Smart Homepany: Customize your workplace

Software Design Specification

2022. 05. 15.

Introduction to Software Engineering (SWE3002-42)

TEAM 1

Team Leader	김종헌
Team Member	김영빈
Team Member	김민현
Team Member	오세훈
Team Member	이미소
Team Member	임세창
Team Member	홍권

목차

표 목차

그림 목차

1. Preface

이 장에서는 재택근무자를 위한 스마트 홈 시스템 Smart Homepany 의 설계를 위한 본 명세서의 독자, 범위, 목적, 그리고 구조에 대해 기술한다.

1.1 Readership

본 문서는 Smart Homepany 의 디자인 요소에 따라 총 10 장으로 구성되어 있고, 아래의 1.4 Document Structure 에서 확인할 수 있다. 본 문서의 주 독자는 Smart Homepany 의 실사용자, 성균관대학교 2022 년 1 학기 소프트웨어공학개론 팀 1 이나, 해당 수업의 다른 수강생, 교수, 조교이다.

1.2 Scope

본 문서는 Smart Homepany 의 구현을 위해 사용될 디자인 요소에 대한 정보를 포함하며, Smart Homepany 의 소프트웨어 공학과 소프트웨어 품질 공학에 사용될 예정이다.

1.3 Objective

본 문서는 Smart Homepany 의 디자인 요소 제공을 주 목적으로 하며, 시스템 개발에 필요한 아래의 정보를 포함하고 있다.

- System Architecture
- System Design Decision

1.4 Document Structure

1. Preface: 본 문서의 독자, 범위, 목적과 구조를 기술한다.
2. Introduction: 본 문서 및 Smart Homepany 에 대해 간략하게 설명한다. 본 문서의 작성에 사용된 Diagram 및 Tool 에 대한 정보 또한 포함되어 있다.
3. System Architecture - Overall: 시스템의 전체적인 구조를 기술한다.
4. System Architecture - Frontend: 시스템의 Frontend 부분 구조를 기술한다.
5. System Architecture - Backend: 시스템의 Backend 부분 구조를 기술한다.
6. Protocol Design: 시스템의 통신에 사용하는 프로토콜을 기술한다.
7. Database Design: 시스템의 데이터베이스 구성을 기술한다.

8. Testing Plan: 시스템 테스트 계획을 기술한다.

9. Development Plan: 시스템 개발에 사용된 Tool 과 시스템에 대한 제약조건, 가정 및 의존성을 기술한다.

10. Supporting Information: 본 문서의 수정 기록을 기술한다.

2. Introduction

Smart Homepany 는 Team1 에서 기획한 스마트 홈 소프트웨어로, 최근 수 년간 급증한 재택 근무자들이 집에서도 업무 시간에 업무에 완전히 집중할 수 있고, 업무 시간 외에는 업무에서 벗어나 편안하게 휴식을 취할 수 있는 주거 환경을 조성하는 것에 그 목적이 있다. Smart Homepany 는 소프트웨어를 사용하는 재택 근무자로 하여금 집에 있는 여러 IoT 기기들을 연동하고 제어할 수 있는 기능을 제공한다. 다양한 IoT 기기들의 조작을 하나의 프리셋에 담아서 집 전체에 반영되게 하고, 각 기능들을 원하는 대로 커스터마이징하여 자신에게 맞는 업무/주거 환경을 구성할 수 있게 한다. 또한, 사용자가 일정을 등록하면 그에 맞춰서 예약해 둔 프리셋을 자동으로 전환하는 기능을 제공하여 편의성을 추구하였다. 본 디자인 명세서에는 Smart Homepany 프로젝트에서 사용되었던 여러 디자인 구조가 묘사되어 있으며, 이 구조들은 이전에 작성된 요구사항 명세서에서 명시해 둔 요구사항들에 따라서 설계되어 있다.

2.1 Objectives

이 장에서는 본 프로젝트의 설계 단계에서 사용한 여러가지 툴 및 다이어그램에 대한 설명과 정의를 제공하고 있다.

2.2 Applied Diagrams

이 섹션에서는 UML 용어와 프로젝트에서 사용한 UML 다이어그램의 유형을 정의한다.

2.2.1 UML

UML 은 Unified Modeling Language(통합 모델링 언어)의 줄임말로, 소프트웨어 공학에서 사용하는 표준화된 범용 모델링 언어이다. UML 은 시스템의 설계를 시각적으로 표현하기 위해 사용된다. UML 을 통해서 프로그램 개발에 대해 전문가와 비전문가가 서로 대화할 수 있으며, 개발이나 시스템 설계에 있어서 개발 인원 전체가 같은 이해를 갖고 개발에 참여할 수 있다.

