AI 자동 환기 시스템

소프트웨어공학개론

Team 1

2018311913 김서윤

2019314658 박종은

2017311753 이재현

2019311851 박태희

Contents

1. Preface

본 디자인 명세서는 본 문서의 독자들을 위한 간략한 정보들, 개요, 범주, 목적을 기술하고 있다.

1.1. Readership

본 디자인 명세서는 AI 자동환기 기기에 대한 설명으로 이루어져 있다. 본 문서는 2022년 1학기 소프트웨어 공학 수업의 학생, 조교, 교수를 주된 이용자로 상정한다.

1.2. Scope

본 디자인 명세서는 AI 자동환기 기기에 필요한 정보들을 제공해 준다. 이 정보에는 시스템 아키텍처, 테스트 케이스, 개발 계획 등이 포함된다.

1.3. Objective

본 디자인 명세서의 주요 목적은 AI 자동환기 기기의 전반적인 상황과, 기술적인 설명을 제공하고, 또한 개발에 필요한 디자인 요소를 묘사함에 있다. 이는 과거 작성한 요구사항 명세서를 구체화하여 시스템을 여러 관점에서 보는 아키텍처가 존재한다. 본 문서를 활용하여 디자인과 구조를 명확히 하고, 프론트 엔드와 백엔드의 아키텍처, 유저 케이스, 다이어그램 등을 보여준다.

1.4. Structure

1) Preface: 본 문서의 독자들을 위한 정보, 범주, 구조를 간단하게 설명한다.
2) Introduction: 디자인 명세서에 명시될 정보들의 특징을 설명하고, 레퍼런스를 기록한다.
3) System Architecture - Overall: 개괄적인 아키텍처를 use case, sequence, context diagram 등을 활용하여 기술한다.
4) System Architecture - Frontend: 프론트엔드의 아키텍처를 기술한다.
5) System Architecture - Backend: 백엔드의 아키텍처를 기술한다.
6) Database Design: 데이터베이스 디자인에 대해 기술한다.
7) Testing Plan: 테스트 계획에 대해 기술한다.
8) Developing Plan: 개발에 사용된 툴과 여러 제약 및 규칙에 대해 기술한다.

2. Introduction

2.1. Objectives

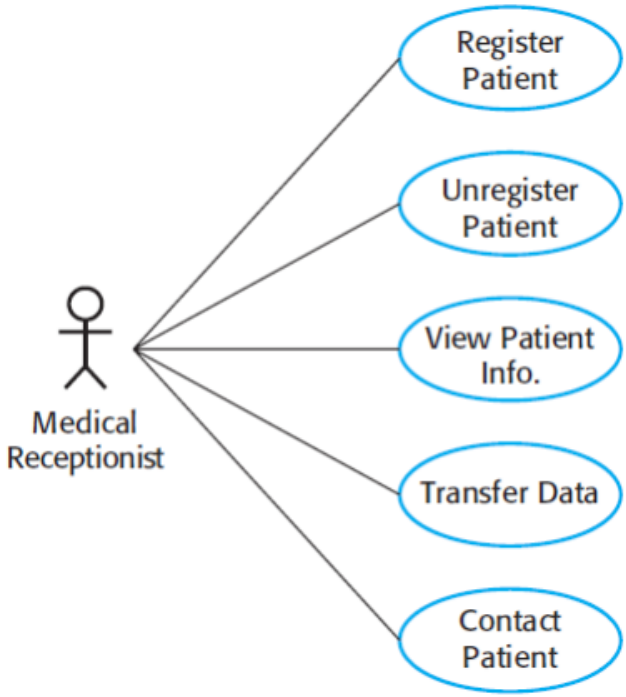
본 챕터는 본 디자인 명세서에서 활용된 툴과 다이어그램들이 설명되어 있다.

2.2. Applied Diagrams

2.2.1. UML

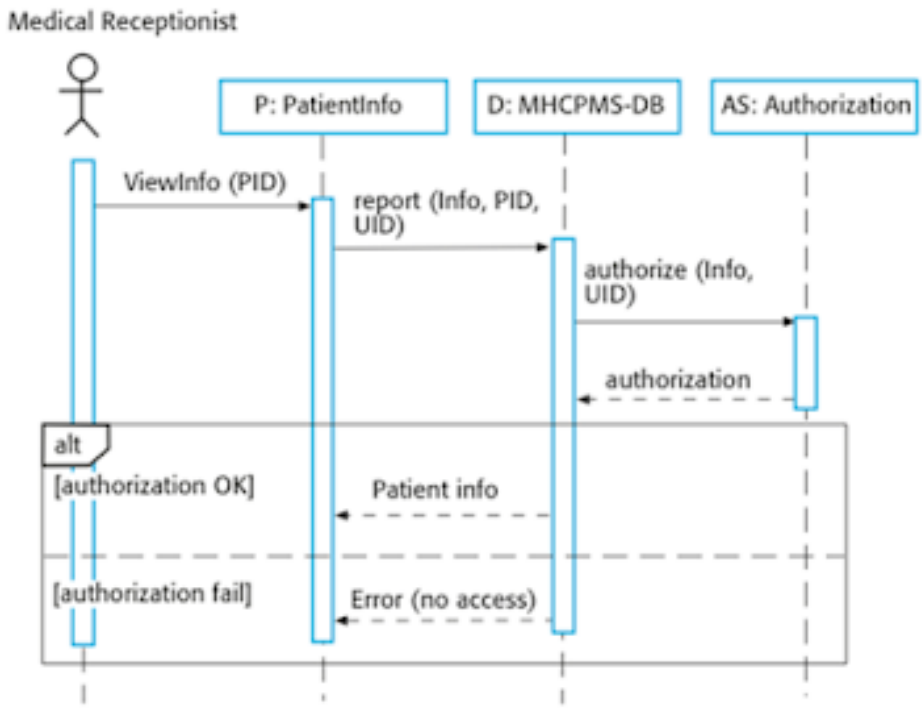
UML이란 통합 모델링 언어(Unified Modeling Language)로, 소프트웨어를 모델링 하는 데 사용되는 표준화된 모델링 언어를 의미한다. UML은 특정 작업, 시스템을 구성하는 구성 요소와 각 요소끼리 상호작용하는 내용을 도표를 활용하여 전체적인 구조를 보여주는 역할을 한다. 또한 이는 구성 요소 간의 관계를 문서화하는 데 활용되는 언어이다. 크게 Structure UML Diagram과 Behavioral UML Diagram으로 두 가지로 나뉜다.

2.2.2. Use Case Diagram



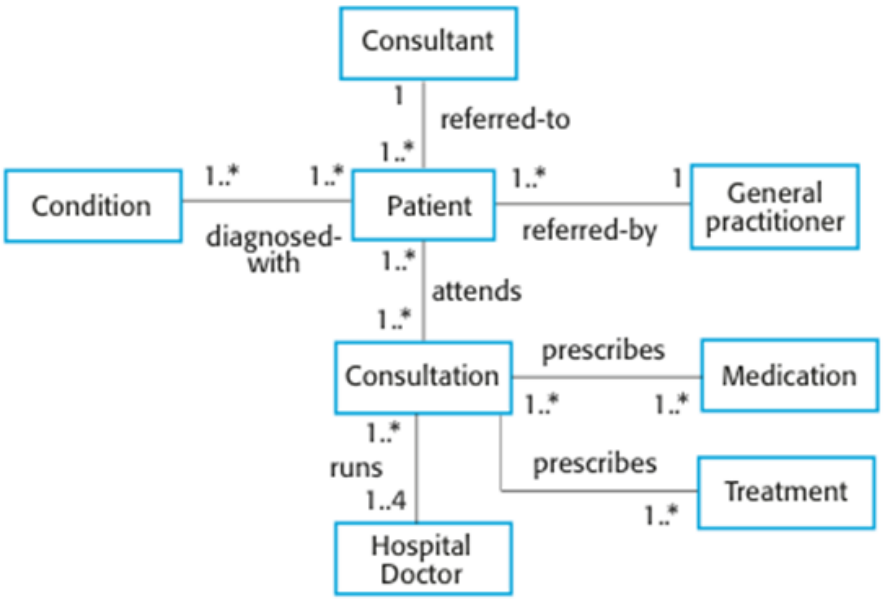
Use Case Diagram은 본 시스템의 주요 사용자의 요구사항에 따른 사용 예시에 대한 Diagram이다. 해당 Diagram을 통해 시스템의 전반적인 설계와 상호작용들을 시각적으로 확인할 수 있다. 해당 Diagram의 구성으로는 크게 Actor, Scope, Scope에 속하는 Use case가 존재한다. Actor란 본 기기에서 사용자이며, Scope이란 선을 통해 시스템의 주요 기능을 볼 수 있는 요소이다. 또한 Scope에 속하는 Use Case란 실제로 사용자가 해당 기능을 사용할 때 마주치는 상황들을 의미한다.

2.2.3 Sequence Diagram



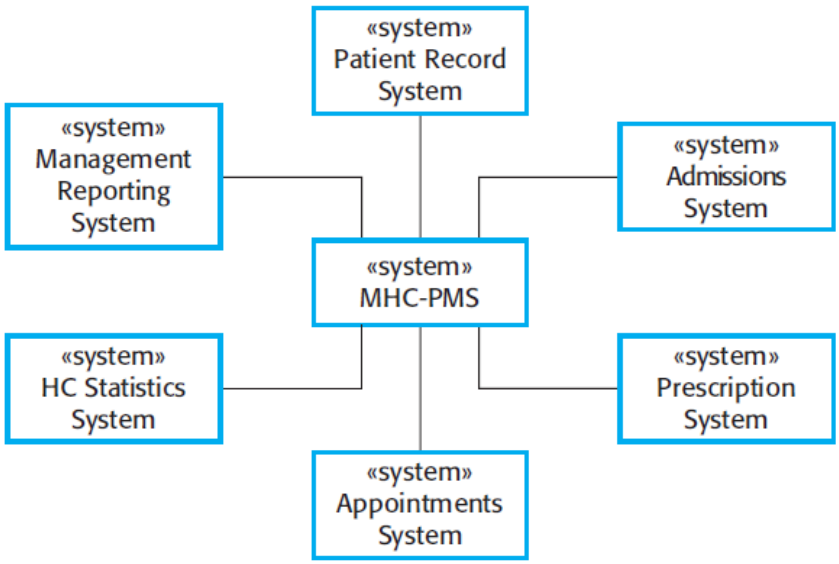
Sequence Diagram이란 시간에 따른 진행 상황을 보여주는 Diagram으로, 작업의 과정을 나타내거나 요소 간의 상호작용을 시각화하여 보여주는 도표이다. 이는 도표의 형태로 일관성 있고 개괄적으로 한눈에 파악할 수 있도록 하는 Diagram이며, 표준화된 작업 과정을 보여줄 수 있다. 객체나 프로세스는 세로 선을 통해, 그들 사이의 상호작용은 가로 화살표를 통해 표현된다.

2.2.4. Class Diagram



Class Diagram이란 시스템의 구조와 동작의 아키텍처를 시각적이고 직접적으로 보여주는 Diagram이다. 이는 객체 지향 모델링에서 사용될 수 있는데, 시스템의 개념적인 틀을 제작하는 데 활용된다. 이 Diagram에서 하나의 class는 사각형으로 표현되며, 사각형은 3개의 줄로 이루어져 있는데 맨 윗줄에는 이름이, 가운데에는 attribute가, 아래에는 method가 표시된다.

2.2.5. Context Diagram



Context Diagram은 시스템의 데이터 흐름을 보여주는 Diagram이다. 이를 통해 구성 요소 간의 상호작용을 확인할 수도 있으며, 시스템에 대한 이해력을 높일 수 있다. 이 Diagram은 모든 이해관계자 간의 관계를 보여주며, 시스템 전체를 하나의 프로세스로 보여준다. 가운데에 시스템을 표시하고, 주변으로 사용자와 영향을 미치는 요소들을 작성하여 표현한다.

2.3. Applied Tools

2.3.1. 파워포인트

해당 툴을 그림을 제작하는 데 사용하였다. 본 프로젝트에서 제작한 모든 Diagram 및 그림은 파워포인트를 활용하여 제작되었다. 또한 이는 후에 기술될 Microsoft Word와 높은 호환성을 보이는 강점을 지닌다.

2.3.2. Microsoft Word

해당 툴은 본 프로젝트의 요구사항 명세서 및 디자인 명세서 등 기본적인 문서 작업이 필요한 모든 공간에 사용되었다. 또한 워드를 활용, 구글 드라이브를 통해 팀원 간의 지속적인 교류를 통해 하나의 공통된 의견으로 취합하는 데에도 사용되었다.

2.4. Project Scope

기존 환기 시스템의 경우 개인이 직접 수동적인 방법을 활용하여 환기를 진행하거나, 유저가 원하는 시간에 자동으로 환기를 진행하는 방법에만 국한되어 왔다. 그러나 최근 코로나 사태 등으로 인하여 지속적으로 환기의 중요성에 대한 인지가 높아지고, 그에 맞춰 수요 또한 증가하고 있는 상황이다. 그런 상황 속에서 본 프로젝트는 유저가 해당 공간에 없더라도 자동으로 환기를 진행하고, 이를 넘어서 해당 유저가 원하는 환기 타입을 분석하여 AI를 활용, 각 유저에게 맞춤화된 환기 방법을 제공한다. 이를 통해 유저는 가장 적절한 시간과 상황에 따라 환기를 진행할 수 있게 된다.

