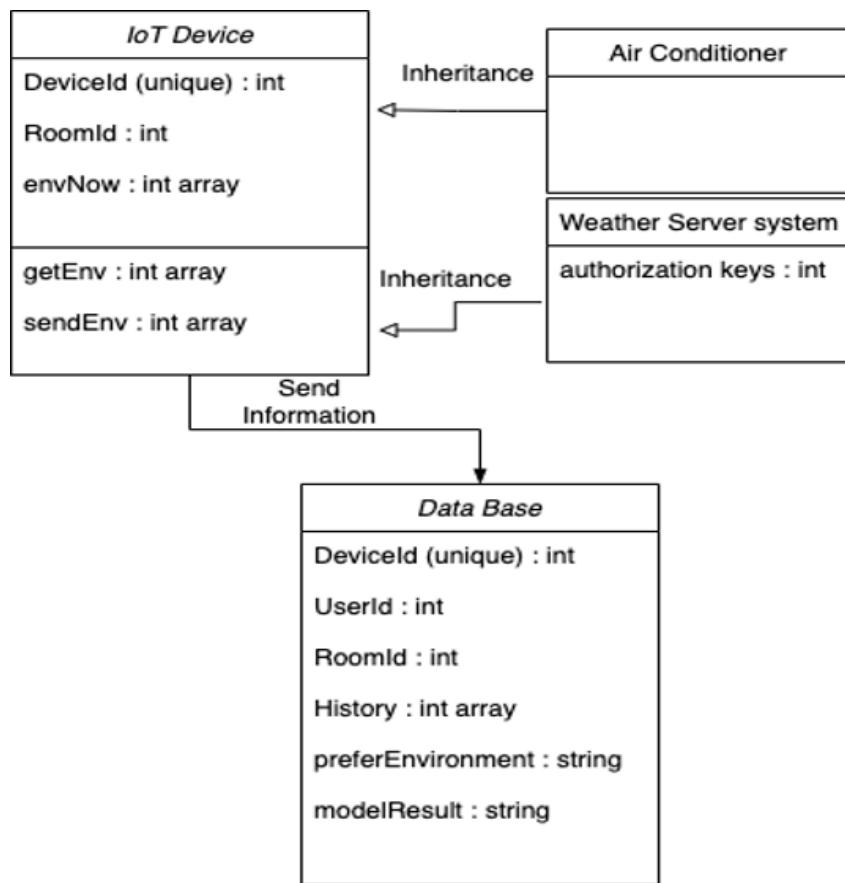
[Figure 20] Class Diagram - User Condition information

5.3.3. 사물인터넷을 통한 사용자 활동 데이터 수집 시스템



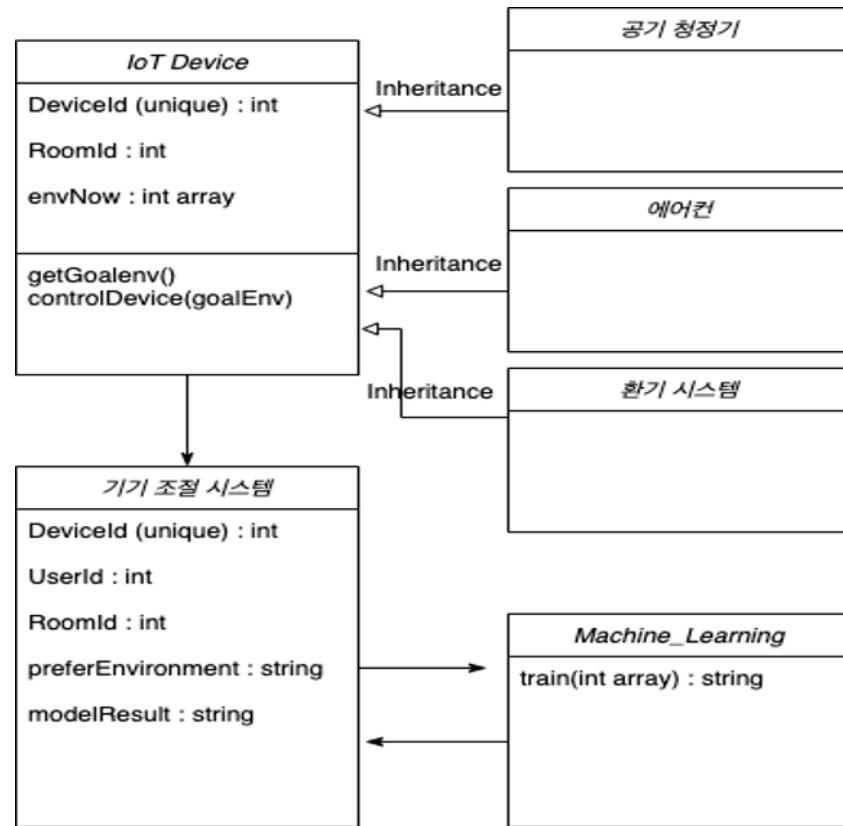
[Figure 21] Class Diagram - User Activity information