2.2.2 Use Case Diagram

Use Case Diagram 은 사용자와 관련된 다양한 use case 들과 시스템 간의 상호 작용을 나타내는 diagram 이다. Use Case Diagram 은 시스템의 설계를 위한 요구 사항의 주된 형태이다. 이 문서는 시스템이 무엇을 해야 하는지 단순화된 형태의 지침이 되고, Use Case, Actor, System 간의 관계 중 일부만을 요약한다. Use Case Diagram 에서 Actor 는 소프트웨어를 실제로 사용하는 소비자에

해당한다. Scope 는 어플리케이션의 주요 기능으로 여겨지는 갈래들이며, 모바일 디바이스, IoT 기기, 프리셋, 타임라인을 관리 및 설정하는 것에 해당된다. Use Case 는 Scope 에 속한 것으로, 사용자가 각 기능들을 사용할 때 마주하게 되는 상황들이다. IoT 기기, 프리셋, 타임라인을 조회, 등록, 삭제, 제어하는 Use case 들이 있다.

2.2.3 Class Diagram

Class Diagram 은 시스템을 구성하는 객체, 객체의 속성, 메소드, 객체 간의 관계를 보여줌으로써 시스템의 구조를 설명하는 정적 구조 다이어그램의 일종이다. Class diagram 은 실제 소프트웨어의 설계 혹은 구현을 위한 용도로 사용되는데, 앞으로 구현될 실제 클래스를 표현하고 있기 때문에 소스 코드와 관계가 깊다. 또한 diagram 의 의미가 모호하지 않게 하기 위해 주의하면서 제약과 규칙, 형식을 지켜야 한다.

2.2.4 Sequence Diagram

Sequence Diagram 은 객체 간의 상호 작용을 나타내는 행위 다이어그램이다. 시간에 따른 순차적인 진행을 보이기 때문에 현재 시스템이 어떤 시나리오로 움직이고 있는지를 나타내기에 좋다. Sequence diagram 을 이용하면 use case 가 실제로 사용자와 기기에서 어떤 상호작용을 통해 작동하는지 표준화된 작업 과정을 보여줄 수 있다.

2.2.5 Context Diagram

Context Diagram 은 시스템 경계를 정의하고, 시스템과 상호작용하는 외부 구성 요소 간의 데이터 흐름을 표현하는 데 사용된다. 이 diagram 은 데이터 흐름도 중 가장 기본적인 수준이며, 전체 시스템을 대표하기 때문에 시스템과 사용자 간 데이터의 큰 흐름만을 표현한다.

2.2.6 Entity Relationship Diagram

Entity Relation Diagram(이하 ER Diagram)은 한국어로 표현하자면 개체관계도로, 데이터와 데이터들의 관계를 표현한 도식화된 그림이다. 시스템 내부에 있는 다양한 entity 가 서로 어떻게 관련되어 있는지를 시각적으로 표현한다. ER Diagram 은 현실 세계의 요구사항들을 반영한 데이터베이스를 설계하는 과정에서 활용된다.

2.3 Applied Tools

2.3.1 Microsoft Word

이 툴은 프로젝트의 요구사항 명세서와 디자인 명세서를 작성하는 데에 사용되었다. 서로 다른 문서 편집 환경에서 모든 팀원이 제약 없이 같은 작업 환경을 사용할 수 있도록 지원하는 툴이다.

2.3.2 Microsoft PowerPoint

이 툴은 본 프로젝트의 계획서를 작성하는 데에 사용되었으며, 추후 최종 발표를 위한 자료를 작성하는 데에도 이용될 것이다.

2.3.3 draw.io

이 툴은 설치가 필요 없는 웹 기반 diagram 제작 소프트웨어로, UML 을 통한 diagram 작성을 지원하여 수많은 flowchart 들과 diagram 들을 쉽게 만들 수 있다. 설치 없이 네트워크가 연결된 환경이라면 웹 브라우저에서도 작성할 수 있고, 데스크탑용 어플리케이션을 설치하면 더욱 편하게 작성할 수 있다.

2.3.4 carbon.now.sh

이 툴은 블로그 등의 웹페이지 및 보고서, 제안서 등에 소스 코드를 공유할 때, 이를 시각적으로 보다 정돈된 형태로 공유할 수 있도록 지원한다. 소스코드를 입력하고 테마와 언어를 선택하면 이미지 파일로 추출하여 코드를 깔끔한 형태로 저장할 수 있다. SQL DDL 을 작성할 때 더 가독성 있게 저장하기 위해 사용되었다.

2.3.5 figma.com

이 툴은 어플리케이션의 목업 이미지를 제작하기 위해 사용되었다. 설치가 필요 없는 웹 기반 소프트웨어이기 때문에 네트워크가 연결된 환경이라면 브라우저만으로 제작이 가능하고, 사용자의 컴퓨터에 네이티브 프로그램을 설치하면 더 빠르게 작업을 진행할 수 있다. 또한 모바일에 최적화되어 있어 플레이 시 세부적인 부분을 실제 화면처럼 만들 수 있다.

2.4 Project Scope

스마트 홈 어플리케이션 Smart Homepany 는 업무를 자택에서 수행해야만 하는 재택 근무자들의 편의성 증진을 위해 고안된 프로젝트이다. 기존의 스마트 홈 소프트웨어들이 IoT 기기들을 이용해서 주거 환경만을 개선하는 데에 그쳤기 때문에 이를 보완하여 자택을 업무에 적합한 환경으로 만들 수 있게 하기 위한 목적을 갖고 있다. 그러므로 자택에 있는 IoT 기기들을 제어하기 위한 통합 환경 시스템, 프리셋과 타임라인을 관리하기 위한 서버가 요구된다. 또한, 개인마다 다른 업무 환경의 특성을 고려하여 커스터마이징 기능을 지원하여 획일화되지 않고 개개인에 맞는 서비스를 제공할 수 있도록 노력할 것이다.