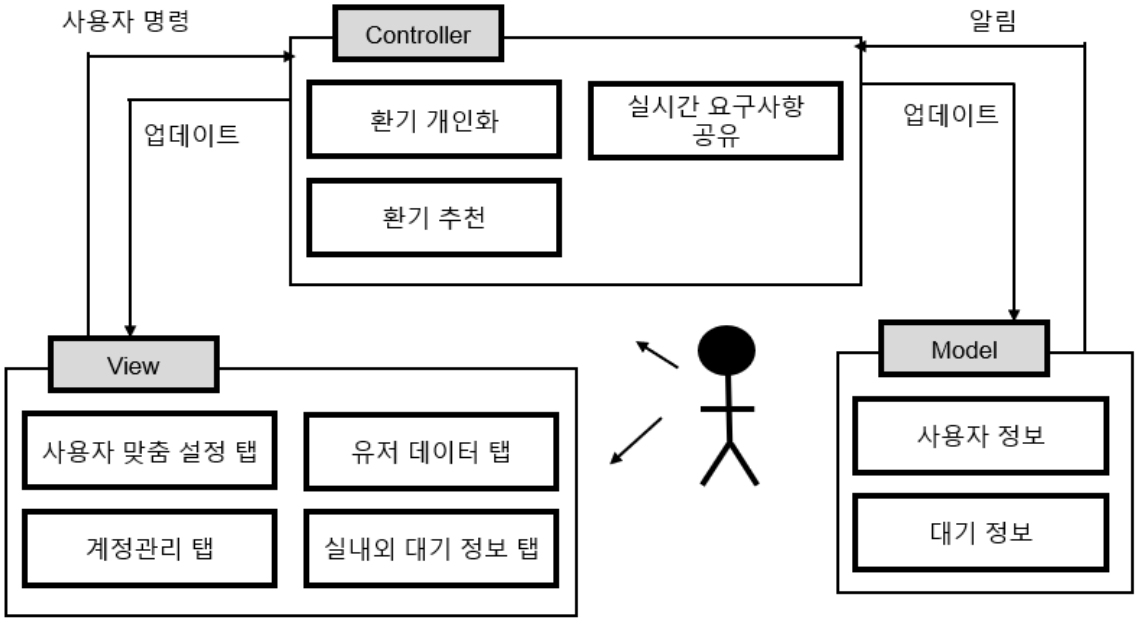
2.5. References

- I Skkuse, https://github.com/skkuse
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEXplore Digital Library http://ieeexplore.ieee.org/Xplore/guesthome.jsp
- Introduction to Software Engineering. "5. Requirements Engineering(new)."

3. System Architecture - Overall

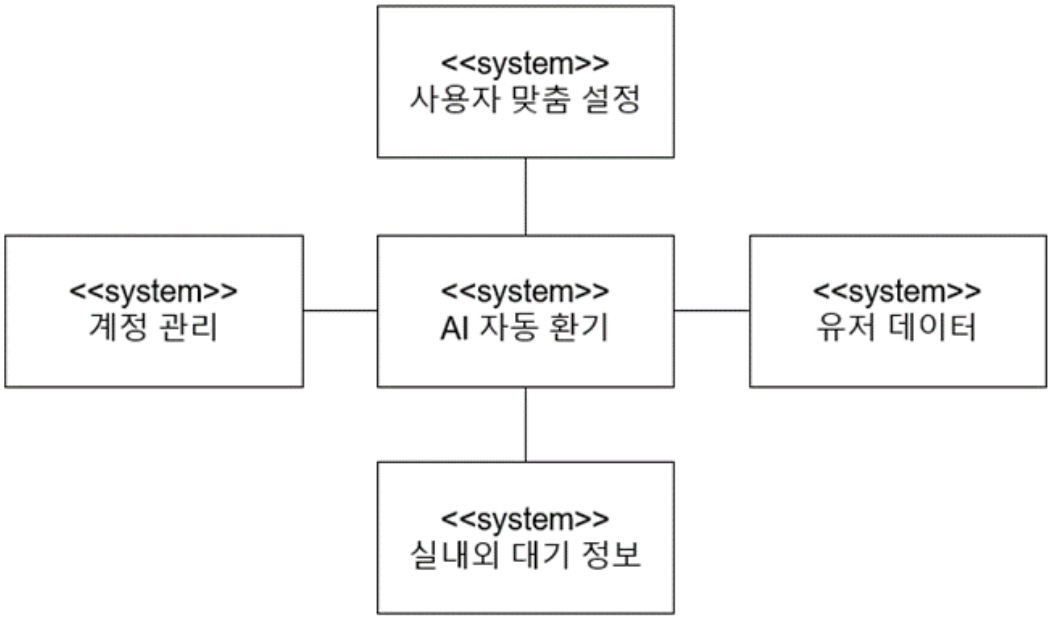
본 챕터에서는 시스템의 프론트엔드 디자인과 백엔드 디자인 등의 디자인 구성이 기술되어 있다.

3.1. System Organization

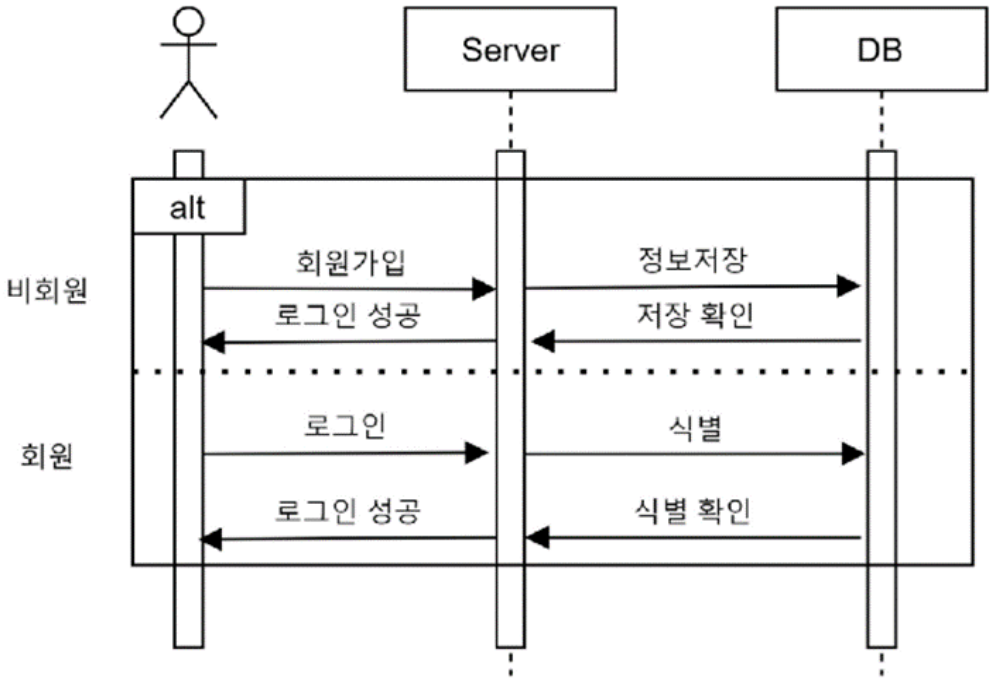


해당 시스템은 client-server model의 구조를 따르고 있다. 이를 MVC 디자인을 통해 표현하였고, 사용자가 프론트엔드인 기기의 view를 활용하여 시스템의 조작을 controller에 명령하면, controller는 해당 명령을 백엔드의 model을 활용, 데이터를 구성하여 view에 반영하여 사용자에게 보여준다. 추가적으로 백엔드를 통해 유저 데이터에 대한 DB를 구성한다.

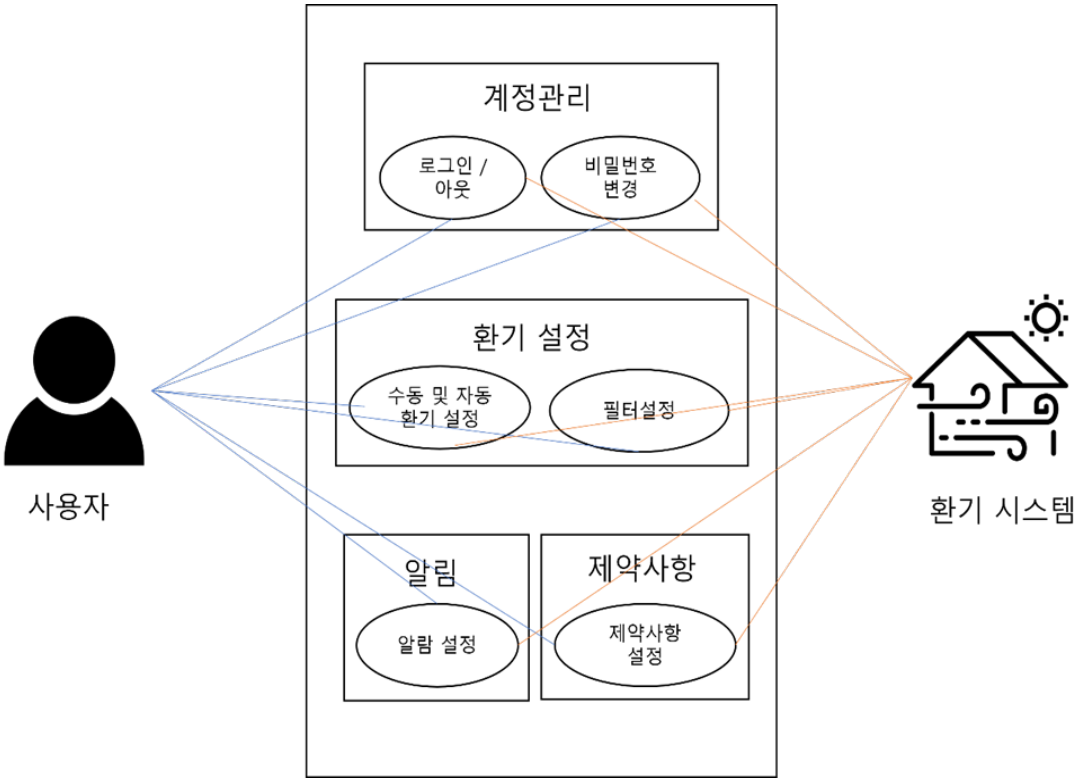
3.1.1. Context Diagram



3.1.2. Sequence Diagram



3.1.3. Use Case Diagram



4. system Architecture – Frontend

4.1. Objectives

System architecture 에서 사용자 인터페이스에 해당하는 frontend 시스템을 이루는 component들의 구성을 class diagram과 sequence diagram으로 도식화 및 설명한다.

4.2. subcomponents

4.2.1. 로그인 및 사용자 정보 등록

4.2.1.1. Attributes

해당 object 가 가진 attribute 는 다음과 같다.

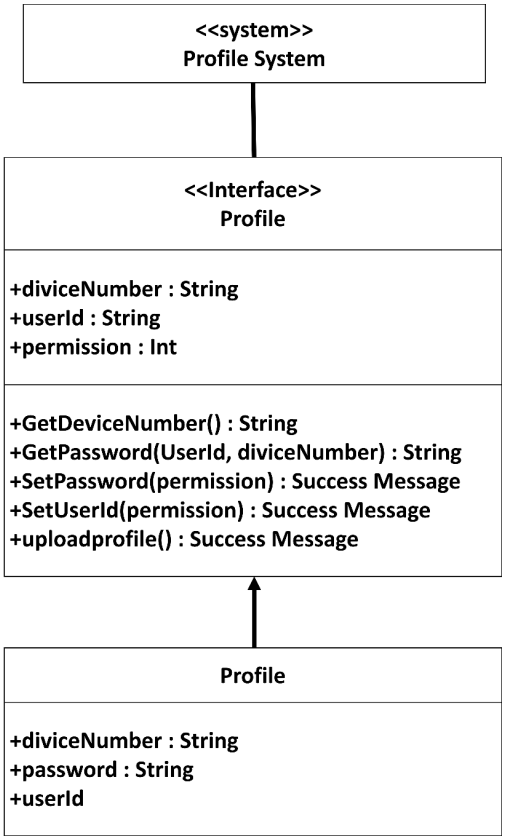
- deviceNumber : 로그인 시 필요한 기기 번호이다
- password : 로그인 시 필요한 비밀번호이다
- userId : 유저의 아이디이다

4.2.1.2. Methods

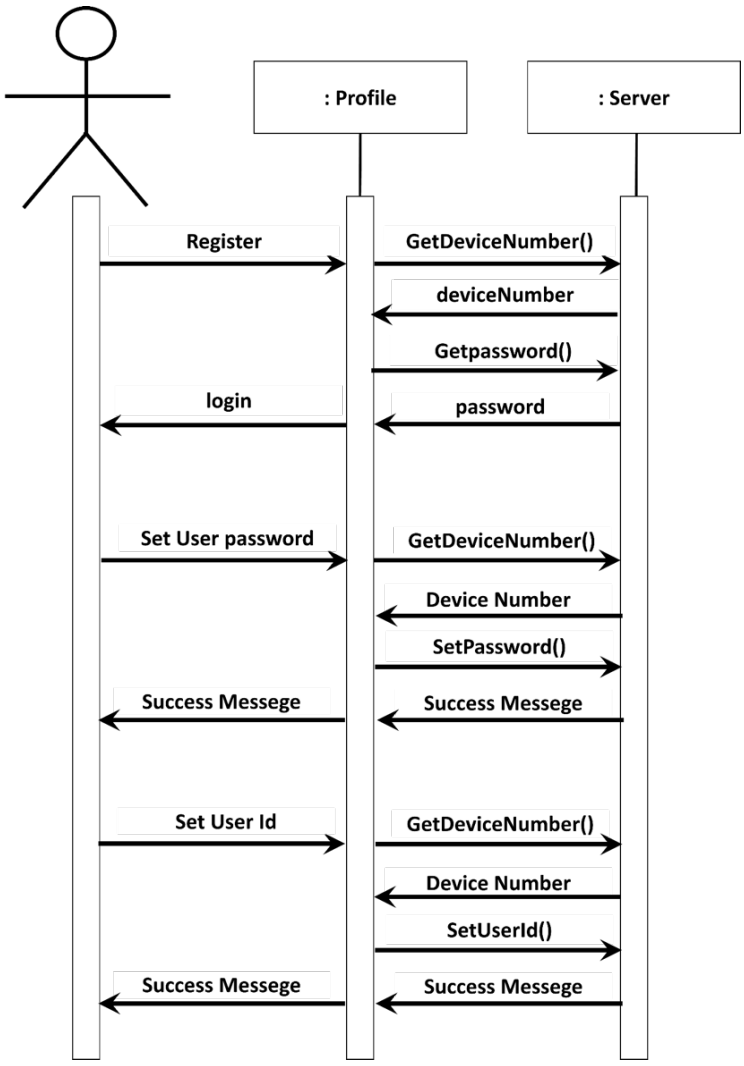
해당 object 가 가진 methods 는 다음과 같다

- GetDeviceNumber()
- GetPassword()
- SetPassword()
- SetUserId()
- Uploadprofile()