5.3.4. 실외 / 실내 환경 수집 시스템



[Figure 22] Class diagram - 실외 / 실내 환경 수집 시스템

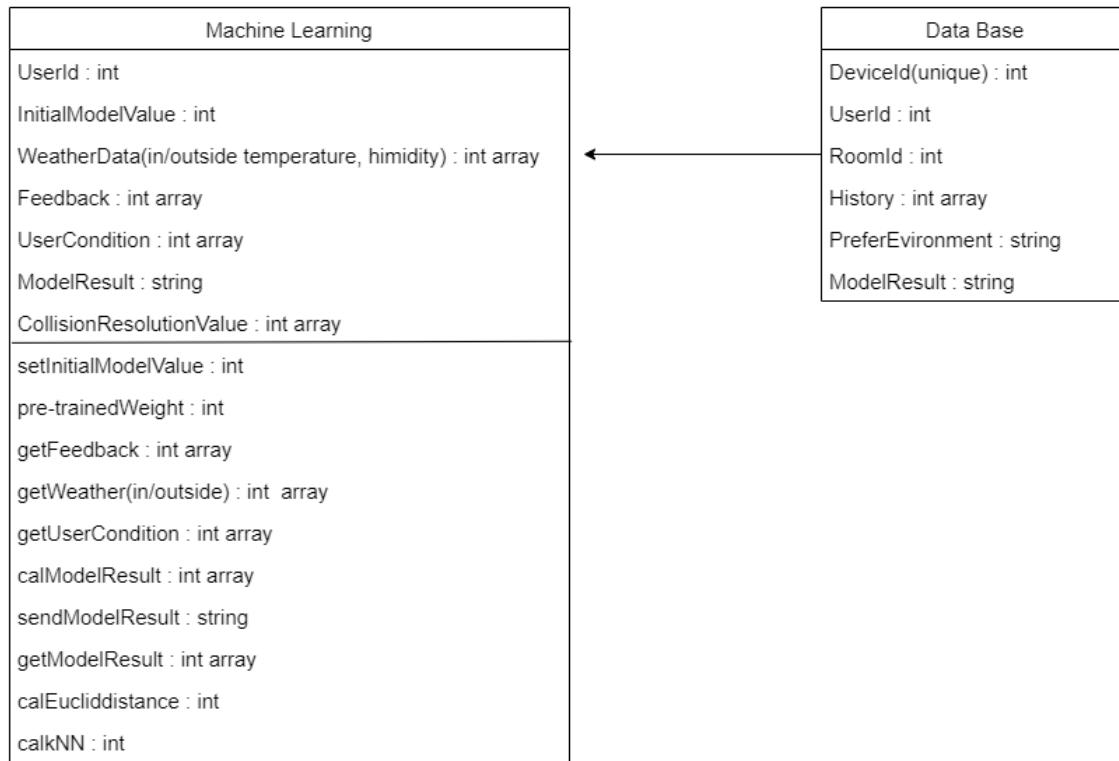
5.3.5. 기기 조절 시스템



[Figure 23] Class diagram - 실외 / 실내 환경 수집 시스템

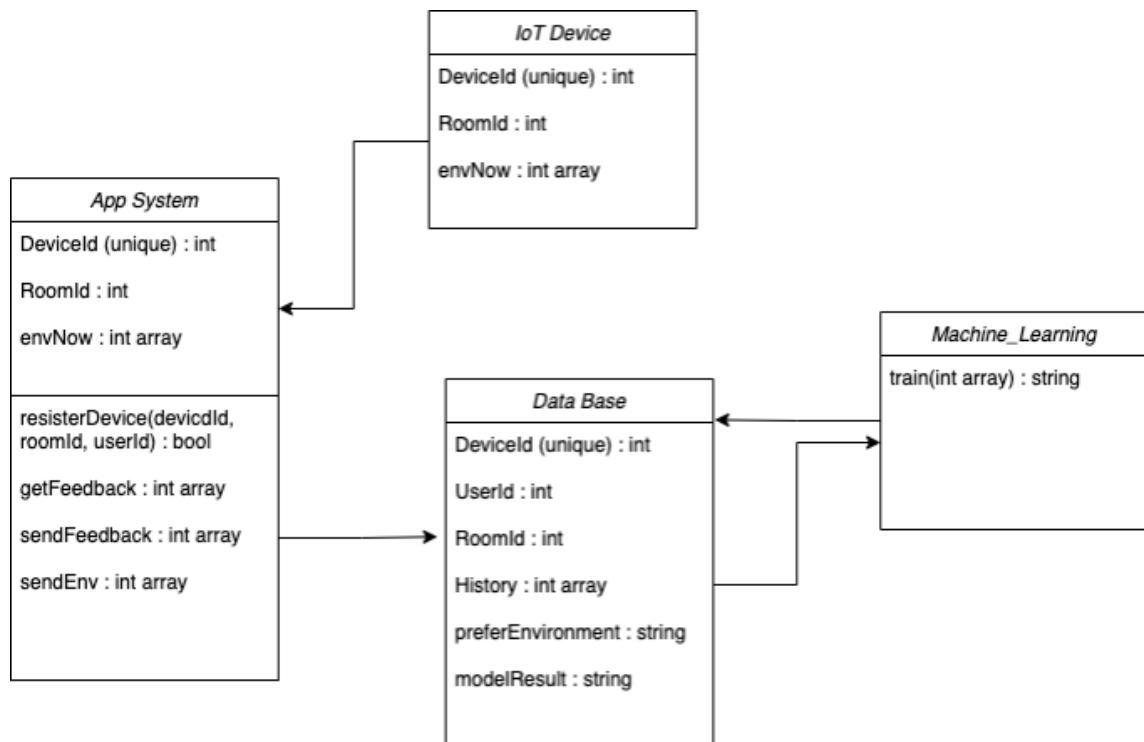
5.3.6. 머신러닝 시스템

Machine Learning System

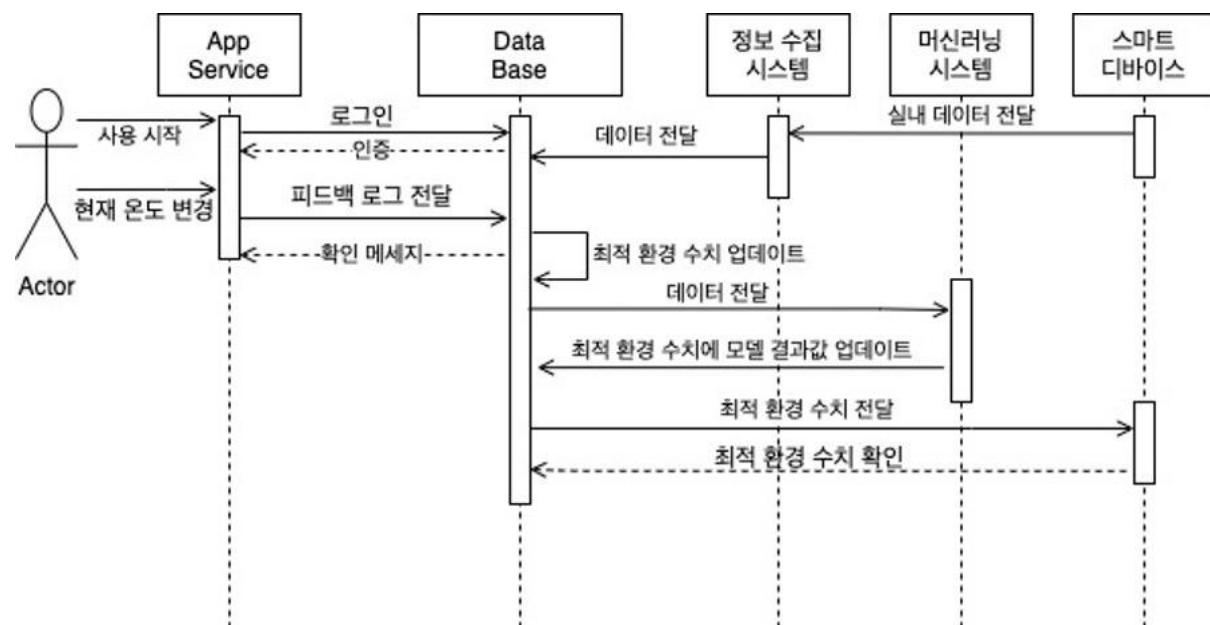


[Figure 24] Class diagram - Machine learning

5.3.7. 피드백 시스템



[Figure 25] Class diagram -피드백 시스템



[Figure 26] Sequence diagram - 피드백 시스템

6. Protocol Design

6.1. Objectives

이 챕터는 ‘스마트 불쾌지수 조절 시스템’의 어플리케이션과 서버 간의 통신이 어떻게 이루어지고 전달되는지를 기술한다.

6.2. HTTP

HTTP는 HyperText Transfer Protocol로 웹 상에서 정보를 주고받을 수 있는 프로토콜이다. 클라이언트와 서버 사이에 이루어지는 요청/응답(request/response) 프로토콜로 주로 HTML 문서를 주고받는 데에 쓰인다. 클라이언트가 서버로 요청 메시지를 전달하면 서버는 이에 대한 응답메시지를 보낸다. 이 응답메시지는 404 Not Found와 같이 세 자리 수로 된 응답 코드와 함께 응답한다.

본 시스템은 HTTP로 일반적인 통신을 담당한다.