2.5 References

"Project Highlight_Design 2.0.4" SKKU 2021 Introduction to Software Engineering Team1, Last Modified on May 16, 2021
https://github.com/skkuse/2021spring_41class_team1/blob/main/doc/Project%20Highlight_Design%202.0.4.pdf
"SE-2021-2-Team4-SDS" SKKU 2021 Introduction to Software Engineering Team 4, Last Modified on 21 Nov, 2021.
https://github.com/skkuse/2021fall_41class_team4/blob/main/docs/SE-2021-2-Team4-SDS.pdf

3. System Architecture - Overall

3.1 Objectives

이 장에는 본 시스템의 프론트엔드부터 백엔드까지의 개괄적인 디자인 구성이 묘사되어 있다.

3.2 System Organization

3.2.1 Context Diagram

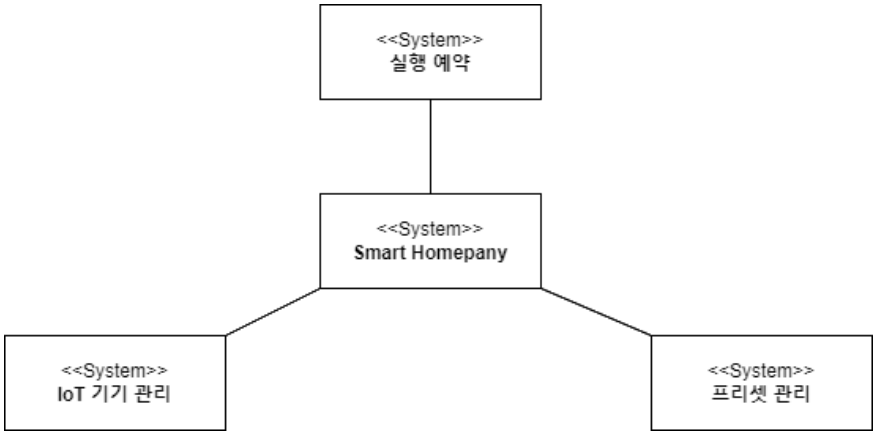