4.2.1.3. Class Diagram



4.2.1.4. Sequence Diagram



4.2.2. 메인 페이지

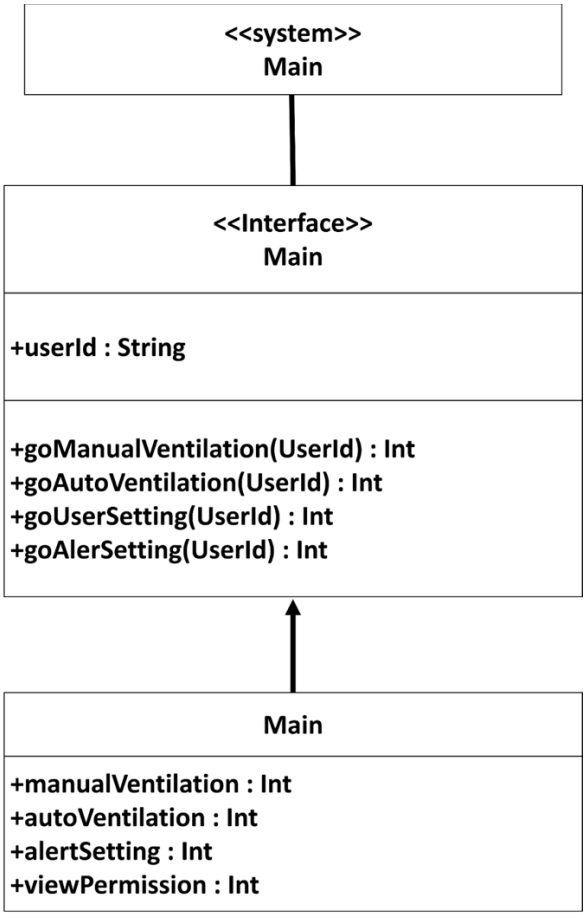
4.2.2.1. Attributes

- userId : 현재 사용자의 id 이다
- manualVentilation : 수동환기에 대한 설정을 할 수 있는 버튼이다
- autoVentilation : 자동환기에 대한 설정을 할 수 있는 버튼이다
- alertSetting : 알람에 대한 설정을 할 수 있는 버튼이다
- viewPermission : 메인 페이지에 접근할 수 있는 권한 여부를 나타낸다.

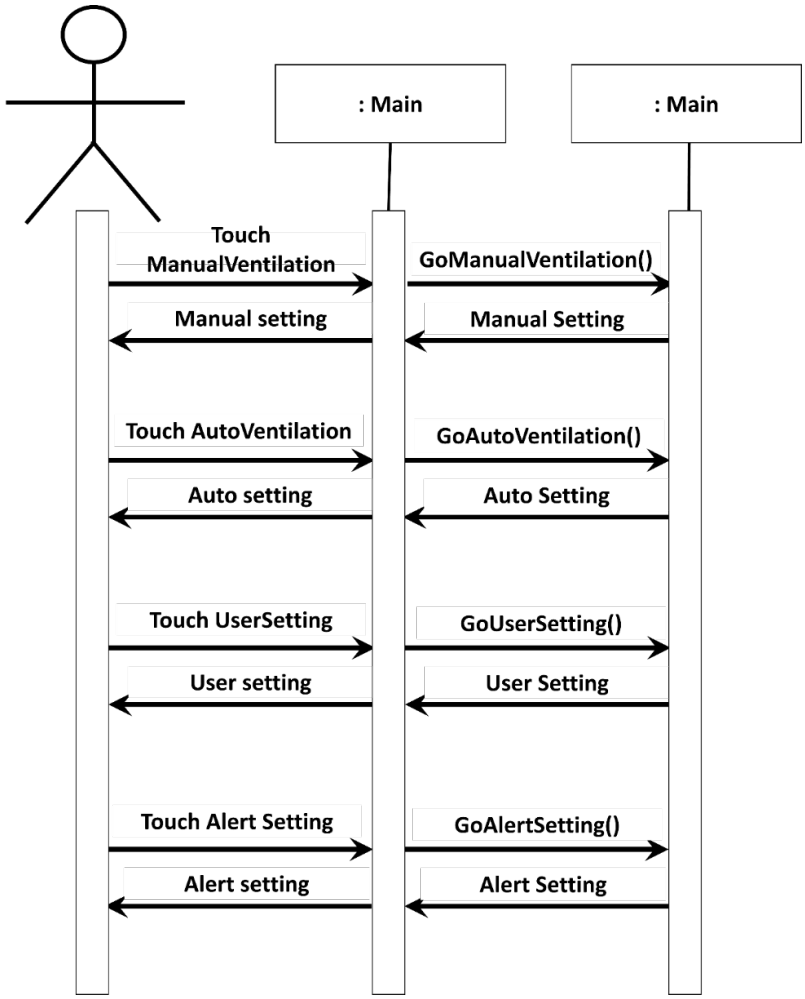
4.2.2.2. Methods

- goManualVentilation()
- goAutoVentilation()
- goUserSetting()
- goAlertSetting()

4.2.2.3. Class Diagram



4.2.2.4. Sequence Diagram



4.2.3 수동 환기 설정

4.2.3.1. Attributes

해당 object가 가진 attribute 는 다음과 같다

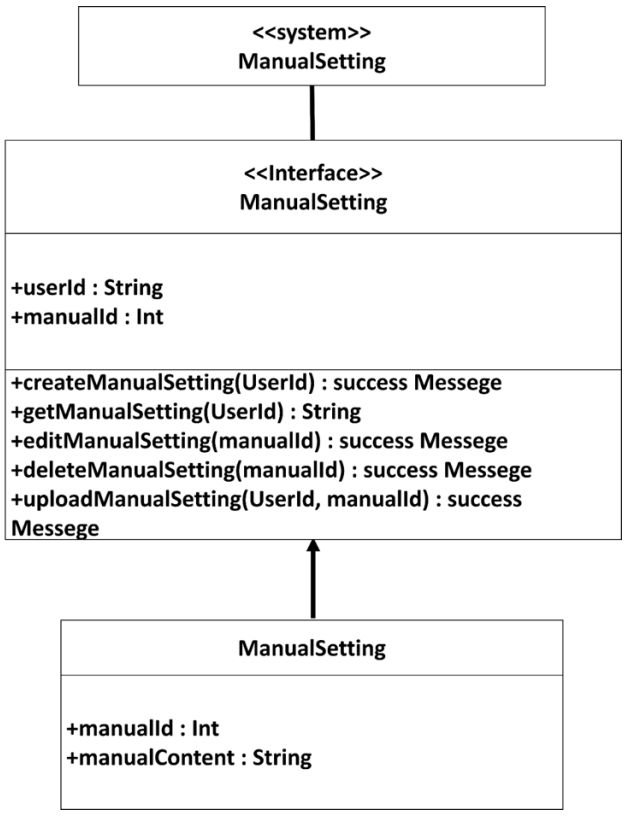
- userId : 현재 사용자의 id 이다
- manualId : 수동 환기 설정의 id 이다.
- manualContent : 현재 사용자가 설정한 수동 환기 설정 내용이다

4.2.3.2. Methods

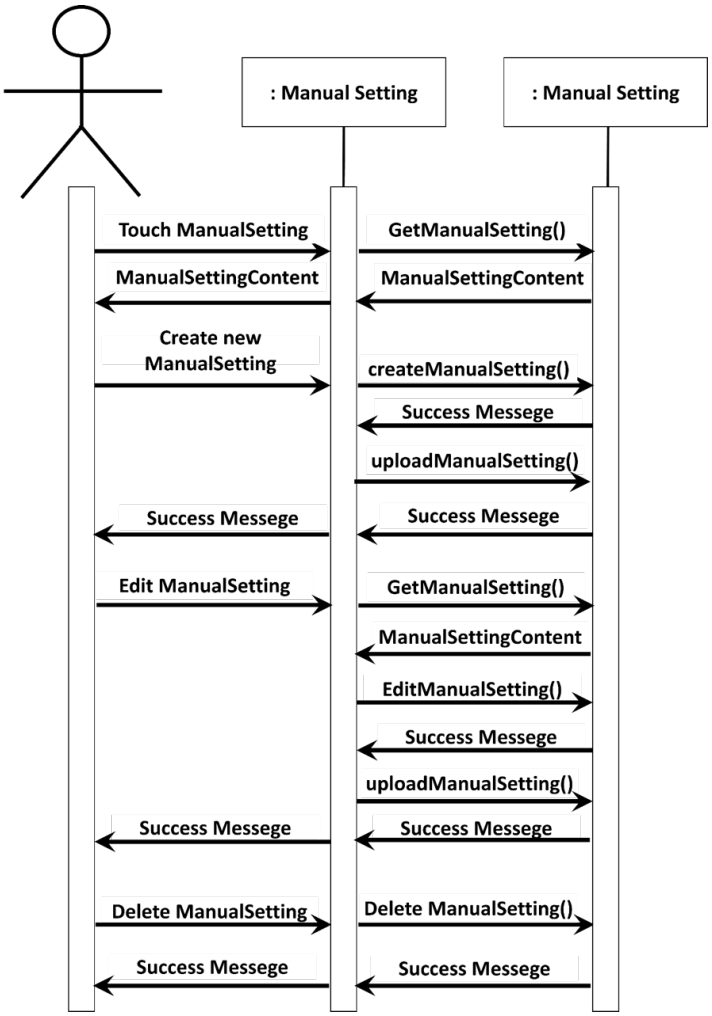
해당 object 가 가진 methods는 다음과 같다

- createManualSetting()
- getManualSetting()
- editManualSetting()
- deleteManualSetting()
- uploadManualSetting()
- getManualContent()

4.2.3.3. Class Diagram



4.2.3.4. Sequence Diagram



4.2.4. 자동 환기 설정

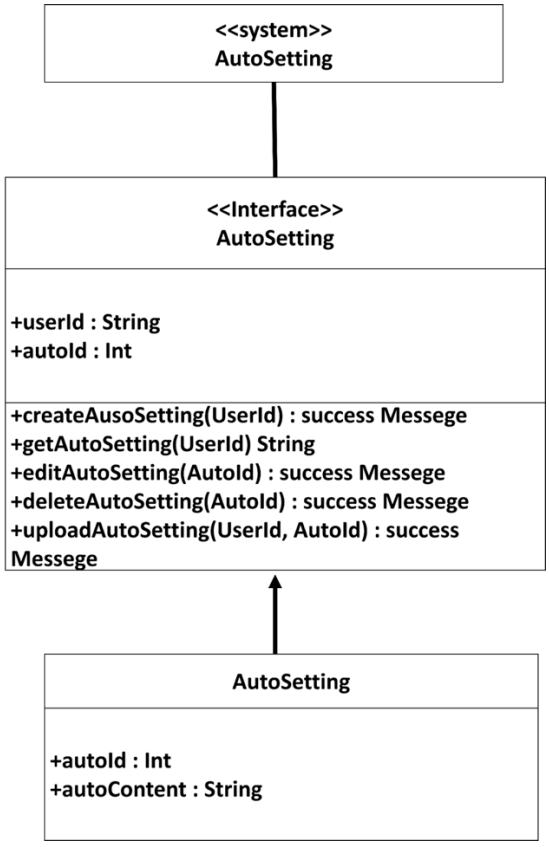
4.2.4.1. Attributes

- userId : 현재 사용자의 id 이다
- autoId : 자동 환기 설정의 id 이다.
- autoContent : 현재 사용자가 설정한 자동 환기 설정 내용이다

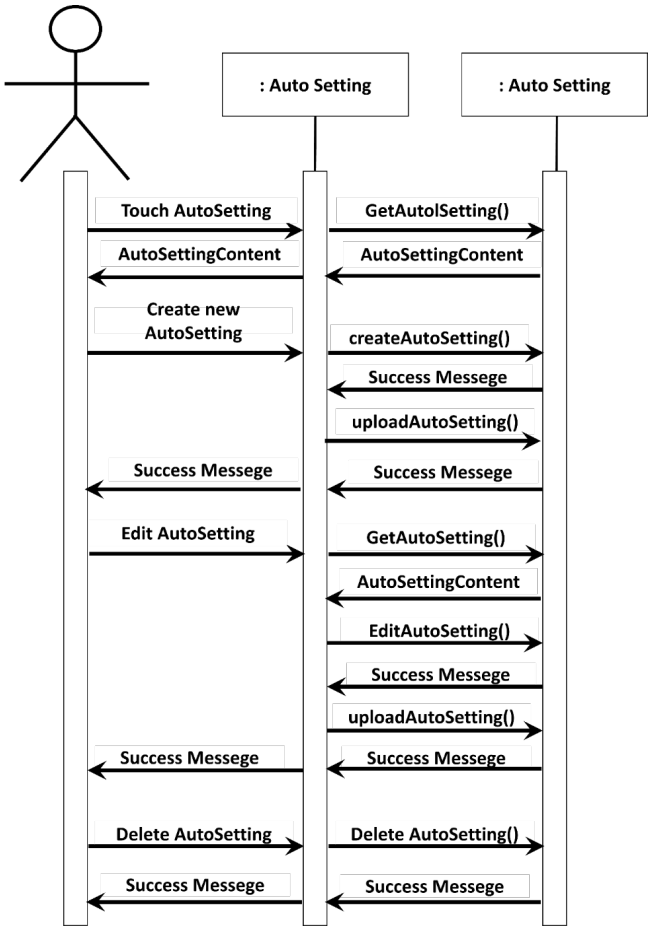
4.2.4.2. Methods

- createAutoSetting()
- getAutoSetting()
- editAutoSetting()
- deleteAutoSetting()
- uploadAutoSetting()
- getAutoContent()