6.3. 로그인 및 회원가입

6.3.1. 회원가입

- Request

[Table 1] register request

Attribute	Detail	
Protocol	OAuth	
Request body	Request token	Token for OAuth
	User	User information

- Response

[Table 2] register response

Attribute	Detail
Success Code	HTTP 200 OK
Failure Code	HTTP 400 Bad Request

	HTTP 401 Unauthorized	
	HTTP 404 Not Found	
	HTTP 500 Internal Server Error	
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] Log-in request

Attribute	Detail	
Protocol	OAuth	
Request body	Request token	Token for OAuth
	User	User information

- Response

[Table 4] Log-in response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 401 Unauthorized	
	HTTP 404 Not Found	
	HTTP 500 Internal Server Error	
	Access Token	Token for access
Success response body		Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.4. 비밀번호 찾기

- Request

[Table 5] Find Password request

Attribute	Detail	
URI	/user/:id/profile/pw	
Method	GET	
Parameter	User	Basic User Information
Header	Authorization	User authentication

- Response

[Table 6] Find Password response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 404 Not Found	
Success response body	Access Token	Token for access
	User Password	Show user password
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.5. 기기 전원 켜기 / 끄기

- Request

[Table 7] Power control request

Attribute	Detail	
URI	/user/:id/list/product/power	
Method	POST	
Parameter	product	power information
Header	Authorization	User authentication

- Response

[Table 8] Power control response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request HTTP 404 Not Found	
Success response body	Access Token Power control Message	
Failure response body	Access Token	Token for access
	Power control	turn on/off the power
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

6.6. 온도/습도 설정

- Request

[Table 9] Air conditioning request

Attribute	Detail	
URI	User/:id/list/product/air_condition	
Method	POST	
Parameter	Air condition	temperature, humidity
Header	Authorization	User authentication

- Response

[Table 10] Air conditioning response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request HTTP 404 Not Found	
Success response body	Access Token	Token for access
	Air condition	Set the room temperature and humidity
	User	Basic user information
	Message	Message: "Access success"
Failure response body	Message	Message: "Access fail"

6.7. 기기/장소 추가

6.7.1. 기기 추가

- Request

[Table 11] Add Product request

Attribute	Detail	
URI	/user/:id/list/product	
Method	POST	
Parameter	Product	Product to add
Header	Authorization	User authentication

- Response

[Table 12] Add Product response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 404 Not Found	
Success response body	Access Token	Token for access
	Message	Message: "Addition success"
Failure response body	Message	Message: "Addition fail"

6.7.2. 장소 추가

- Request

[Table 13] Add Place request

Attribute	Detail	
URI	/user/:id/list/place	
Method	POST	
Parameter	Product	Place to add
Header	Authorization	User authentication

- Response

[Table 14] Add Place response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 404 Not Found	
Success response body	Access Token	Token for access
	Message	Message: “Addition success”
Failure response body	Message	Message: “Addition fail”

6.8. 목록 불러오기

6.8.1. 기기 목록 불러오기

- Request

[Table 15] Load Product List request

Attribute	Detail	
URI	/user/:id/list/product	
Method	GET	
Parameter	List name	Name of list
Header	Authorization	User authentication

- Response

[Table 16] Load Product List response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request HTTP 404 Not Found	
Success response body	Access Token	Token for access
	List	Product List
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

6.8.2. 장소 목록 불러오기

- Request

[Table 17] Load Place List request

Attribute	Detail	
URI	/user/:id/list/place	
Method	GET	
Parameter	List name	Name of list
Header	Authorization	User authentication

- Response

[Table 18] Load Place List response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request HTTP 404 Not Found	
Success response body	Access Token	Token for access
	List	Place List
	Message	Message: “Access success”
Failure response body	Message	Message: “Access fail”

6.9. 피드백 작성

- Request

[Table 19] Send Feedback request

Attribute	Detail	
URI	/user/:id/feedback	
Method	POST	
Parameter	List name	Name of list
	Satisfaction	User's system satisfaction
	Current state	Value of current system
Header	Authorization	User authentication

- Response

[Table 20] Send Feedback response

Attribute	Detail	
Success Code	HTTP 200 OK	
Failure Code	HTTP 400 Bad Request	
	HTTP 404 Not Found	
Success response body	Access Token	Token for access
	User	Basic user information
	Message	Message: "Sending success"
Failure response body	Message	Message: "Sending fail"

7. Database Design

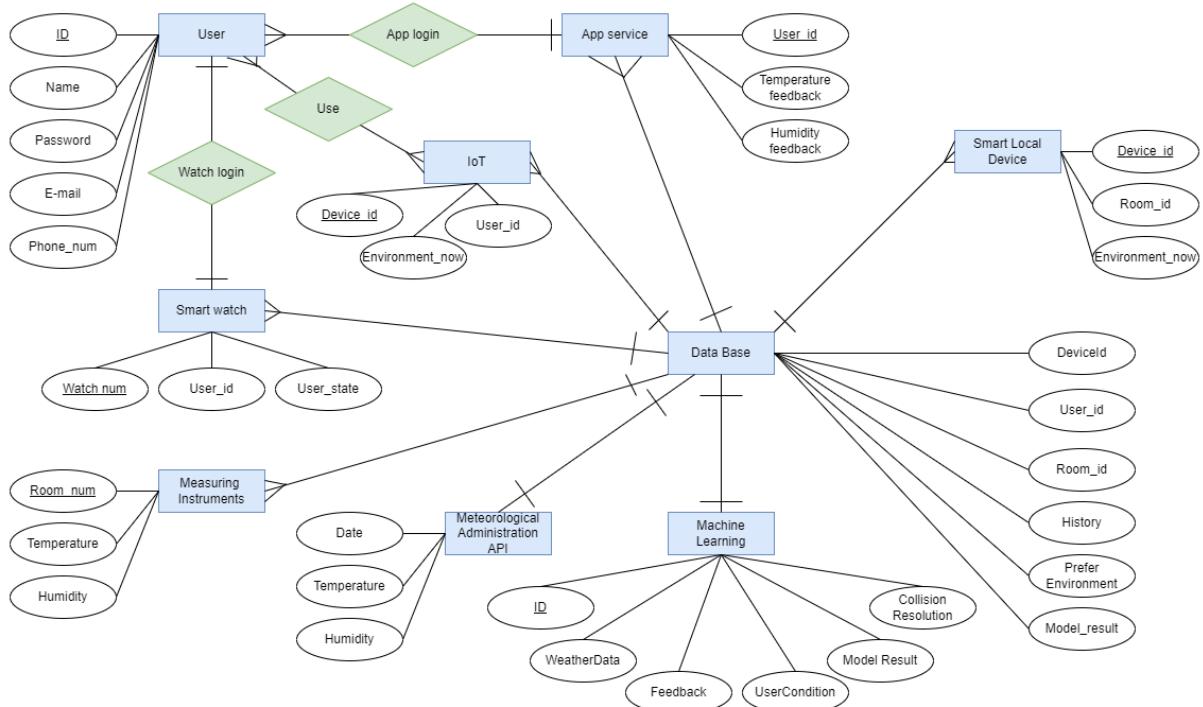
7.1. Objectives

해당 섹션에서는 데이터베이스의 구성과 관계를 상세히 기술한다. 데이터베이스의 E-R 다이어그램, 각 엔티티, 관계 스키마, 그리고 데이터베이스를 실제로 구현한 SQL DDL을 작성하였다.