그림 1 System Organization - Context Diagram

3.2.2 Sequence Diagram

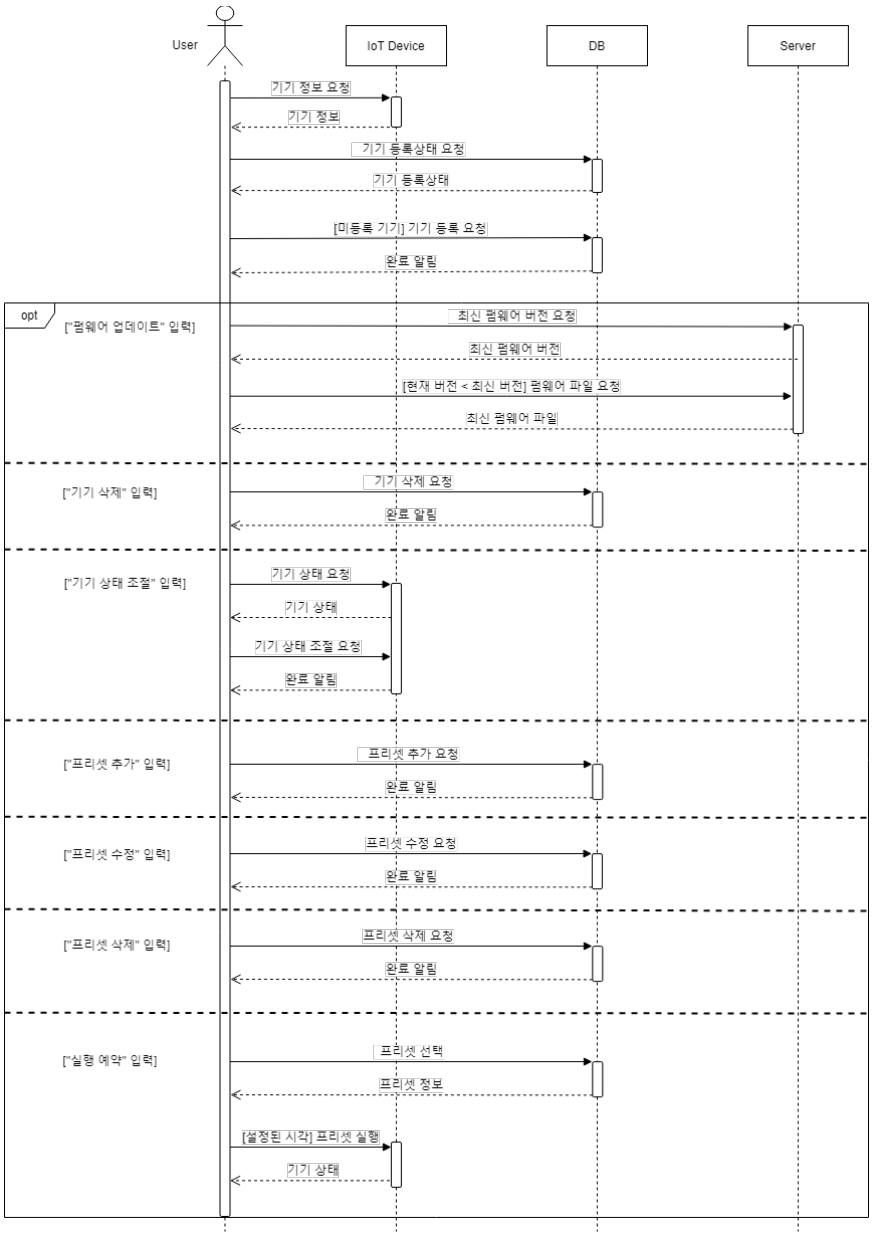

그림 2 System Organization - Sequence Diagram

3.2.3 Use Case Diagram

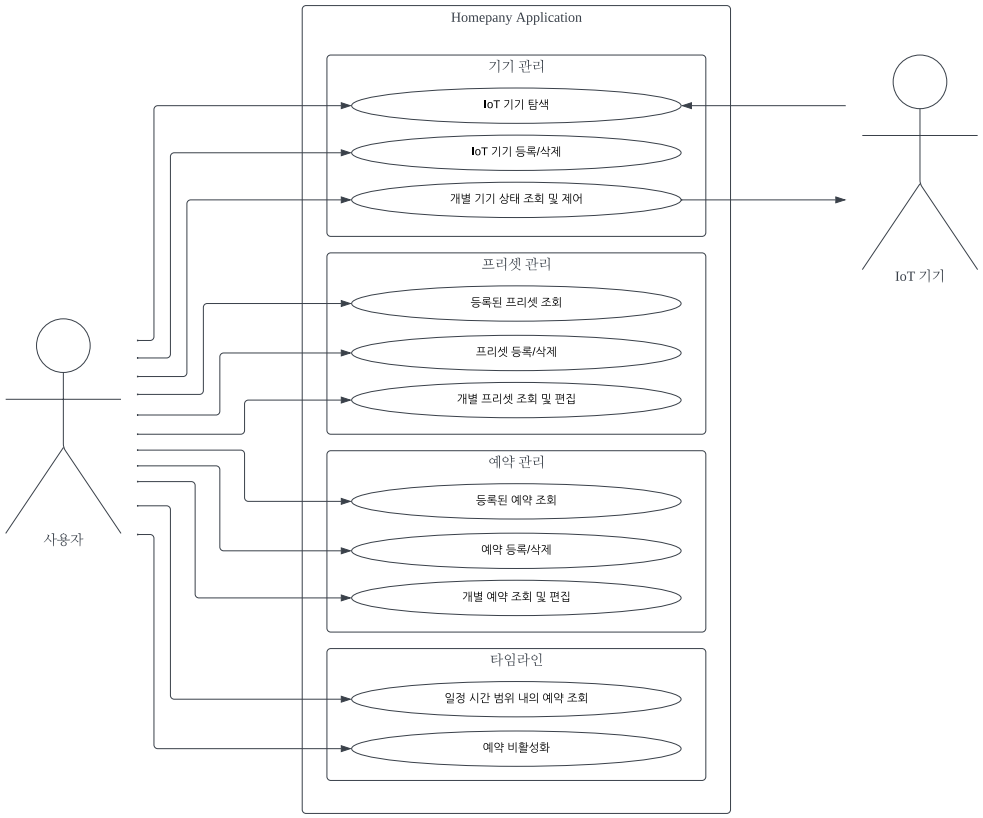

그림 3 System Organization – Use Case Diagram

4. System Architecture - Frontend

4.1 Objectives

이 장에서는 System architecture 를 이루는 Component 들을 Class Diagram 과 Sequence Diagram 을 이용하여 설명한다.

4.2 Subcomponents

4.2.1 Profile

4.2.1.1 Attributes

다음은 Profile object 를 구성하는 attribute 이다.

- user_id: 로그인시 사용되는 사용자 기기의 고유한 id
- nickname: 사용자 기기의 별명

4.2.1.2 Methods

다음은 Profile object 에서 제공하는 method 이다.