4.2.4.3. Class Diagram



4.2.4.4. Sequence Diagram



4.2.5. 사용자 설정

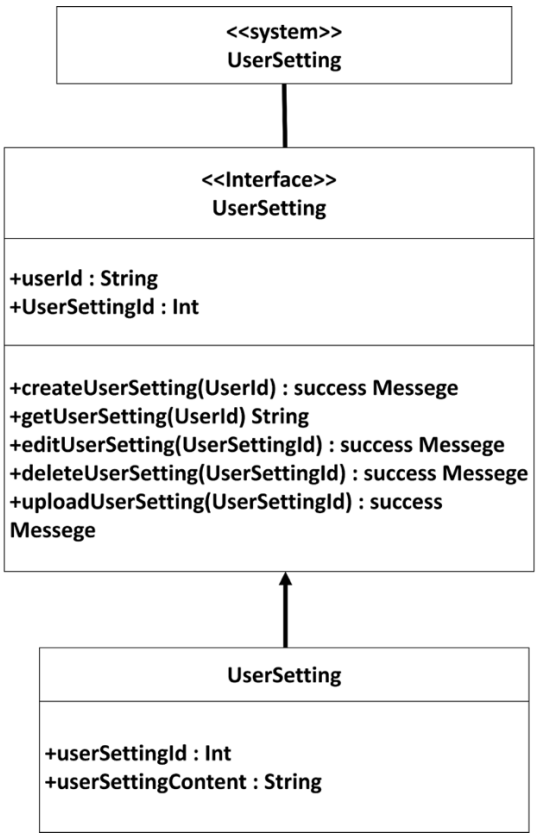
4.2.5.1. Attributes

- userId : 현재 사용자의 id 이다
- userSettingId : 사용자 설정의 id 이다.
- userSettingContent : 현재 사용자가 설정한 유저 환경 설정 내용이다.

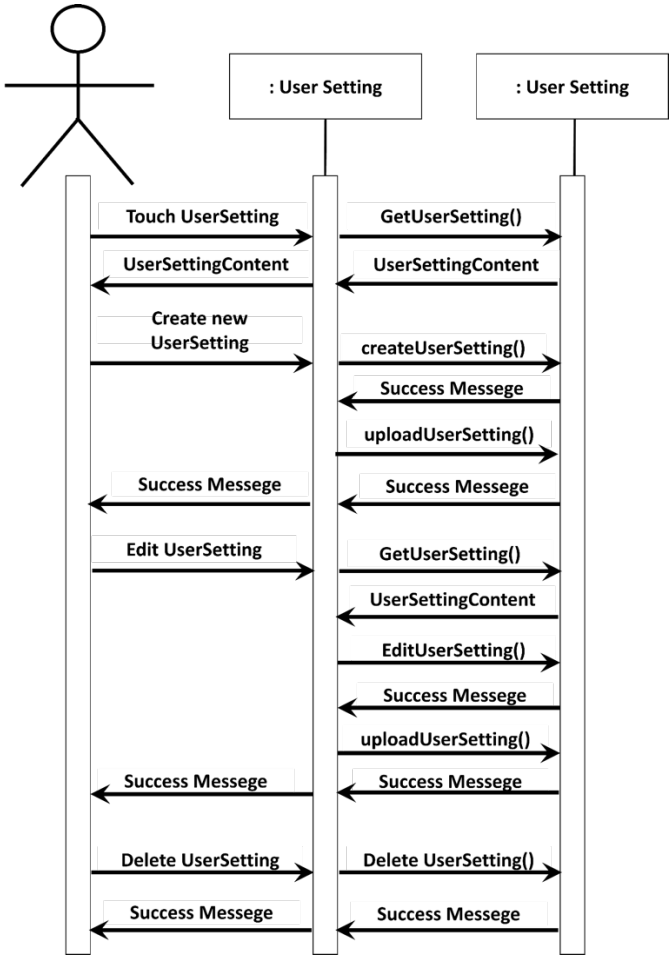
4.2.5.2. Methods

- createUserSetting()
- getUserSetting()
- editUserSetting()
- deleteUserSetting()
- uploadUserSetting()
- getUserContent()

4.2.5.3. Class Diagram



4.2.5.4. Sequence Diagram



4.2.6 알림 설정

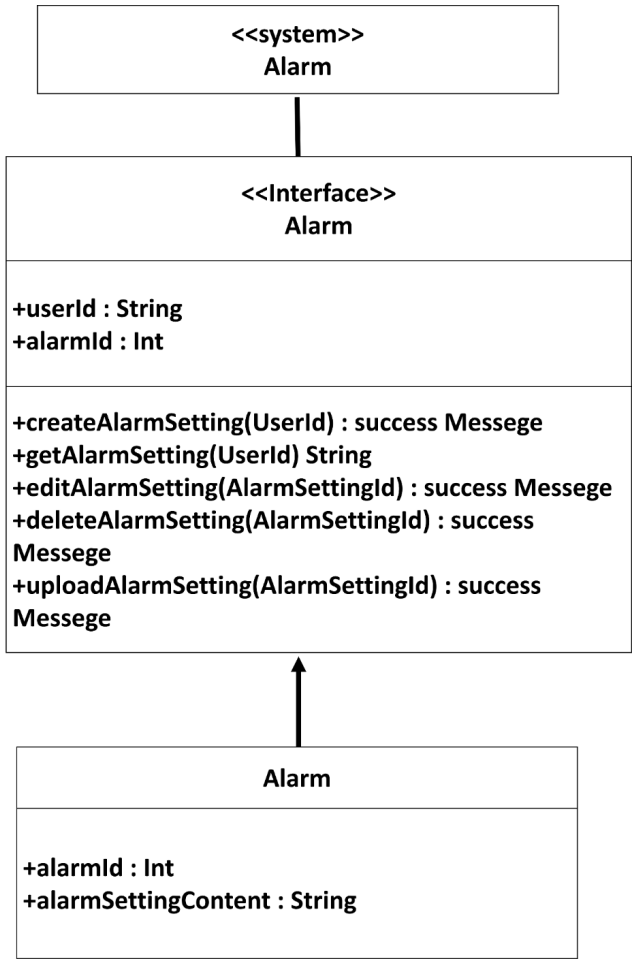
4.2.6.1. Attributes

- userId : 현재 사용자의 id 이다
- alarmId : 알람 설정의 id 이다.
- alarmSettingContent : 현재 사용자가 설정한 알람 설정 내용이다

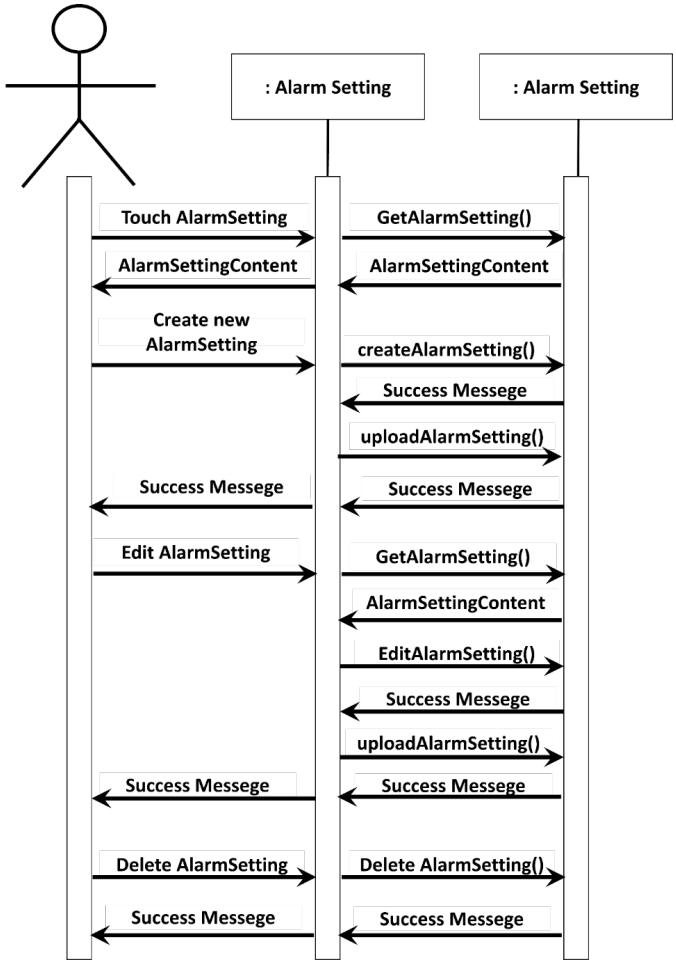
4.2.6.2. Methods

- createAlarmSetting()
- getAlarmSetting()
- editAlarmSetting()
- deleteAlarmSetting()
- uploadAlarmSetting()
- getAlarmContent()

4.2.6.3. Class Diagram



4.2.6.4. Sequence Diagram



5. System Architecture – Backend

5.1. Objectives

이 챕터에서는 System Architecture 중에서 Firebase를 통해 구현할 백엔드 시스템의 구조와 기능에 대해서 설명한다.

5.2. Overall Architecture

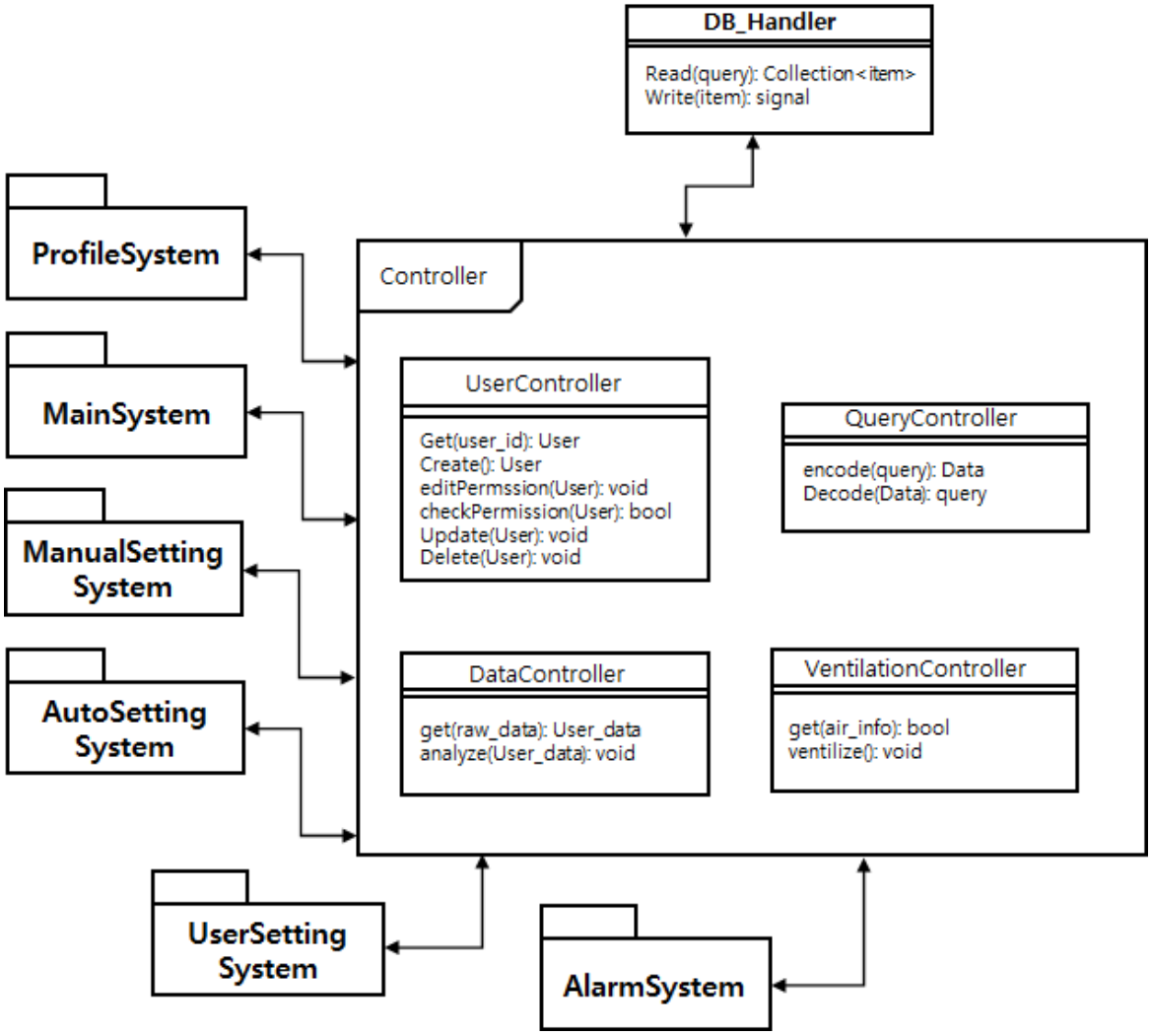


그림 30 Overall Architecture

위 그림은 백엔드 시스템의 전체적인 아키텍쳐를 나타낸다. 백엔드 시스템은 Firebase 를 통해 구현되며, System, Handler, 그리고 Controller 로 구성된다

5.2.1. System

각 System 은 세부 기능을 처리하는 역할을 하며 Profile System, Main System, Manual Setting System, Auto Setting System, User Setting System, 그리고 Alarm System 이 있다.

5.2.2. Handler

Handler 는 DB Handler 가 있으며, DB Handler 는 데이터베이스에서 데이터를 읽고 쓸 수 있도록 하는 인터페이스 역할을 한다.