7.2. ER Diagram

데이터는 총 9개의 엔티티로 구성된다. 사용자, 스마트 워치, IoT, 앱 서비스, 온도/습도 측정기, 기상청 API, 머신러닝, 로컬 기기, 통합 DB로 총 데이터베이스가 구성된다. 각 엔티티의 관계는 다음과 같다.

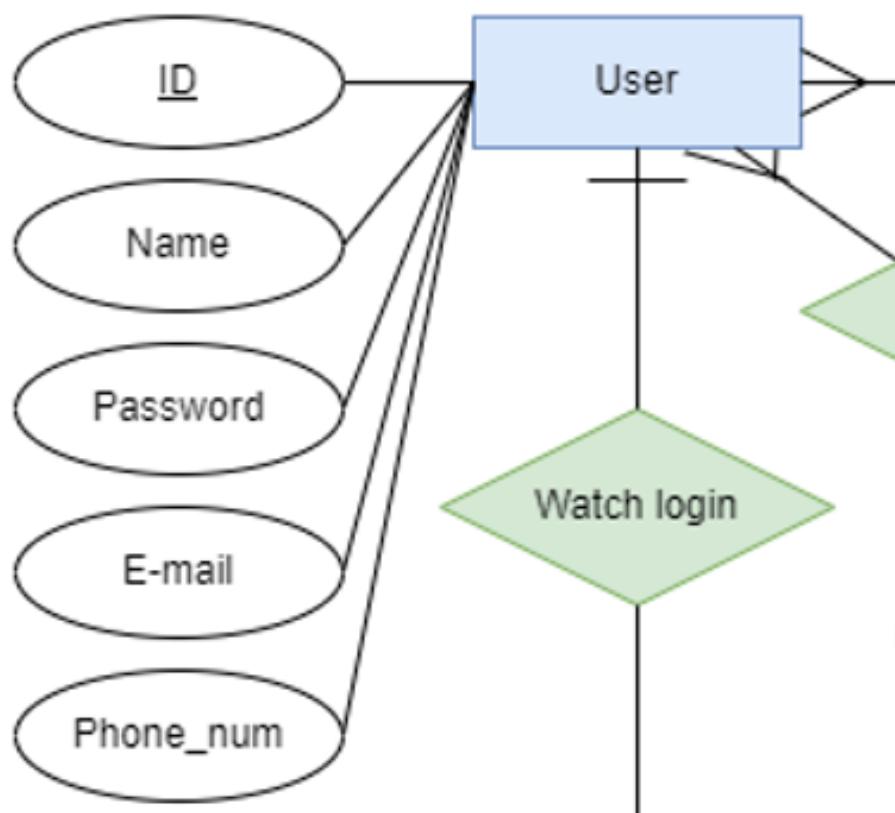
[Figure 27] ER-Diagram



7.2.1 Entities

7.2.1.1. User

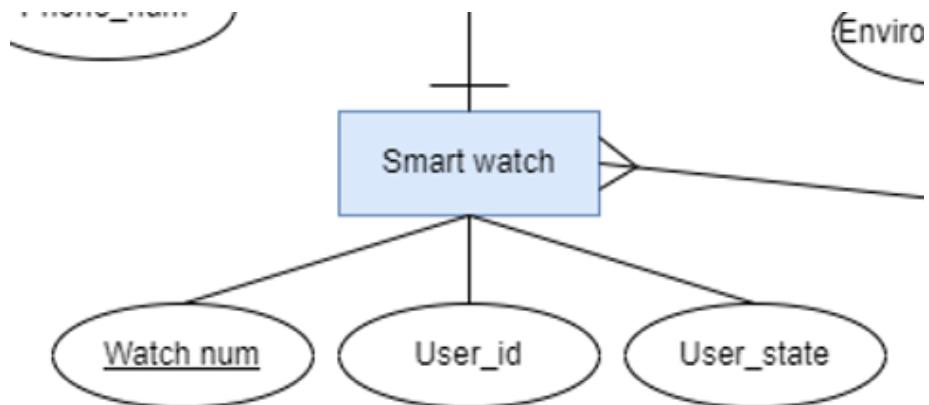
[Figure 28] ER diagram, Entity, User



User 엔티티는 유저에 관련된 정보를 포함한다. 아이디, 이름, 비밀번호, 이메일, 전화번호를 수집한다. 이 중, 아이디는 기본키로 지정되어 중복될 수 없다. 스마트워치와 일대일 관계를 맺으며, IoT와는 다대다 관계, 앱 서비스와는 다대일 관계를 맺는다.

7.2.1.2. Smart watch

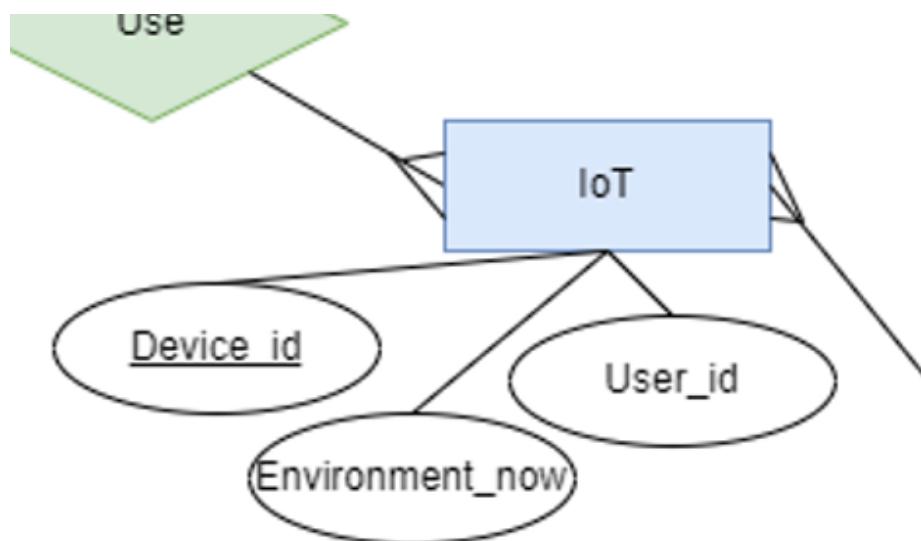
[Figure 29] ER diagram, Entity, Smart watch



스마트 워치 엔티티는 스마트 워치를 착용한 사용자의 상태 정보를 포함한다. 워치 번호를 기본키로 가지며, 사용자 아이디, 사용자 상태를 포함한다. User와 일대일 관계를 맺고, Data Base와 다대일 관계를 맺는다.

7.2.1.3. IoT

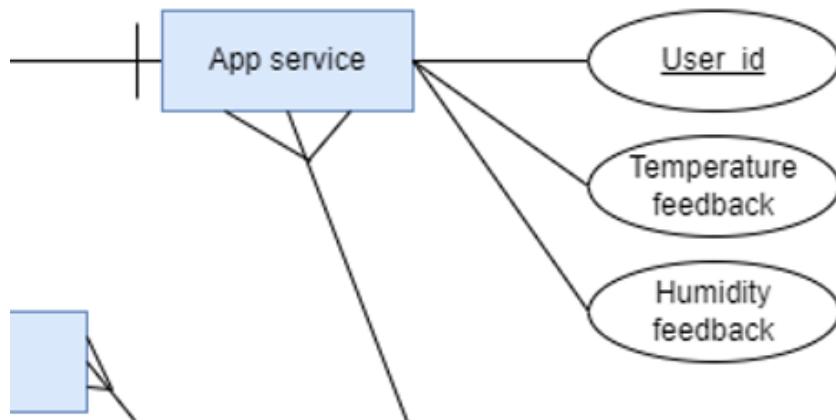
[Figure 30] ER diagram, Entity, IoT



IoT 엔티티는 사물 인터넷으로 연결된 집 안의 전자기기, 가전제품을 사용하는 사용자와 관련된 정보를 포함한다. 기기 아이디를 기본키로 가지며, 사용자 아이디, 현재 상태를 포함한다. User와 일대다 관계를 맺고, Data Base와 다대일 관계를 맺는다.