- setProfile(user_id):success message
- getProfile(user_id):success message
- showProfile()

4.2.1.3 Class Diagram

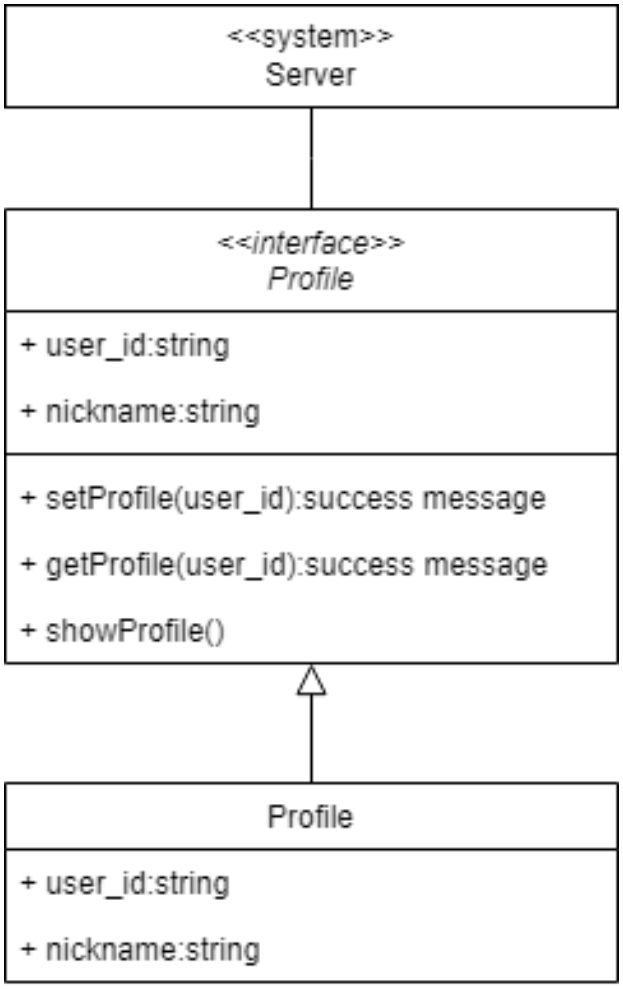

그림 4 Profile - Class Diagram

4.2.1.4 Sequence Diagram

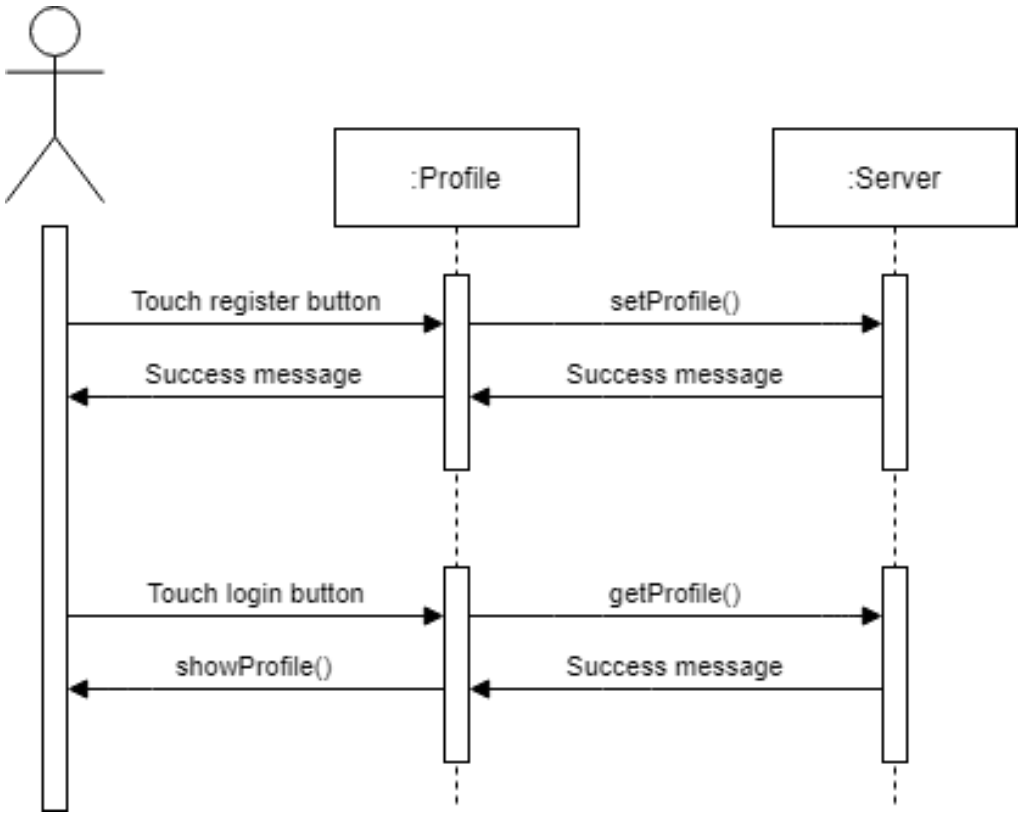

그림 5 Profile - Sequence Diagram

4.2.2 MainPage

4.2.2.1 Attributes

다음은 MainPage object 를 구성하는 attribute 이다.

- page_list:메인 페이지의 하위 페이지 리스트이다.
- user_id:현재 접속해 있는 유저의 id 이다.

4.2.2.2 Methods

다음은 MainPage object 에서 제공하는 method 이다.