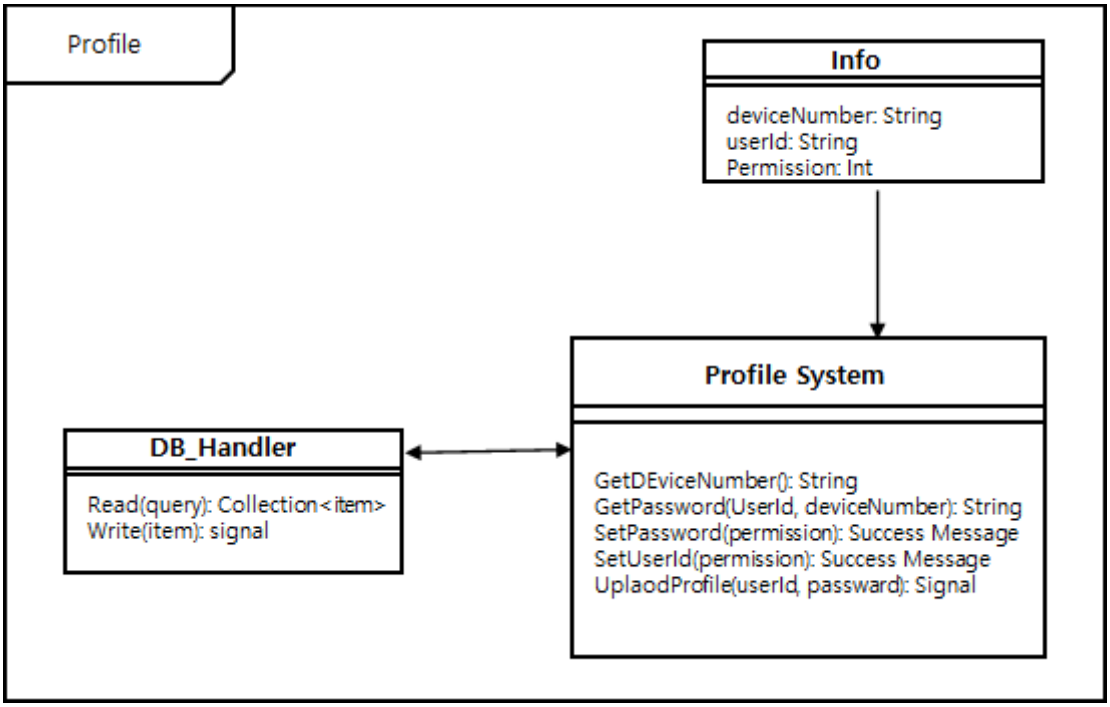
5.2.3. Controller

Controller 는 System 과 Handler 사이에 있으며 User Controller, Data Controller, Ventilation Controller, 그리고 Query Controller 가 있다.

5.3. Subcomponents

5.3.1. Profile System

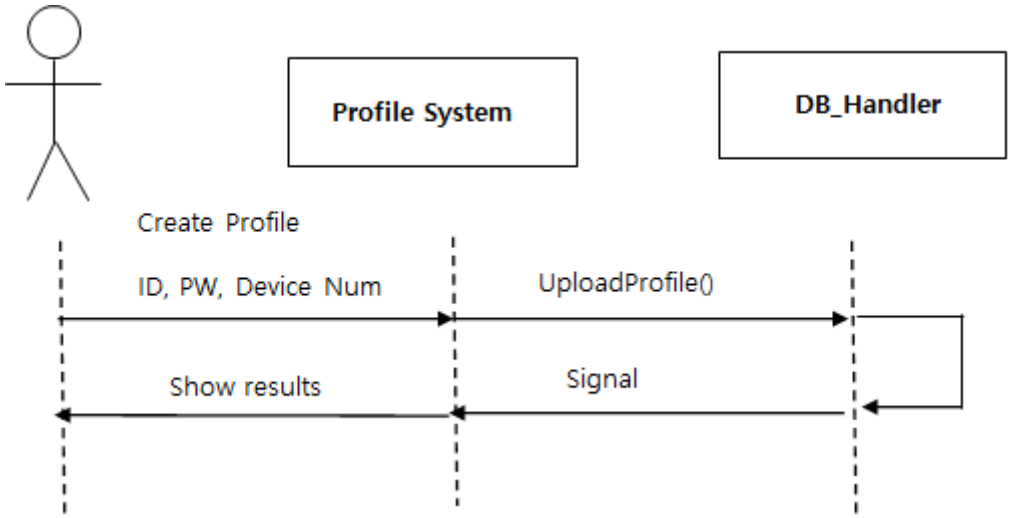
5.3.1.1. Class Diagram



Profile System Class 는 사용자의 프로필을 생성할 수 있도록 한다. 사용자는 아이디, 기기 번호, 승인 여부를 입력으로 넣어 계정을 생성할 수 있으며 UploadProfile() 함수와

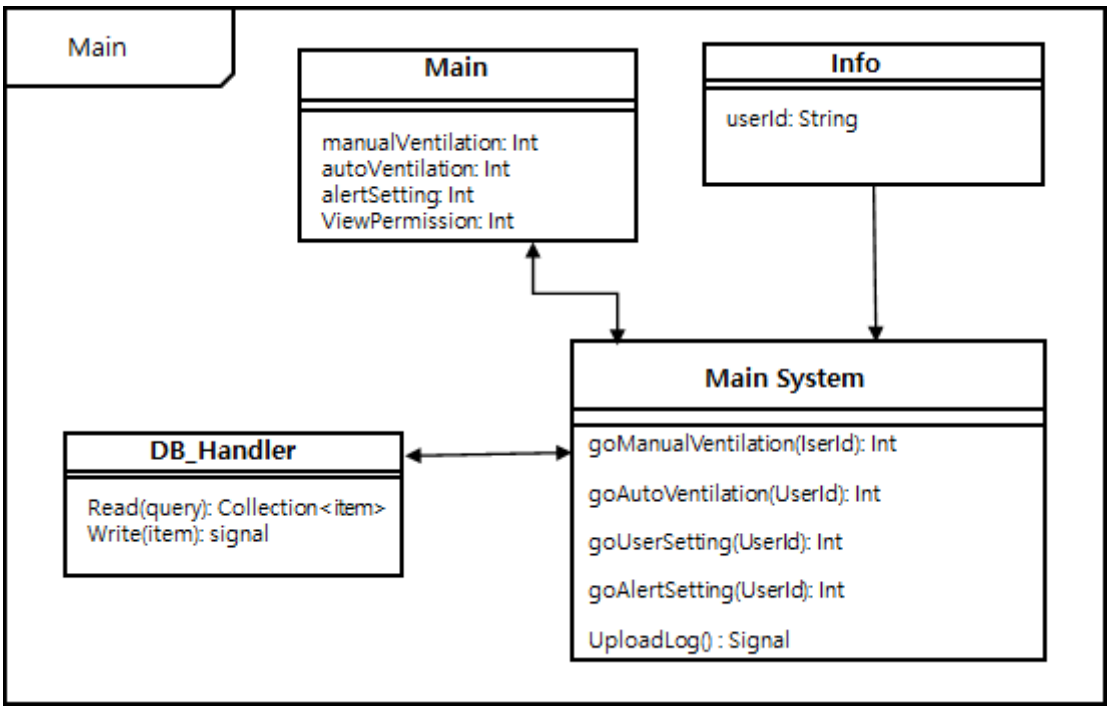
DB_Handler 를 통해 생성된 계정 정보를 저장할 수 있다.

5.3.1.2. Sequence Diagram



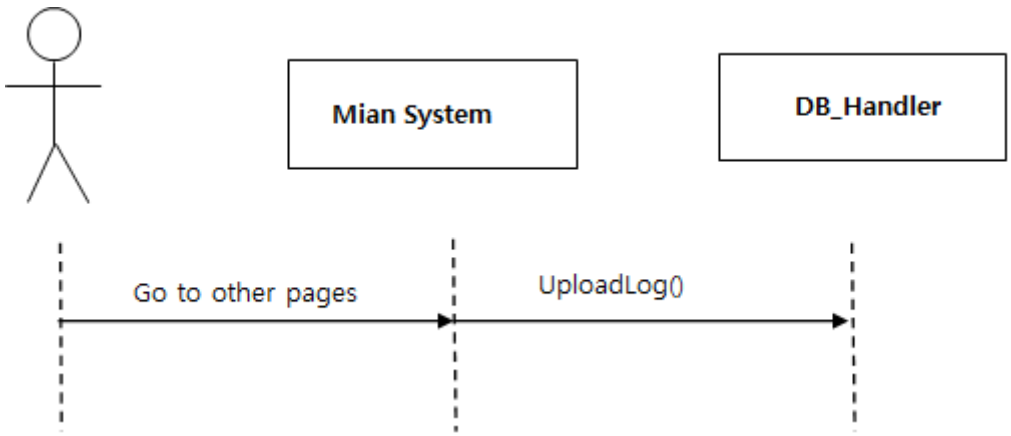
5.3.2. Main System

5.3.2.1. Class Diagram



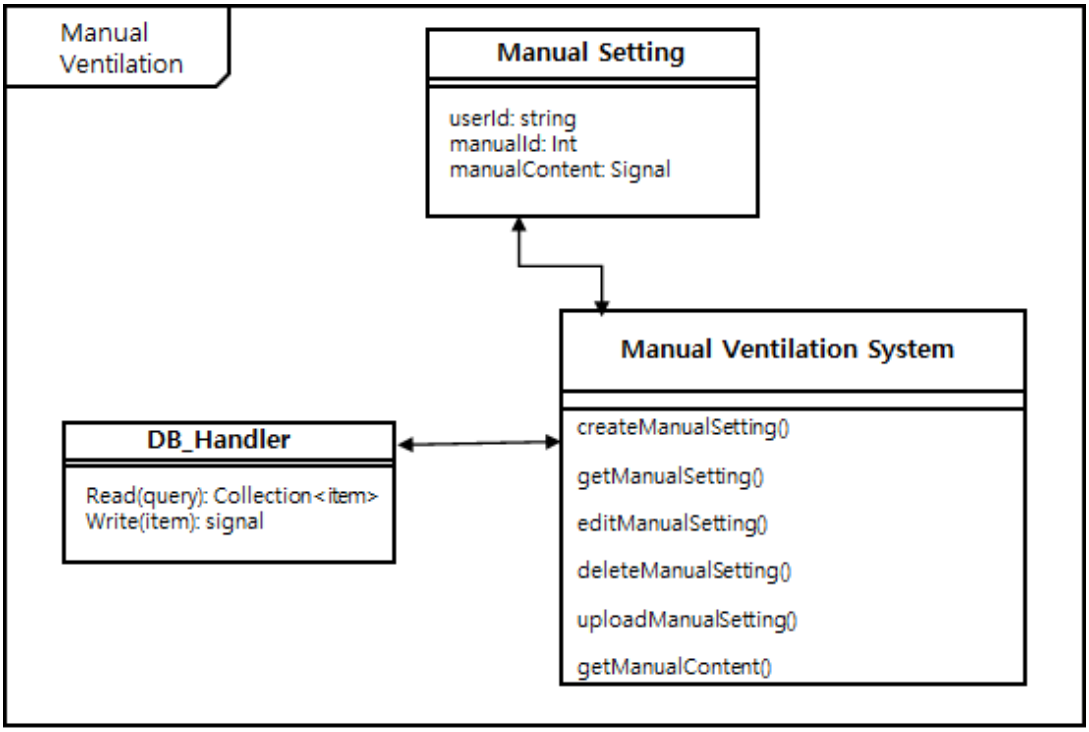
Main System Class 는 사용자가 메인 화면에서 다양한 설정 페이지로 이동할 수 있도록 하는 역할을 한다 사용자는 "go~()" 함수를 통해 원하는 페이지로 이동할 수 있다. UploadLog() 함수와 DB_Handler를 통해 사용자의 로그 데이터를 저장 가능하다.