7.2.1.4. App service

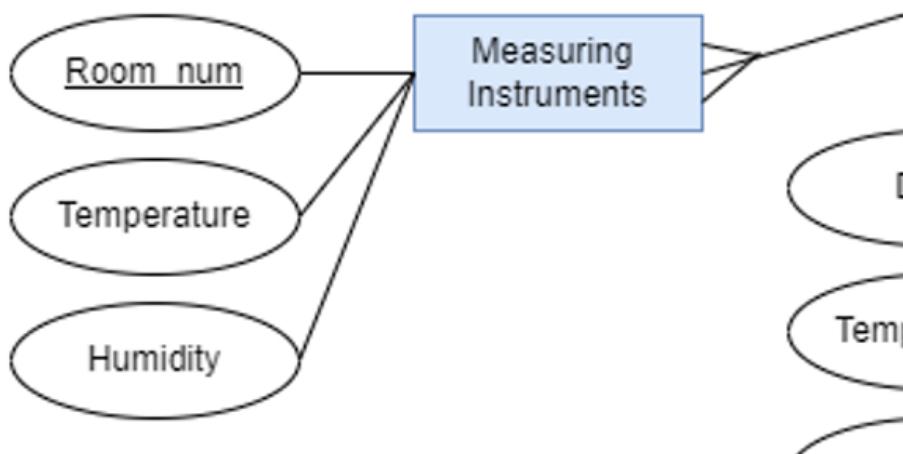
[Figure 31] ER diagram, Entity, App service



App service 엔티티는 사용자가 추천 온도/습도에 피드백을 입력한 정보를 포함한다. 사용자 아이디를 기본키로 가지며, 온도 피드백, 습도 피드백을 포함한다. User와 일대다 관계를 맺고, Data Base와 다대일 관계를 맺는다.

7.2.1.5. Measuring instruments

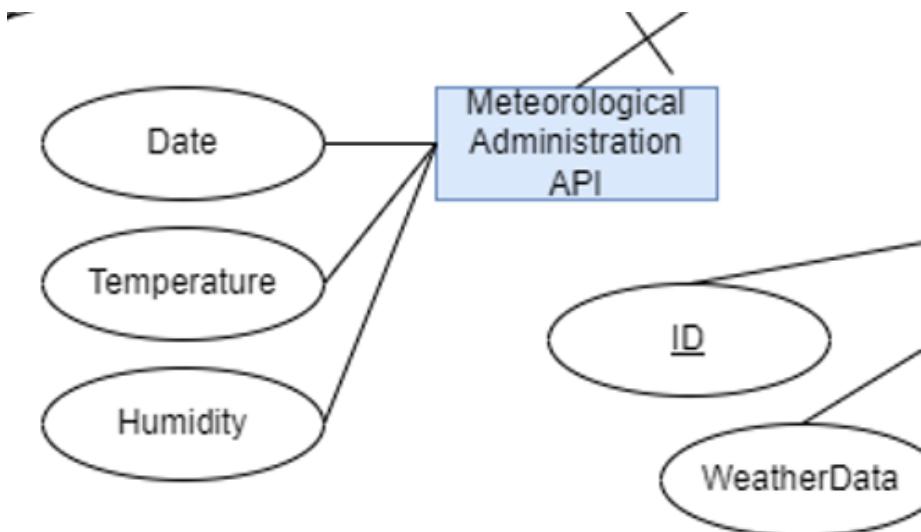
[Figure 32] ER diagram, Entity, Measuring instruments



온도/습도 측정기 엔티티는 각 방의 실내 온도와 습도 정보를 포함한다. 방 번호를 기본키로 가지며, 온도와 습도를 포함한다. Data Base와 대일 관계를 맺는다.

7.2.1.6. Meteorological administration API

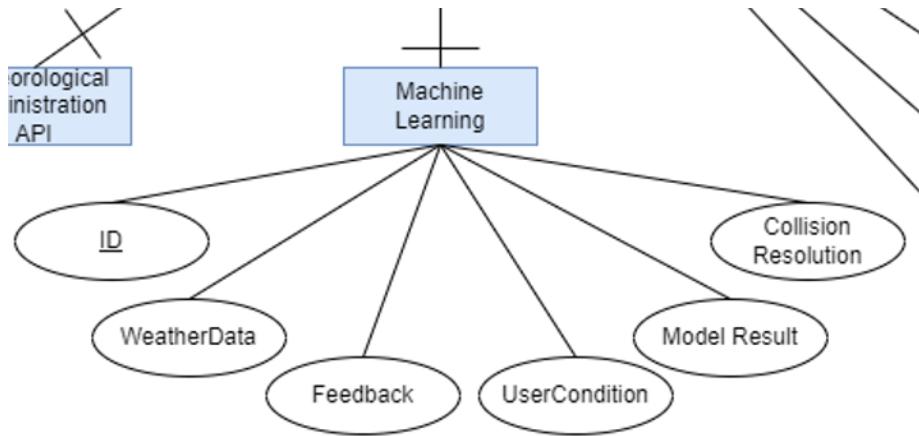
[Figure 33] ER diagram, Entity, Meteorological administration API



기상청 API는 해당 지역의 실외 온도와 습도 정보를 포함한다. 날짜와 시간에 따른 온도, 습도를 포함한다. Data Base와 일대일 관계를 맺는다.

7.2.1.7. Machine learning

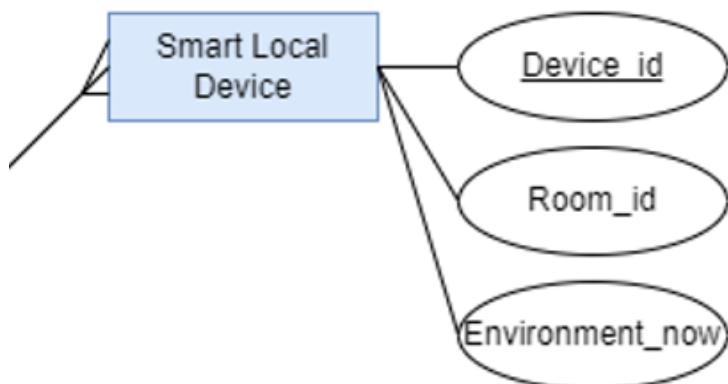
[Figure 34] ER diagram, Entity, Machine learning



Machine learning 엔티티는 각 사용자에게 상황에 따라 최적화 된 온도와 습도의 결과값, 그리고 공용 공간의 최적의 온도와 습도 값을 결과값으로 포함한다. 사용자의 아이디를 기본키로 가지고 날씨 데이터, 사용자 피드백, 사용자 상태, 모델 결과값, 충돌해소값을 포함한다. Data Base와 일대일 관계를 맺는다.