- getPages(user_id)
- showPages(user_id, page_list)
- enterPage(Page)
- showPage(user_id, page_id)

4.2.2.3 Class Diagram

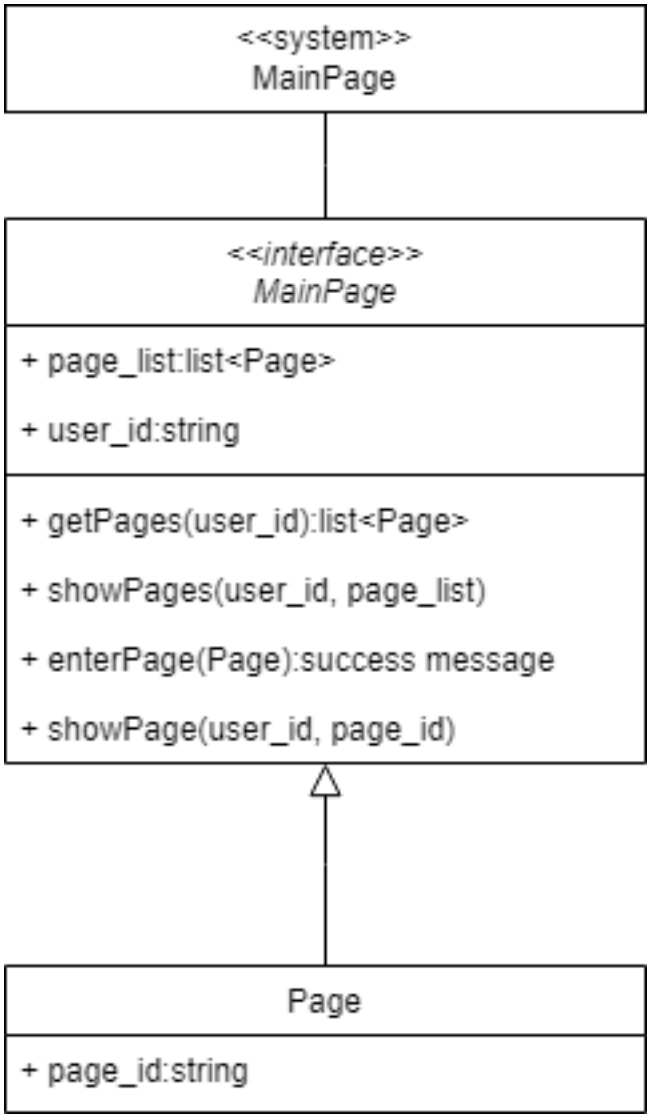

그림 6 MainPage - Class Diagram

4.2.2.4 Sequence Diagram

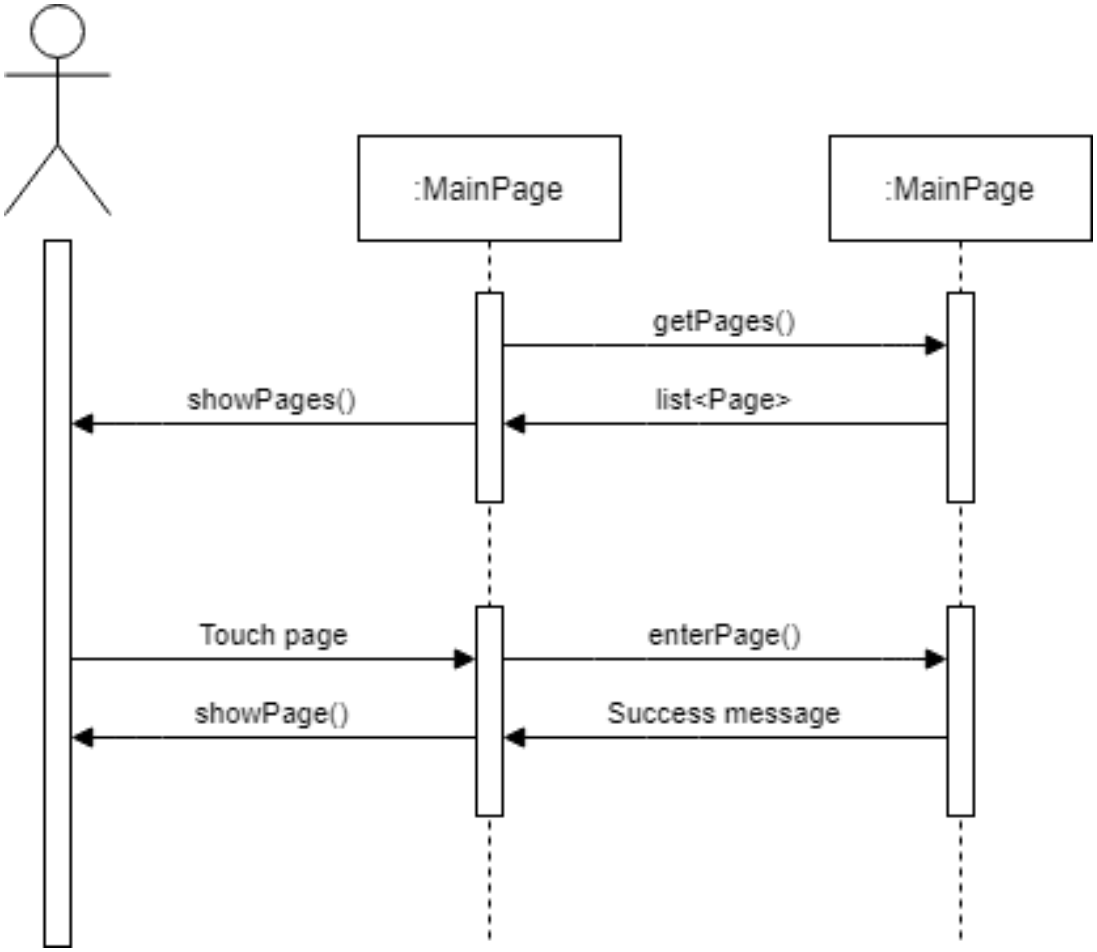

그림 7 MainPage - Sequence Diagram

4.2.3 ManageDevice

4.2.3.1 Attributes

다음은 ManageDevice object 를 구성하는 attribute 이다.

- user_id: 로그인시 사용되는 사용자 기기의 고유한 id
- nickname: 사용자 기기의 별명
- RegisteredDevices: 등록된 IoT 기기들의 리스트이다.
- SearchedDevices: 탐색된 IoT 기기들의 리스트이다.

4.2.3.2 Methods

다음은 ManageDevice object 에서 제공하는 method 이다.

- SearchDevices()
- registerDevice(Device)
- addDevice(Device)
- getRegisteredDevices()
- removeRegisteredDevice(RegisteredDevice)