5.3.2.2. Sequence Diagram



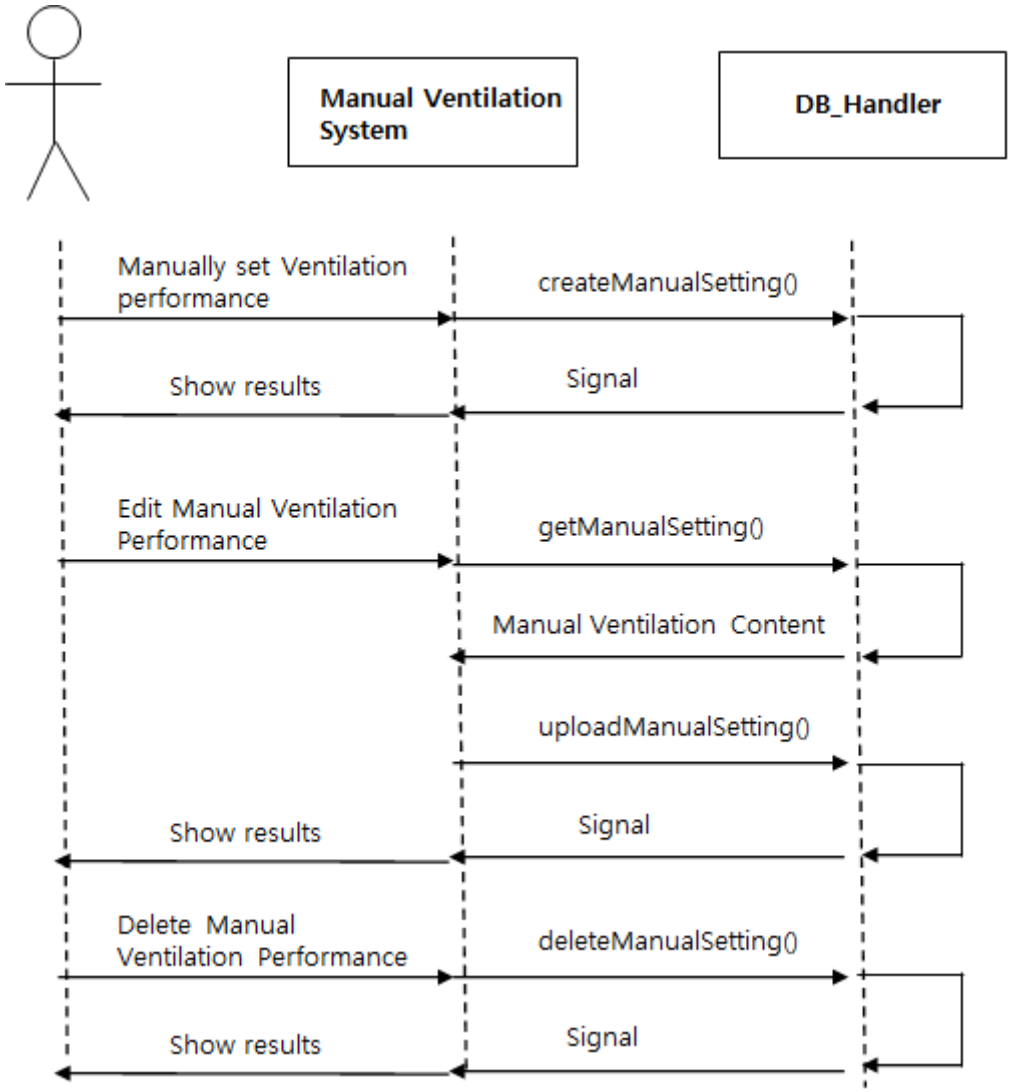
5.3.3. Manual Ventilation System

5.3.3.1. Class Diagram



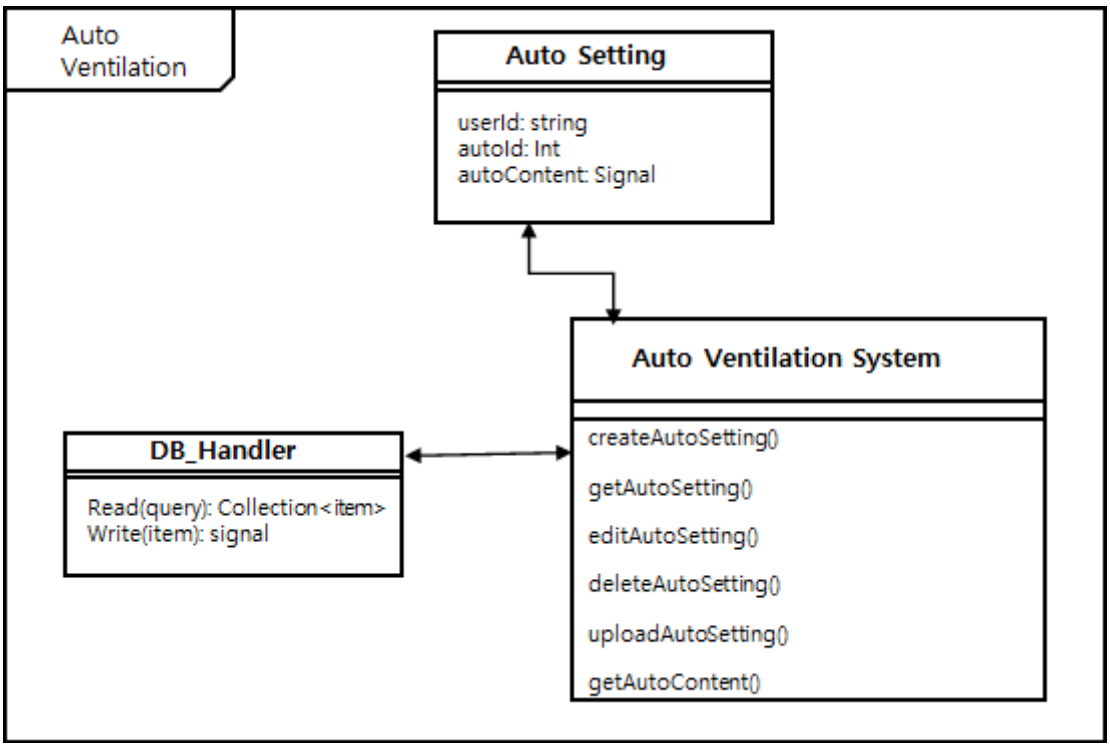
Manual Ventilation System Class 는 사용자가 환기 방식을 수동으로 설정 할 수 있도록 하는 역할을 한다. 사용자는 수동 환기 정보를 입력, 수정, 삭제 가능하며 uploadManualSetting() 함수와 DB_Handler를 통해 수동 환기 정보를 데이터베이스에 저장할 수 있다.

5.3.3.2. Sequence Diagram



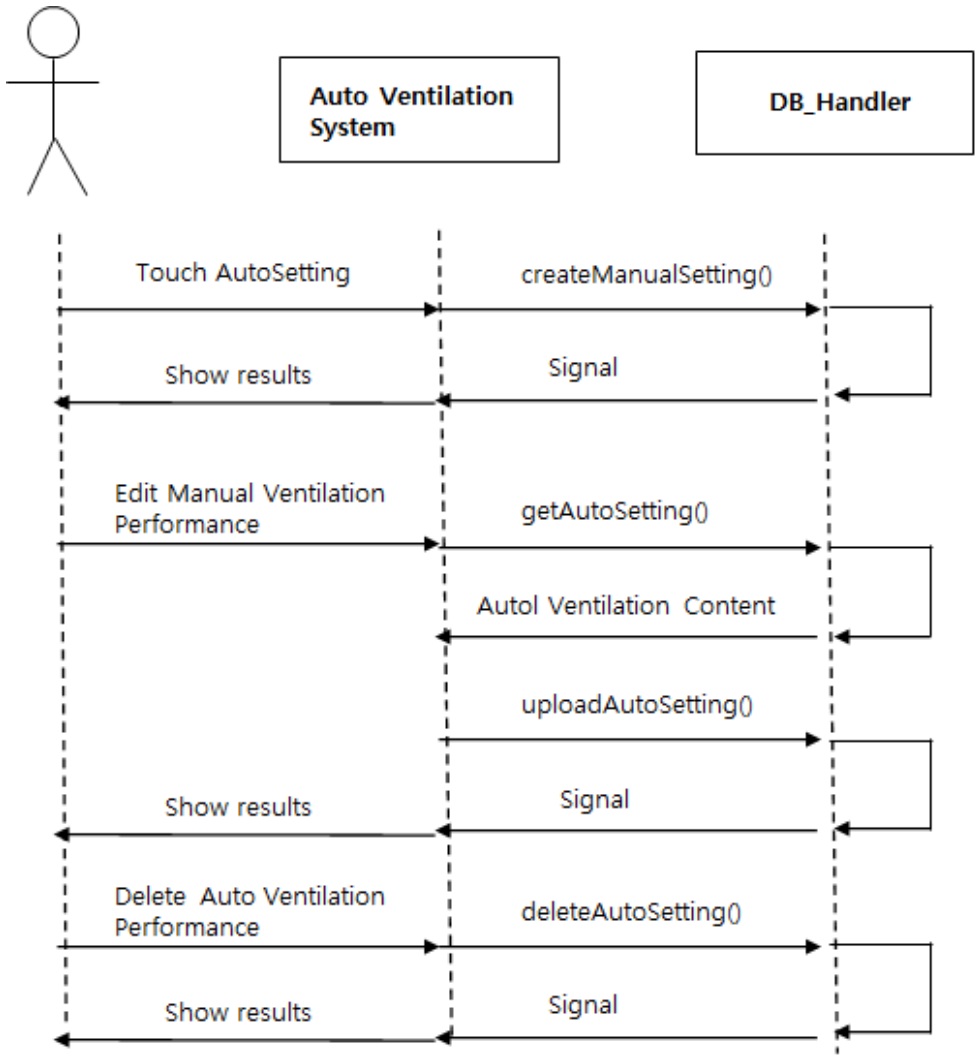
5.3.4. Auto Ventilation System

5.3.4.1. Class Diagram



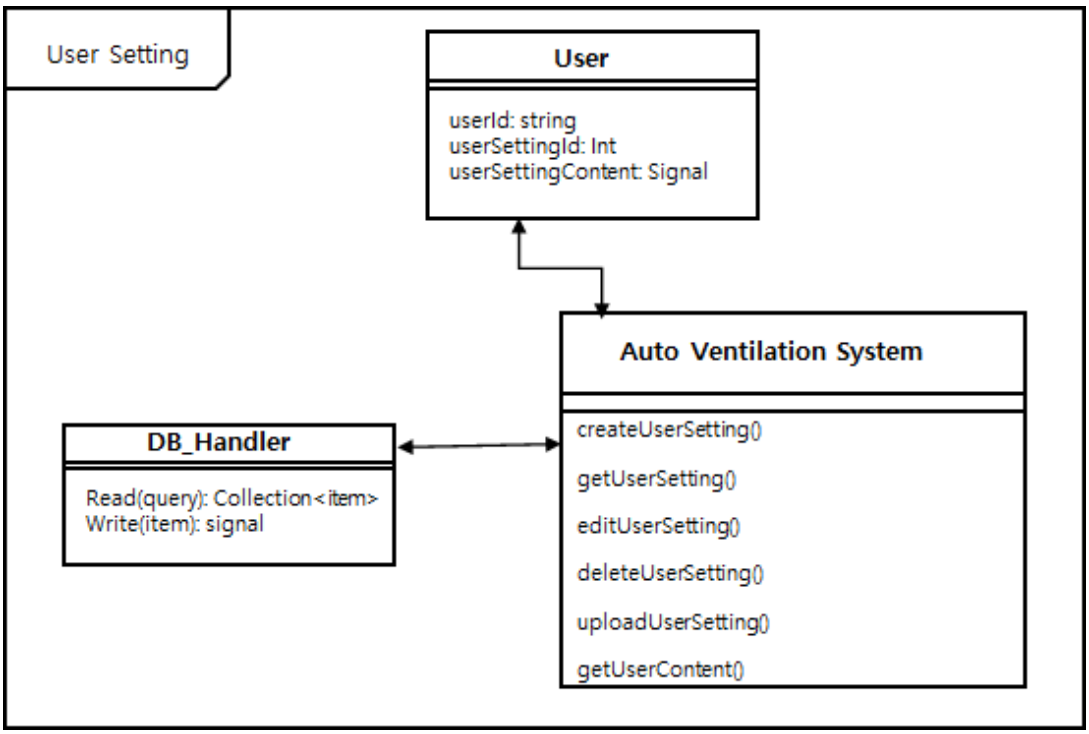
Auto Ventilation System Class 는 자동으로 제안된 환기 방식을 설정 할 수 있도록 하는 역할을 한다. 사용자는 자동 환기 정보를 입력, 수정, 삭제 가능하며 uploadAutoSetting() 함수와 DB_Handler를 통해 자동 환기 정보를 데이터베이스에 저장할 수 있다.