7.2.1.8. Smart local devices

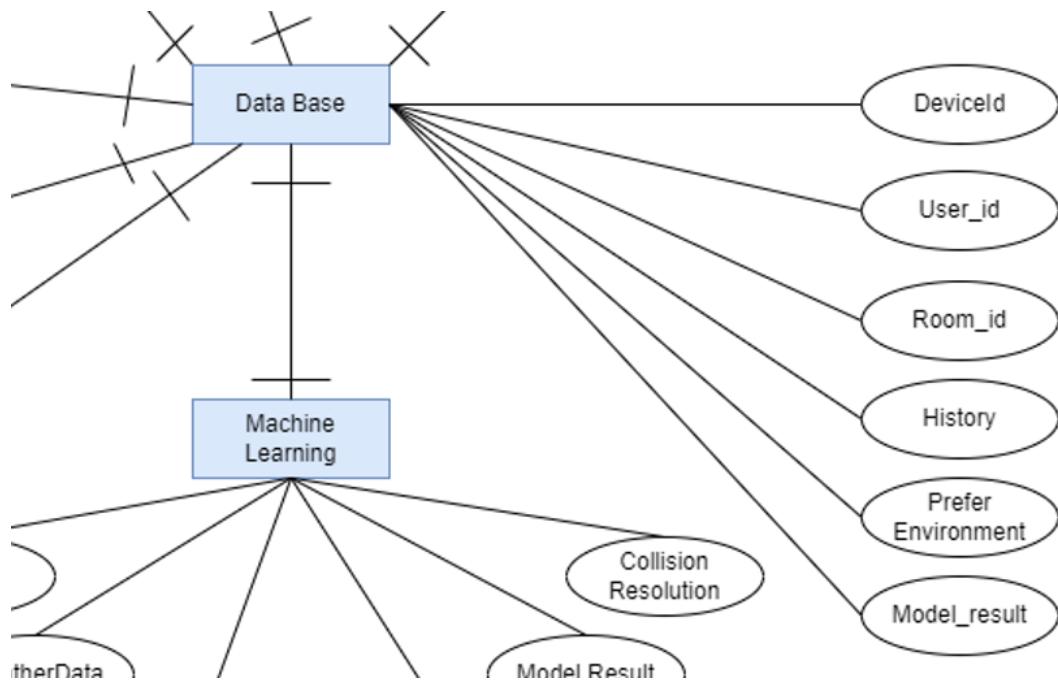
[Figure 35] ER diagram, Entity, Smart local devices



Smart local devices 엔티티는 집 안의 온도와 습도를 조절할 수 있는 기기로 현재 내부 환경 정보를 포함한다. 디바이스 아이디를 기본키로 가지고 방 아이디, 현재 환경을 포함한다. Data Base와 다대일 관계를 맺는다.

7.2.1.9. Data base

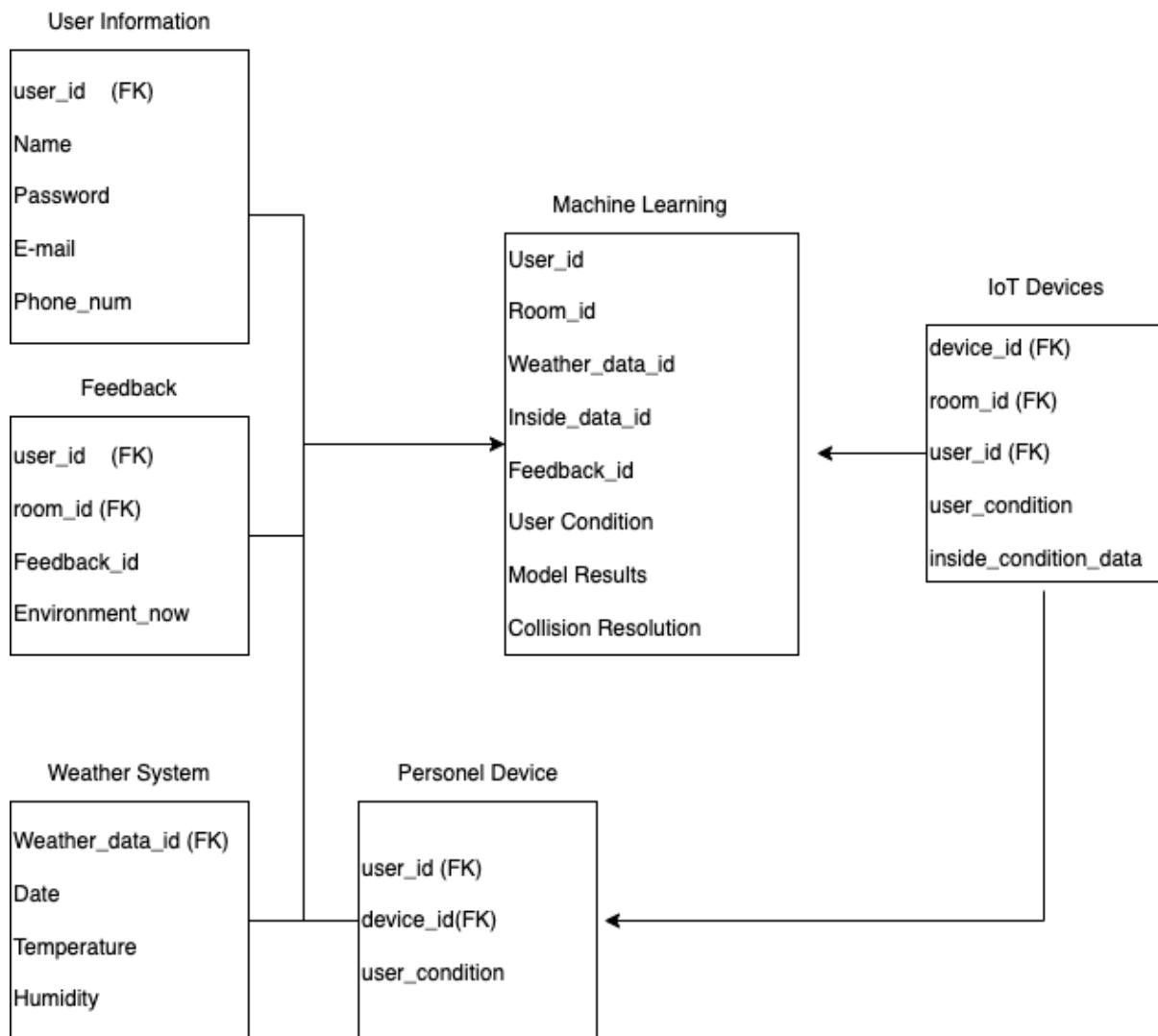
[Figure 36] ER diagram, Entity, Data base



Data base 엔티티는 7개의 엔티티에서 얻은 사용자와 관련된 정보, 날씨 데이터, 모델 결과 값 등 스마트홈의 불쾌지수 조절을 위한 모든 데이터를 포함한다. 디바이스 아이디, 사용자 아이디, 방 아이디, 데이터 기록(사용자 상태 데이터, 사용자 활동 데이터, 실내/외 날씨 데이터), 사용자 최적 환경, 모델 결과값을 포함한다. Smart watch, IoT, App service, Measuring instruments, Smart local device 엔티티들과 일대다 관계를 맺고, Meteorological administration API, Machine learning 엔티티들과 일대일 관계를 맺는다.

7.3. Relational Schema

[Figure 37] Relational Schema



7.4. SQL DDL

7.4.1 User

```
CREATE TABLE User
(
    id CHAR(20) NOT NULL,
    name CHAR(10) NOT NULL,
    password CHAR(20) NOT NULL,
    e_mail CHAR(30),
    phone_num CHAR(12) NOT NULL,

    PRIMARY KEY (id)
);
```

7.4.2 Smart_watch

```
CREATE TABLE Smart_watch
(
    watch_num INT NOT NULL,
    user_state VARCHAR NOT NULL,
    user_id CHAR(20) NOT NULL

    PRIMARY KEY (watch_num),
    FOREIGN KEY (user_id) REFERENCES User (id)
);
```

7.4.3 IoT

```
CREATE TABLE IoT
(
    device_id INT NOT NULL,
    environment_now VARCHAR NOT NULL,
    user_id CHAR(20) NOT NULL,
    PRIMARY KEY (device_id),
    FOREIGN KEY (user_id) REFERENCES User (id)
);
```

7.4.4 App_service

```
CREATE TABLE App_service
(
    user_id CHAR(20) NOT NULL,
    temperature feedback VARCHAR,
    humidity feedback VARCHAR
    PRIMARY KEY (user_id)
);
```

7.4.5 Measuring_instruments

```
CREATE TABLE Measuring_instruments
(
    room_num INT NOT NULL,
    temperature INT NOT NULL,
    humidity INT NOT NULL

    PRIMARY KEY (room_num)
);
```

7.4.6 Meteorological_administration_API

```
CREATE TABLE Meteorological_administration_api
(
    date DATE NOT NULL,
    temperature INT NOT NULL,
    humidity INT NOT NULL

    PRIMARY KEY (date)
);
```

7.4.7 Machine_learning

```
CREATE TABLE Machine_learning
(
    id CHAR(20) NOT NULL,
    weatherData VARCHAR NOT NULL,
    feedback INT NOT NULL,
    user_condition INT NOT NULL,
    model_result INT NOT NULL,
    collision_resolution INT NOT NULL

    PRIMARY KEY (id)
);
```

7.4.8 Smart_local_device

```
CREATE TABLE Smart_local_device
(
    device_id INT NOT NULL,
    room_id INT NOT NULL,
    environment_now INT NOT NULL

    PRIMARY KEY (device_id)
);
```

7.4.9 Date_base

```
CREATE TABLE Data_base
(
    device_id INT NOT NULL,
    user_id CHAR(20) NOT NULL,
    room_id INT NOT NULL,
    history VARCHAR NOT NULL,
    prefer_environment VARCHAR NOT NULL,
    model_result INT NOT NULL
);
```

8. Testing Plan

8.1. Objectives

이 챕터는 본 시스템의 테스트 계획을 개발 단계, 배포 단계, 그리고 사용자 테스트의 부문으로 나누어 기술한다. 이러한 테스트들은 시스템의 결점들을 찾아내서 시스템이 별다른 이상 없이 정상적으로 작동하는데 중요한 역할을 하고, 시스템에 대한 소비자들의 신뢰도를 높여준다.

8.2. Testing Policy

8.2.1. Development Testing

개발단계에서의 테스트는 시간과 비용을 줄일 수 있도록 개발 도중 일어날 수 있는 결점들을 방지하고 찾아내는 전략이 사용된다. 이 단계에서는 소프트웨어가 충분히 테스트되지 않았기 때문에 작동이 불안정하고, 시스템의 구성요소들 간의 충돌이 발생할 수 있다. 따라서 정적 코드 분석(static code analysis), 데이터 흐름 분석(data flow analysis), 메트릭스 분석(metrics analysis), 동료 코드 평가(peer code review), 코드 커버리지 분석(code coverage analysis) 등의 방법들이 개발 과정에서의 테스트에 사용된다.

8.2.1.1. Performance

본 시스템은 스마트 홈의 여러 기기 들로부터 수집한 데이터와 사용자의 피드백을 받고 이를 기반으로 사용자에게 최적의 불쾌지수 환경을 제공하는 것이 목표이다. 따라서, 사용자가 어플리케이션 상에서 기록하는 데이터들이 서버와 실시간으로 동기화 되어야 하고 이 기록들은 데이터 베이스 내에 저장되어야 한다. 또한 사용자 편의를 위해 어플리케이션의 초기 화면에서부터 각 기능별 화면까지의 응답속도는 4sec를 넘겨서는 안 된다. 이러한 요구사항들이 잘 지켜질 수 있도록 개발에 사용되는 IOS 기기들을 통해 충분한 테스트가 진행될 것이다.

머신 러닝에 필요한 데이터들은 스마트 홈 내에서 사물인터넷을 통해 연결된 기기들과 사용자가 소지한 스마트 디바이스를 통해 수집된다. 머신 러닝의 특성상 충분한 데이터가 지속적으로 확보되어야 하기 때문에 사물인터넷 기기들과 스마트 디바이스들이 시스템과 정상적으로 연결될 수 있는지 테스트한다. 또한 수집 과정에서 오류가 발생하여 이상치가 수집되어서는 안 되기 때문에 시스템의 정보 수집 기능이 정상적으로 작동하는지 확인한다.

8.2.1.2. Reliability

본 시스템이 정상적으로 작동되기 위해선 시스템의 구성요소들이 작동 과정에서 서로 충돌하여 오류를 일으키지 않아야 한다. 또한 각 구성요소들이 시스템에 추가될 때는 각 요소들이 제 기능을 완전히 수행할 수 있는지 테스트한다. 이러한 테스트 과정은 각 요소들이 개발되어 추가되는 과정에서 반복적으로 수행된다.

8.2.1.3. Security

본 시스템은 사용자의 여러 개인 정보들을 다루는 시스템이기 때문에 정보 보호는 매우 중요한 부분이다. 시스템의 어떤 정보도 정상적으로 허락되지 않은 이용자에 의해 열람되어서는 안 된다. 오직 시스템 관리자와 같은 권한이 있는 이용자들이 정보를 관리해야 한다.

시스템의 정보 보호를 위해 어플리케이션의 새로운 버전이 나올 때마다 시스템에 잠재적인 취약점이 발생하지 않았는지 검증해야 한다. 또한 주기적인 데이터베이스 점검을 통해 시스템에 치명적인 위험을 가할 수 있는 외부의 침입을 대비해야 한다.

8.2.2. Release Testing

“스마트 불쾌지수 조절 시스템”의 정식 출시를 위해선, 출시 전 사전 배포 테스트가 이루어져야 한다. 앞서 언급한 개발과정에서의 테스트들을 충분히 진행하고 소프트웨어가 제 기능을 다 한다고 판단될 때, 소수의 테스트 이용자에게 알파 버전의 소프트웨어를 배포하여 검증을 받는다. 이 과정에서도 개발 단계에서 진행하던 테스트는 계속 진행된다. 그 후 배포된 소프트웨어에 대한 테스터들의 에러 리포트나

피드백 등을 수렴하여 개발자들이 미처 찾아내지 못한 취약점들을 개선한다. 알파 버전의 배포 테스트로 소프트웨어가 추가적인 보완을 마친 뒤에는 더 많은 사용자들을 대상으로 베타 버전을 배포하여 최대한 결점이 없도록 시스템을 보완한다.