5.3.4.2. Sequence Diagram



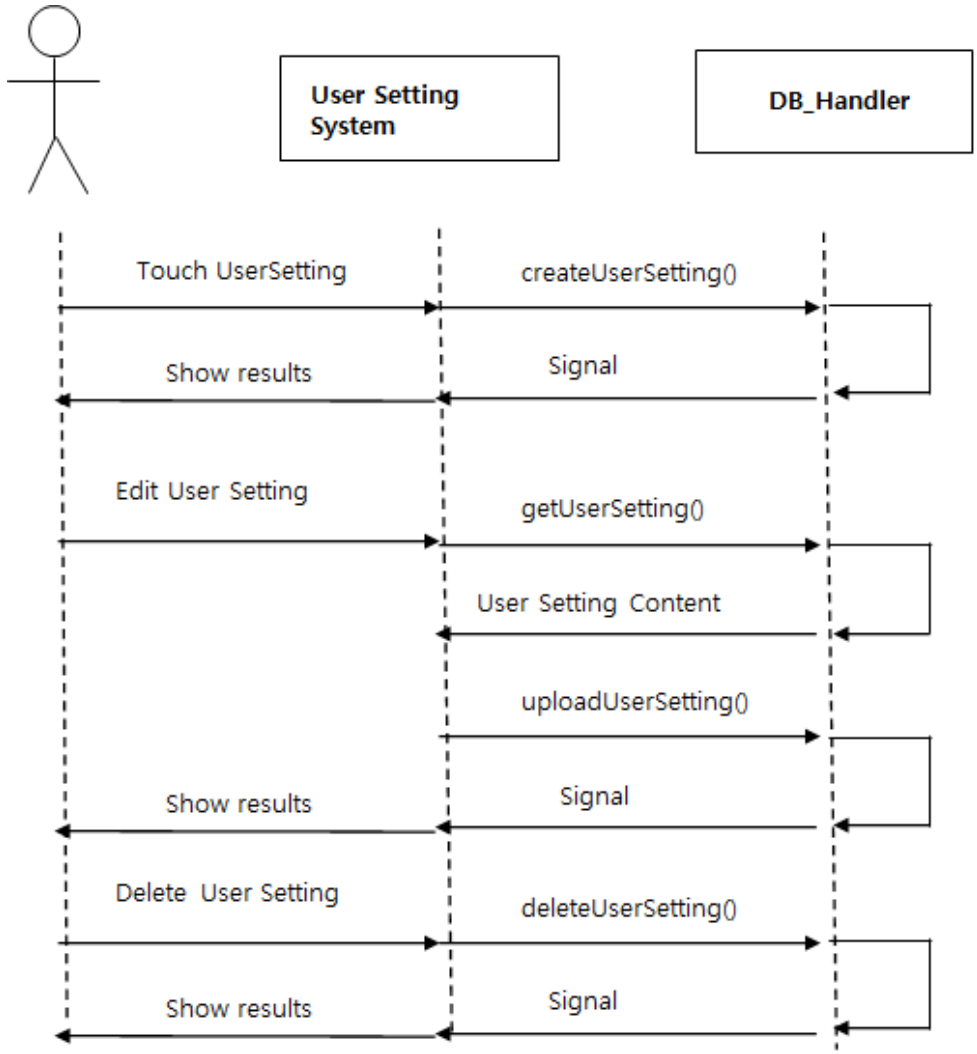
5.3.5. User Setting System

5.3.5.1. Class Diagram



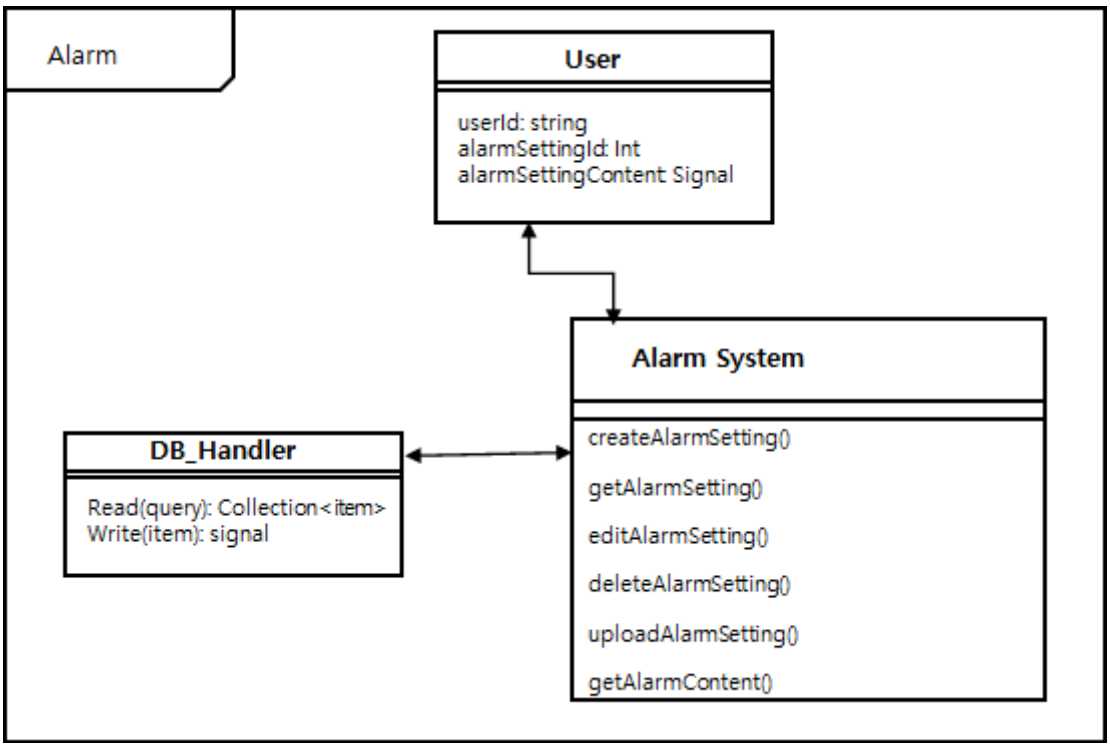
User Setting이란 현재 사용자가 설정한 유저 환경 설정 내용이다. 사용자는 해당 시스템을 이용하여 개인화된 제약 사항과 필더 등을 맞춤 설정 가능하다. 또한 생성된 사용자 설정 내용은 uploadUserSetting() 함수와 DB_Handler를 통해 데이터베이스에 저장된다.

5.3.5.2 Sequence Diagram



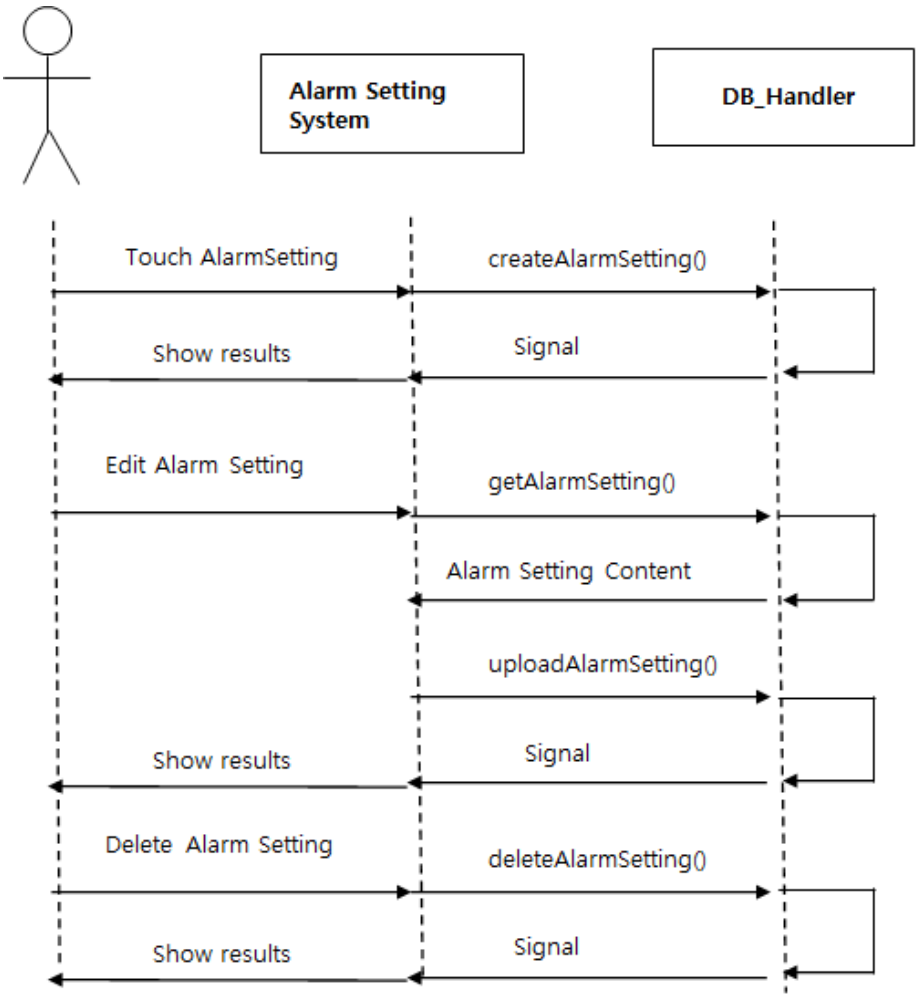
5.3.6. Alarm System

5.3.6.1. Class Diagram



Alarm System Class 는 사용자가 환기 알림을 원하는 대로 생성, 수정, 삭제할 수 있도록 한다. 사용자가 새로운 알림 설정 내용을 생성하면 uploadAlarmSetting() 함수와 DB_Handler 를 통해 생성된 알림 설정 데이터를 업로드 할 수 있으며 유저의 알림 설정 리스트에 불러올 수 있다. 유저가 알림을 수정하면 editAlarmSetting() 함수와 DB_Handler 를 통해 기존 알림 설정 데이터를 불러오고 수정 후 재 업로드할 수 있다. 유저가 알림을 삭제하면 deleteAlarmSetting() 함수와 DB_Handler 를 통해 알림 설정 데이터를 삭제한다.

5.3.6.2. Sequence Diagram



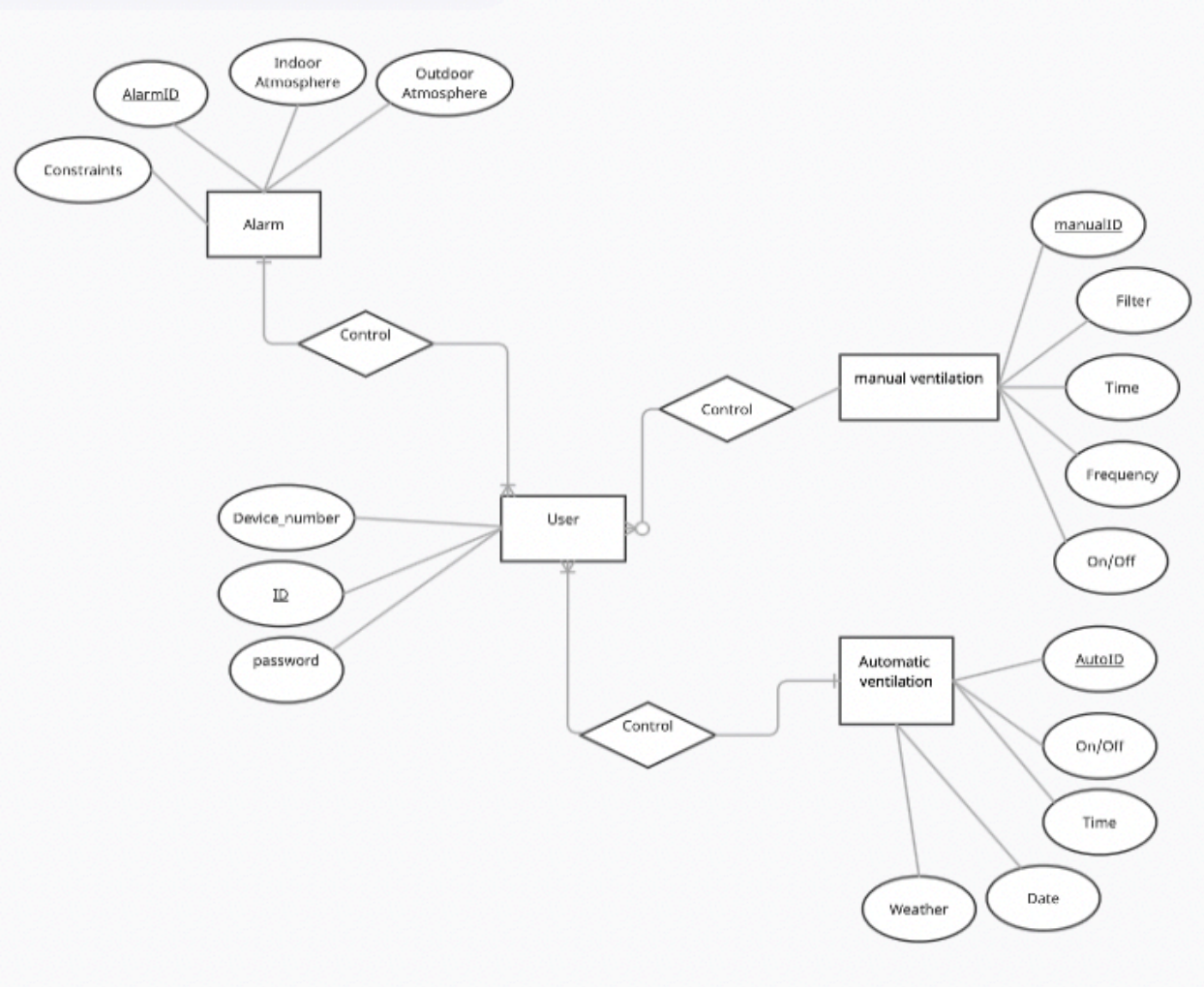
6. Database Design

6.1 Objectives

: 이 섹션에서는 데이터베이스의 구성과 그 상세를 보여준다. 이 섹션은 데이터베이스의 ER 다이어그램과 여러 관계 및 객체들을 보여준다. SQL 로 표현될 수 없는 데이터는 표현하지 않았다.

6.2 E-R diagram

데이터는 크게 4 개의 엔티티로 구성된다. User, Alarm, Manual ventilation, Automatic Ventilation 으로 총 구성된다. 각 개체와 관계는 다음 ER 다이어그램으로 표현된다.



7. Testing Plan

7.1. Objectives

이 챕터에서는 개발, 배포, 사용자 부문의 3 가지 분류에 따라 시스템의 테스트애 대한 계획을 기술한다. 테스트는 시스템이 갖는 잠재적인 에러 및 제품의 약점을 찾아내고, AI 자동환기 시스템을 소비자에게 배포하는 데에 무결점성과 안전성을 보장하기에 아주 중요하다.

7.2. Testing Policy

7.2.1. Development Testing

Development test 는 주로 소프트웨어 개발의 잠재적 위험을 줄이고 시간과 비용을 절약하기 위해 광범위한 오류 예방 및 탐지 전략의 동기화된 적용을 위해 수행된다. 이 단계에서 소프트웨어가 충분히 테스트되지 않았기 때문에 불안정할 수 있으며 구성 요소가 충돌할 수 있다. 따라서 이 단계에서 정적 코드 분석, 데이터 플로우 분석, 피어 코드 리뷰 및 단위 테스트를 수행해야 한다. 이러한 프로세스를 통해 1) 성능, 2) 안정성, 신뢰성 및 무장애 운영 보장, 3) 소프트웨어의 Identity 를 정의하는 보안 달성에 주로 중점을 둔다.

7.2.1.1. Performance

응용 프로그램 내 자동환기 작업은 시스템에서 가장 시간이 많이 걸리는 작업이며 개발자는 많은 동시 사용자들의 데이터 분석 시간을 줄이는 것이 중요하다. 권장 사양에 명시된 대로 시스템은 주어진 시간 내에 사용자에게 결과를 제공해야 한다. 우리는 다양한 선호도에 대한 테스트 사례를 준비하고, 환기 기능의 속도를 평가하고, 데이터 분석 알고리즘과 데이터베이스 서버와의 통신에 관한 코드의 흐름을 개선할 것이다.

7.2.1.2. Reliability

하위 시스템 수준에서 종종 수행되는 신뢰성 시험에는 기기 장치의 고장 빈도, 연속 작동, 수리 불가능한 장치의 평균 고장 시간, 수리 가능한 시스템의 평균 고장 간격 및 모든 장치의 신뢰성 성능 함수로서의 임무 성공 확률을 추정하는 시험이 포함된다.

시스템 개발 단계부터 지속적으로 개발 테스트를 진행하여 각 유닛이 시스템에 통합되는 동안 반복적으로 고장 여부를 점검할 것이다.

7.2.1.3. Security

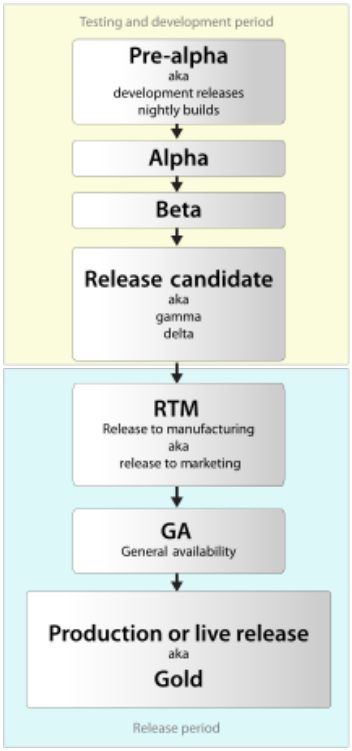
오늘날 거의 모든 기업이 디지털 방식으로 기술 기업으로 변모함에 따라 리스크에 대한 누적 노출이 기하급수적으로 증가하고 있다. 그래서 우리는 신뢰성을 매우 중요시해야 하며 이것을 견고한 보안의 토대 위에 마련해야 한다.
보안 테스트는 잠재적으로 악의적인 공격을 허용할 수 있는 취약성을 식별하기 위해 배포되는 소프트웨어 테스트의 한 유형으로 설명할 수 있다. 이 활동을 수행함으로써, 우리는 시스템의 모든 허점을 밝혀내어 정보, 수익, 그리고 브랜드 가치에 대한 부정적인 영향을 방지할 수 있다. 여기서의 주요 목표는 소프트웨어를 엔터프라이즈 인프라에 통합하기 전에 가능한 모든 위험을 감지하는 것이다. 이 접근 방식은 또한 개발자들에게 중요한 보안 사고가 되기 전에 이러한 문제를 해결할 충분한 시간을 제공한다. 또한 어플리케이션 시스템의 보안을 위해, 우리는 거의 완성된 버전의 앱에 접속하여 보안 문제를 파악하고 수동 코드 검토를 통해 보고서를 작성할 것이다. 이밖에 Ostlorlab, Appvigil 등이 제공하는 다른 모바일 앱 보안 테스트 서비스도 이용함으로써 개발자들이 간과한 앱 취약점을 검토할 것이다.

7.2.2. Release Testing

Release Testing 는 소프트웨어 Release 후보가 사용자에게 준비되었다는 확신을 팀에 제공하는 코딩 관행 및 테스트 전략을 뜻한다. Release Testing 은 소프트웨어 Release 에서 오류와 버그를 찾아 제거하여 사용자에게 배포할 수 있도록 하는 것을 목표로 한다. 이러한 Release Testing 은 반드시 소프트웨어의 정식 버전 배포 전에 실행되어야 한다. 다음은 Release Testing 를 수행하는 데 사용되는 몇 가지 방법이다.

소프트웨어 배포 생명 주기(Software Release Life Cycle)에 따르면 Release Testing 는 소프트웨어 기본적인 구현이 끝난 시점(Release)부터 시작한다(Alpha test). 이 Release 는 선택한 모든 새 소프트웨어 기능과 버그 수정을 포함하는 코드 기준의 스냅샷을 나타낸다. 이 소프트웨어 후보는 패키지화되어 "생산 준비 완료"라는 레이블이 지정된다. 사용자가 사용할 수 있도록 테스트하고자 하는 것이 바로 이 소프트웨어 Release

후보이다. 이 Release 버전은 Beta 테스트 단계에서 실제 유저들과 함께 테스트를 진행하게 된다.



Software Release Life Cycle

즉, Release Testing 목적은 Release 후보에게 확신을 심어주는 것이다. Release Testing 는 단일대규모 테스트 방법이 아닌 테스트 접근 방식 또는 전략이다.

다른 테스트 접근법과 마찬가지로 Release Testing 는 통제된 환경에서 소프트웨어 Release 후보를 실행하는 시스템을 파괴하는 것을 목표로 한다. 개발 팀은 이 한 릴리스 후보자에게 품질 보증 노력을 집중하여 실제 사용자에게 전달되기 전에 수정할 수 있도록 이 릴리스 후보를 해결하려한다.

7.2.3. User Testing

User Testing 은 실제 사용자가 시스템을 사용하는 방법에 대한 직접적인 입력을 제공하는 필수불가결한 사용 적합성 관행이다. 이러한 테스트는 최종 제품의 출시에 가장 중요한데, 이는 사용자 사이에 혼동을 일으키는 기능은 오래가지 못하기 때문이다.

제품의 편의성 및 직관성에 더욱 신경을 쓰고 사전에 노출되지 않은 사용자를 대상으로 테스트해 볼 것이다. AI 자동환기를 통한 이용자의 편의성 및 User 적합성에 대한 양적 평가를 시행함으로 써 얻은 피드백을 수렴 및 개선할 것이다.

7.2.4. Testing Case

Testing Case 에는 사용자에게 정확한 정보를 그래픽을 통해 가시적으로 나타내는 지, 실제 구동과정에서 예기치 못한 오류가 생기는 지, 실제 사용자의 패턴 및 왜/내부 날씨 데이터를 잘 분석했는지"의 각 측면에 대해 3 개 이상의 테스트를 실시하고, 테스트에 참여한 사용자로부터 평가를 받을 예정이다. 이를 기반으로 시스템 평가서를 작성할 것이다.

8. Development Plan

8.1. Objectives

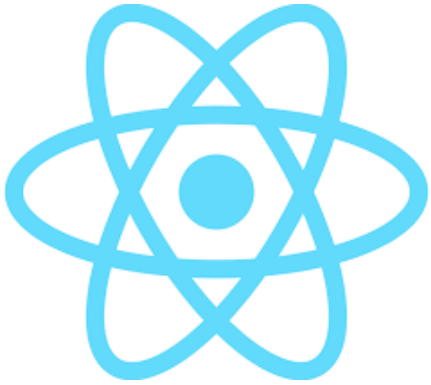
이 챕터는 어플리케이션 환경에 대한 기술과 프로그램이 기술되어 있다.

8.2. Frontend Environment

8.2.1. Adobe Xd

Adobe Xd 는 웹 및 모바일 앱 설계를 위한 일체형 UX/UI 솔루션 프로그램이다. 다양한 플랫폼에서 대화형 프로토타입을 제공하여 팀원 간의 커뮤니케이션을 촉진하고 피드백을 신속하게 반영할 수 있다. 프로토타입을 시각화하여 팀원들과 실시간으로 공유할 수 있다.

8.2.2. React Native



유저 인터페이스를 만드는 데 사용되는 오픈 소스 자바스크립트 라이브러리인 React 의 문법으로 안드로이드, Ios 앱을 개발할 수 있는 프레임워크이다. React 를 배웠던 개발자라면 몇 시간만에 익숙해질 수 있을 만큼 React 와 거의 유사한 문법을 가지고 있다. 현재는 웹과 동일한 CSS 요소 사용을 지원하므로 웹개발자나 퍼블리셔도 무리 없이 UI 구성이 가능하다.

리렌더링이 잦은 동적인 모던 웹에서 엄청나게 빠른 퍼포먼스를 내는게 가능하다. 기본적으로 모듈형 개발이기 때문에 생산성 또한 상당히 높은 라이브러리인지라 순식간에 대세로 떠올랐다. 거기에 기본적으로 라이브러리인지라 다른 프레임워크에 간편하게 붙여서 사용하는 것도 가능하며, React Hooks 라는 강력한 메소드들을 지원하면서 사실상 웹 프론트엔드 개발의 표준으로 자리잡았다.

8.2.3. Figma

웹 기반 UI/UX 디자인 및 프로토타이핑 툴로 설치가 필요하지 않으며, 브라우저에서 바로 실행할 수 있다. 즉, 인터넷이 되는 환경이라면 어떤 운영체제를 사용하든지, 어떤 기기를 사용하든지 사용이 가능하다. 좀 더 나은 퍼포먼스를 위해 네이티브 앱도 지원하고 있어 연동해서 사용이 가능하다.
또한 협업에 특화되어 있다. 다른 프로토타이핑 툴은 그 툴에서 작업한 내용을 공유 프로그램을 거쳐서 전달하는 방식을 사용한다면, 피그마는 자체가 웹 기반이기 때문에 피그마 계정 소유자가 링크를 공유하여 여러 명이 아트 보드를 확인하며 동시에 온라인으로 실시간 작업을 진행할 수 있다. 디자이너와 개발자의 소통을 도와주는 기능이 많다. 개발자가 참조할 수 있는 정보를 주는 '개발 툴바'가 있으며, 마우스 툴바로도 수치값을 확인할 수 있어 가이드라인이 없어도 빠른 작업이 가능하다.

8.3. Backend Environment

8.3.1. Github

대표적인 무료 Git 저장소로 Git 을 이용한 소프트웨어 개발 버전 관리를 위한 호스팅을 제공한다. 외부 라이브러리의 취약점이 확인된 경우 사용자에게 해당 사실을 통보하고 자동으로 최신 버전으로 교체하는 기능을 제공한다. 팀원들은 Github 를 통해 단일 프로젝트를 함께 개발할 수 있어 구성 요소를 쉽게 통합할 수 있다.

8.3.2. Firebase

Firestore ,클라우드 스토리지, 실시간 데이터베이스, 머신러닝 키트 등 다양한 기능을 제공해 모바일과 웹 애플리케이션 개발을 지원한다. 그 중, 우리는 날씨 data 와 User pattern 분석 data 등의 데이터를 관리하기 위해 실시간 데이터베이스 기능을 사용할 것이다. 실시간 데이터베이스 덕분에 이 데이터베이스에 연결된 모든 클라이언트에서 데이터가 동기화된다. 즉, 모든 클라이언트가 단일 실시간 데이터베이스 인스턴스를 공유하고 자동 업데이트를 통해 업데이트된 데이터를 수신할 수 있다.

8.3.3. Zerynith

제린스(Zerynth)는 마이크로컨트롤러를 프로그래밍하기 위한 파이썬 프로그래밍 언어의 소프트웨어 구현체이다. 32-bit 마이크로컨트롤러 플랫폼을 대상으로 하며 파이썬과 C 코드를 혼합하도록 설계되었다. Zerynth 는 사물인터넷(IoT) 제품을 개발하기 위해 마이크로컨트롤러를 클라우드에 연결한다. Zerynth 는 낮은 수준의 구성과 프로그래밍에 집중하지 않고 제품 설계와 아이디어를 위한 디자이너, 엔지니어, 프로그래머, IoT 전문가들을 위해 만들어진 상호교류복합 Tool 이다..

8.3.4. TensorFlow

구글의 오픈소스 기계학습 라이브러리로 딥러닝과 기계학습 분야를 일반인들도 사용하기 쉽도록 다양한 기능들을 제공한다. 하이 레벨 프로그래밍 언어로 알려진 Python, C++, GO, Java, R 을 활용하여 연산처리를 작성할 수 있다. 데이터 플로우 그래프를 통한 풍부한 표현력을 가졌으며 아이디어 테스트에서 서비스 단계까지 이용이 가능하다. 계산 구조와 목표 함수만 정의하면 자동으로 미분 계산을 처리하는 유용한 기능을 제공한다.

8.4. Constraints

이 시스템은 본 문서에 언급된 내용을 기반으로 설계 및 구현될 것이다. 기타 세부사항은 개발자가 선호하는 방향을 선택하여 설계하고 구현하지만, 다음과 같은 사항에 따라 세부적인 구현 방향성이 결정될 것이다.

- 이미 널리 입증된 기술을 사용할 것
- 매핑 속도는 5 초를 초과하지 않아야 할 것
- 별도의 라이센스가 필요하거나 로열티를 지불하는 기술 또는 소프트웨어를 사용하지 말 것 (이것이 시스템에 필요한 유일한 기술 또는 소프트웨어인 경우 이 조항은 제외한다.)
- 전반적인 시스템 성능 개선을 모색하는 방향으로 결정할 것.
- 보다 사용자 친화적이고 편리한 방향으로 결정할 것
- 가능한 경우 항상 오픈 소스 소프트웨어 사용할 것
- 시스템 리소스 낭비를 방지하기 위해 소스 코드 최적화할 것
- 향후 유지보수를 고려하고 소스 코드를 작성할 때 충분한 주석을 추가할 것
- 최소 Android 버전 6.0(API 23) 및 타겟 Android 버전 29, iOS 13.0 이상의 버전으로 개발할 것
- Android 버전 10(API 29), iOS 13.0 이상의 버전을 사용하여 시스템 에뮬레이트할 것
- 앱이 실행되면 3 초 안에 메인 화면에 진입해야 할 것
- 로그인 프로세스는 2 초 내에 완료되어야 할 것
- 1GB RAM, 1.6GHz CPU, 32GB 저장공간, Android 6.0 이상 또는 iOS 13.0 이상
- 한 화면에서 다른 화면으로 진입할 때는 1 초 내에 완료되어야 할 것
- 환경 설정 화면 전환은 2 초 내에 완료되어야 할 것

- 앱 내 환시 요구사항은 입력 후 2 초 안에 적용되어야 할 곳

System Requirement	
Section	**Minimum Specification**
Processor	Intel® i5-4590 ~ AMD FX 8350 ~
RAM	1GB
Graphic card	AMD 임베디드 Radeon™ E9565 MXM 모듈
Storage	4GB

8.5. Assumptions and Dependencies

이 시스템은 React native cross platform 을 이용하여 설계 및 구현된다. 그러므로 모든 함수와 내용은 모든 플랫폼에서 동작할 수 있다. 그러나 최소 Android 6.0 (API 23) 이상의 OS, iOS 13.0 이상의 버전이 요구된다. 따라서 다른 버전에는 적용되지 않을 수 있다.