8.2.3. User Testing

위의 배포 테스트 뿐만 아니라, 실제 사용자들의 테스트도 진행되어야 한다. 본 테스트는 스마트 홈 시스템을 구축하고 있는 사용자들을 대상으로 시스템이 실제 상황에서도 잘 작동하는지 확인해야 한다. 예를 들어, 스마트 홈 시스템 사용자 10명을 모집하여 스마트 디바이스들과 사물인터넷 기기들이 시스템의 통제에 따라 제대로 작동되는지 테스트를 진행할 수 있다. 이들을 통해서 본 시스템의 주요 기능들인 정보수집, 머신 러닝, 피드백 반영이 실제 사용 환경에서도 제대로 작동하는지 확인할 수 있고, 오류가 발생한다면 이를 통해 시스템을 수정하고 보완할 수 있을 것이다.

8.2.4. Testing Case

테스트 케이스는 “스마트 불쾌지수 조절 시스템”이 앞서 언급한 성능(performance), 소프트웨어 신뢰도(reliability), 보안(security)의 세 측면을 시험할 수 있도록 만들 계획이다. 각 측면별로 10개의 테스트 케이스를 만들어 시스템을 검증하고, 이를 기반으로 시스템 평가서를 작성할 것이다.

9. Development Plan

9.1. Objectives

이 챕터에서는 시스템 및 어플리케이션 개발에 사용된 외부 기술 및 개발 환경들을 기술한다.

9.2. Frontend Environment

9.2.1. Adobe Photoshop (UI/UX Design)

[Figure 38] Adobe Photoshop logo



어도비사가 윈도우와 macOS용으로 개발하고 배급한 raster graphics editor이다. 이 프로그램은 UX / UI 디자인을 위한 레이아웃과 아이콘 제작에 사용된다.

9.2.2. Adobe Xd (UI/UX Design)

[Figure 39] Adobe Xd logo



웹 및 모바일 앱 설계를 위한 일체형 UX/UI 솔루션 프로그램이다. 다양한 플랫폼에서 상호작용 프로토타입을 제공하여 팀원 간의 커뮤니케이션을 촉진하고 피드백을 신속하게 반영할 수 있다.

9.2.3. Flutter

[Figure 40] Flutter logo



Flutter는 구글에서 제공하는 크로스 플랫폼 개발 환경이다. Flutter를 통해 Highlight의 구현된 소스코드는 Android OS, iOS의 두 환경에서 구동 가능한 제품으로 컴파일 될 수 있다. 또한 Flutter의 사용은 웹 브라우저, Windows, Mac, Linux 등의 수많은 환경에서 가능하기 때문에 다수의 개발자가 통일된 환경에서 컴파일 하는 것이 가능하다는 장점이 있다.

9.3. Backend Environment

9.3.1. 깃허브



[Figure 41] Github logo

깃허브는 소프트웨어 개발 협업을 용이하게 하는 툴이다. 이는 팀원 간 개발 과정 상에서의 소통을 용이하게 하고, 히스토리 및 브런치 관리를 통해서 아주 용이하게 소프트웨어 개발을 관리할 수 있다. Team4는 깃허브를 이용하여 스마트 불쾌지수 조절 시스템의 개발을 조직하고 관리할 것이다.

9.3.2. Visual Studio Code



[Figure 42] Visual Studio Code logo

Visual Studio Code는 Microsoft에서 만든 코드 컴파일러로, 다양한 언어를 지원해주는 장점이 있다. 그에 더해, 깃허브와의 연결이 용이하고 리눅스 환경을 지원해주어서 백엔드와 프론트 엔드간의 효과적인 개발을 용이하게 해줄 수 있다.

9.3.3. Tensorflow



[Figure 43] TensorFlow logo

Tensorflow는 서버나 웹 등 다양한 환경에서 언어나 플랫폼 관계없이 모델을 쉽게 학습시키고 배포할 수 있다. Tensorflow에서 머신러닝을 개발하고 학습시켰다.

9.3.4. MySQL Workbench



[Figure 44] MySQL Workbench logo

MySQL Workbench는 데이터베이스를 실제로 구현하는데 사용한다. 테이블과 관계 다이어그램을 자동으로 그려주고, SQL DDL 파일을 입력하면 바로 테이블 생성이 가능하다.

9.3.5. Spring Boot



[Figure 45] Spring Boot logo

Spring Boot는 내장 웹 서버에서 백엔드와 프론트엔트 개발을 하여 웹 프로그래밍이 가능하다. 자주 사용하는 라이브러리의 버전 관리가 자동으로 이루어지고 실행 가능한 JPA로 개발이 가능하다.

9.4. Constraints

“스마트 불쾌지수 조절 시스템”은 앞서 작성된 요구사항 명세서와 본 디자인 명세서에 기술된 내용들을 바탕으로 구현될 예정이다. 그 외 세부적인 구현사항들은 개발자의 재량으로 구현될 것이며, 이는 아래의 제약사항들을 지키는 수준에서 구현될 예정이다.

- 사용자는 시스템이 제공하는 수치를 임의적으로 조절할 수 있어야 한다.
- 시스템은 사용자의 동의 없이 사용자의 실내외 데이터를 수집해서는 안 된다.
- “스마트 불쾌지수 조절 시스템”의 어플리케이션의 초기 버전의 용량은 200MB를 넘지 않아야 한다. 단, 추후 시스템에 새로운 기능이 추가될 경우에는 200MB를 초과할 수 있다.
- 시스템의 소스 코드와 데이터베이스는 사용자 정보 보호를 위해 엄격히

관리되어야 한다.

- 본 시스템은 오픈소스 소프트웨어들을 최대한 활용하여 개발한다.
- 시스템의 지속적인 유지 보수를 통해 사용자가 항상 높은 성능을 누릴 수 있도록 한다.
- ‘스마트 불쾌지수 조절 시스템’은 IOS 환경에서 모든 기능을 수행할 수 있어야 한다.

9.5. Assumptions and Dependencies

본 시스템은 IOS 기기를 기반으로 설계되었다. 시스템은 IOS 15.4.1 기반으로 개발됨을 전제한다. 따라서 IOS 15.4.1 이전 버전의 IOS 기기나 Android OS와 같이 다른 OS를 사용하는 경우에는 시스템이 동작하지 않을 수 있다. 또한, 네트워크를 통해 시스템을 작동시키기 때문에 네트워크 환경이 갖추어 지지 않은 곳에서는 사용자가 시스템을 동작할 수 없다.

본 시스템은 사용자로부터 수집된 데이터를 기반으로 작동하기 때문에 데이터 수집에 대한 권한을 요구한다. 사용자가 권한을 부여하지 않을 경우 시스템이 본래의 모든 기능을 제공하지 못할 수 있다.

10. Supporting Information

10.1. Software Design Specification

이 소프트웨어 명세서는 IEEE Recommendation (IEEE Recommended Practice for Software Design Description, IEEE-Std-1016). 서식을 따라 제작되었다.

10.2. Document History

[Figure 46] Document History

Date	Version	Description	Writer
2022/05/06	0.1	Style and overview	Gwiwan Go
2022/05/13	1.0	Addition of 1, 4, 6, 8	Minseong Kim
2022/05/14	1.0	Addition of 5, 7, 9	Junwoo Lim
2022/05/15	1.0	Addition of 4,6,9 / layout	Haeun Oh