4.2.3.3 Class Diagram

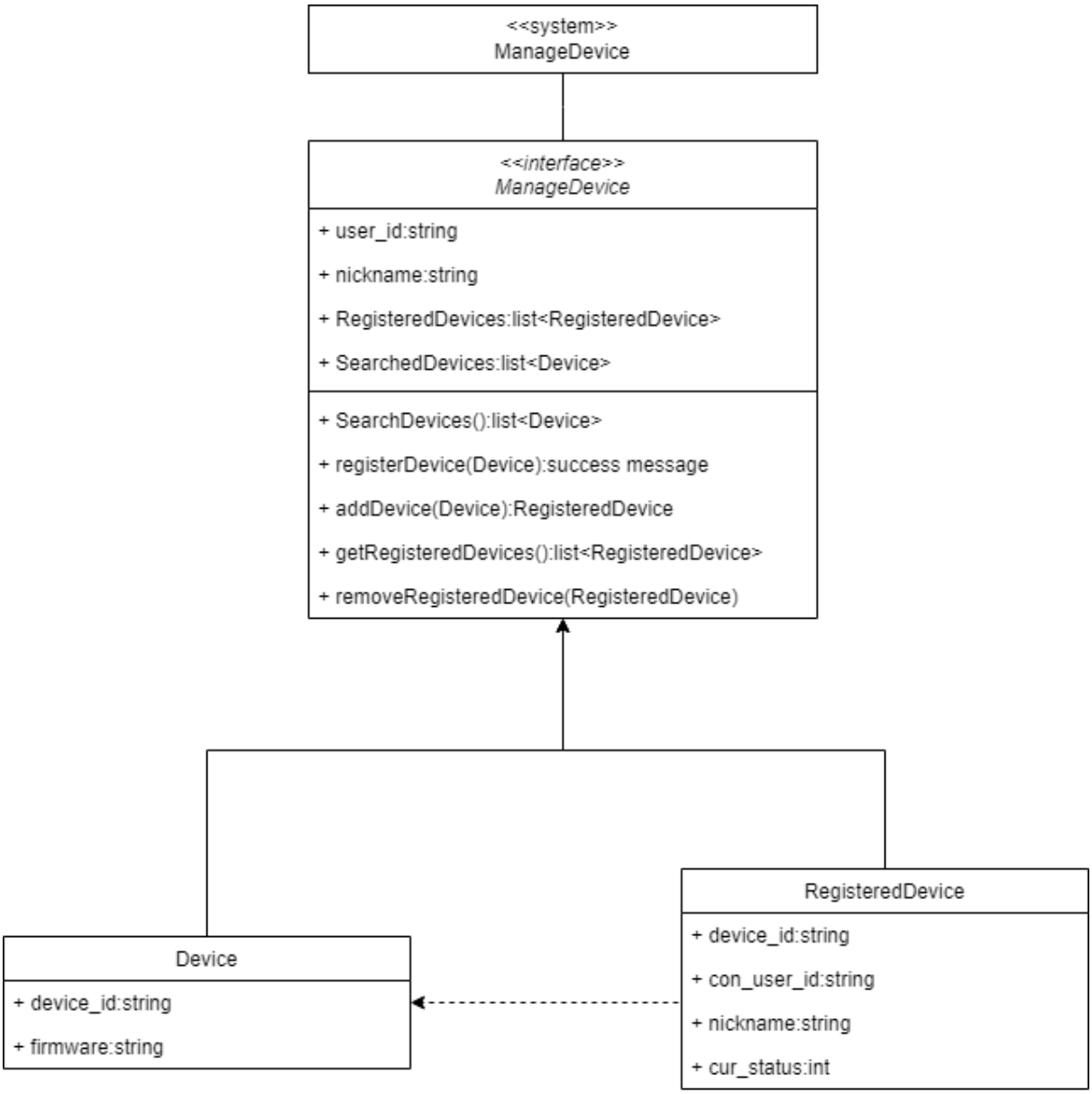

그림 8 ManageDevcie - Class Diagram

4.2.3.4 Sequence Diagram

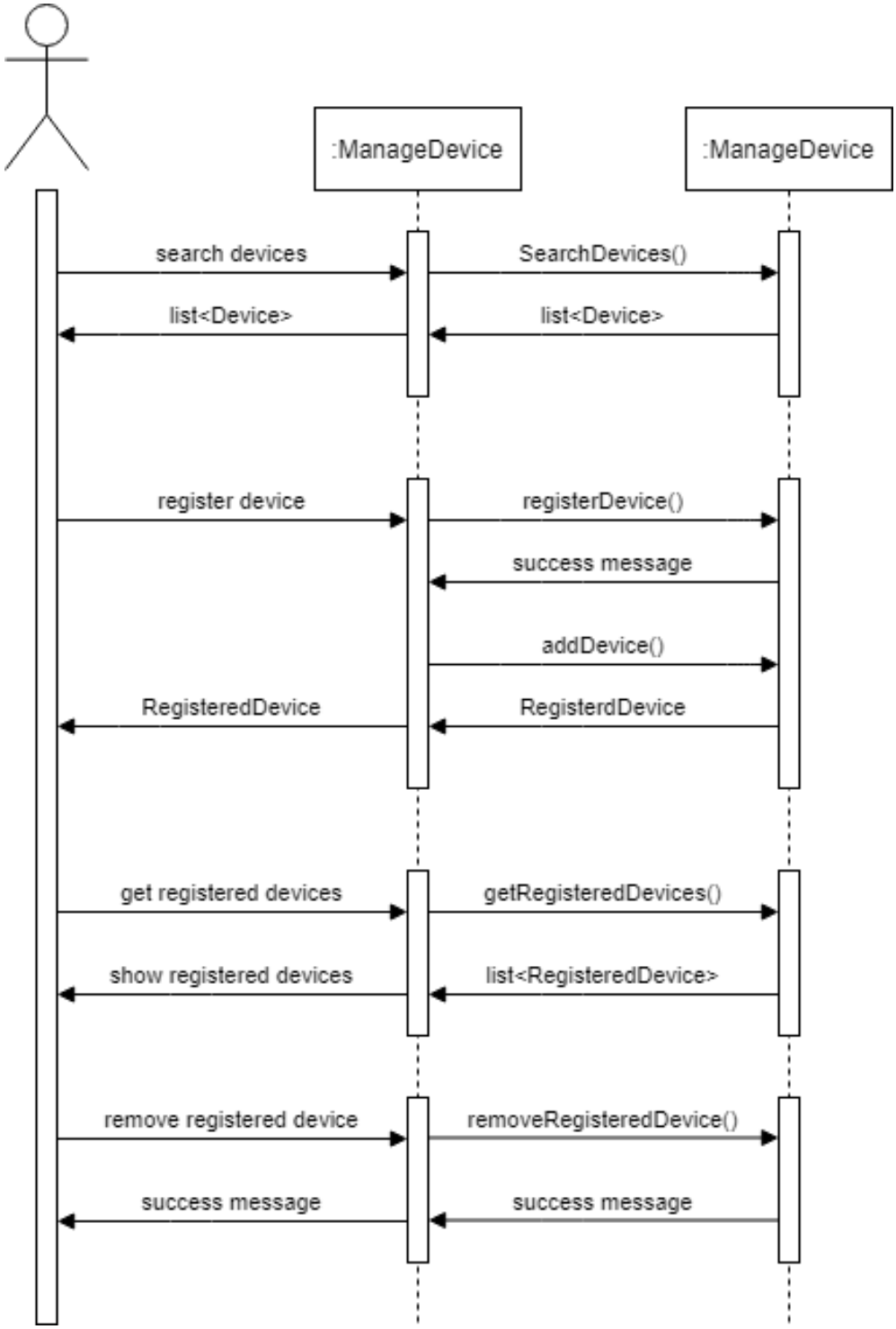

그림 9 ManageDevice - Sequence Diagram

4.2.4 DeviceDetail

4.2.4.1 Attributes

다음은 DeviceDetail object 를 구성하는 attribute 이다.

- Device: 현재 편집하고자 하는 IoT 기기이다.
- device_id: IoT 기기의 고유번호이다.
- attribute_change: IoT 기기를 변경하기 위한 상태값이다..
- firmware: 현재 IoT 기기의 펌웨어 버전이다.
- modified_user: IoT 기기의 상태를 마지막으로 변경한 유저이다.

4.2.4.2 Methods

다음은 DeviceDetail object 에서 제공하는 method 이다.

- updateFirmware(Device)
- checkFirmware(Device)
- modifyDevice(Device, attribute_change)
- getDeviceStatus(Device)

4.2.4.3 Class Diagram

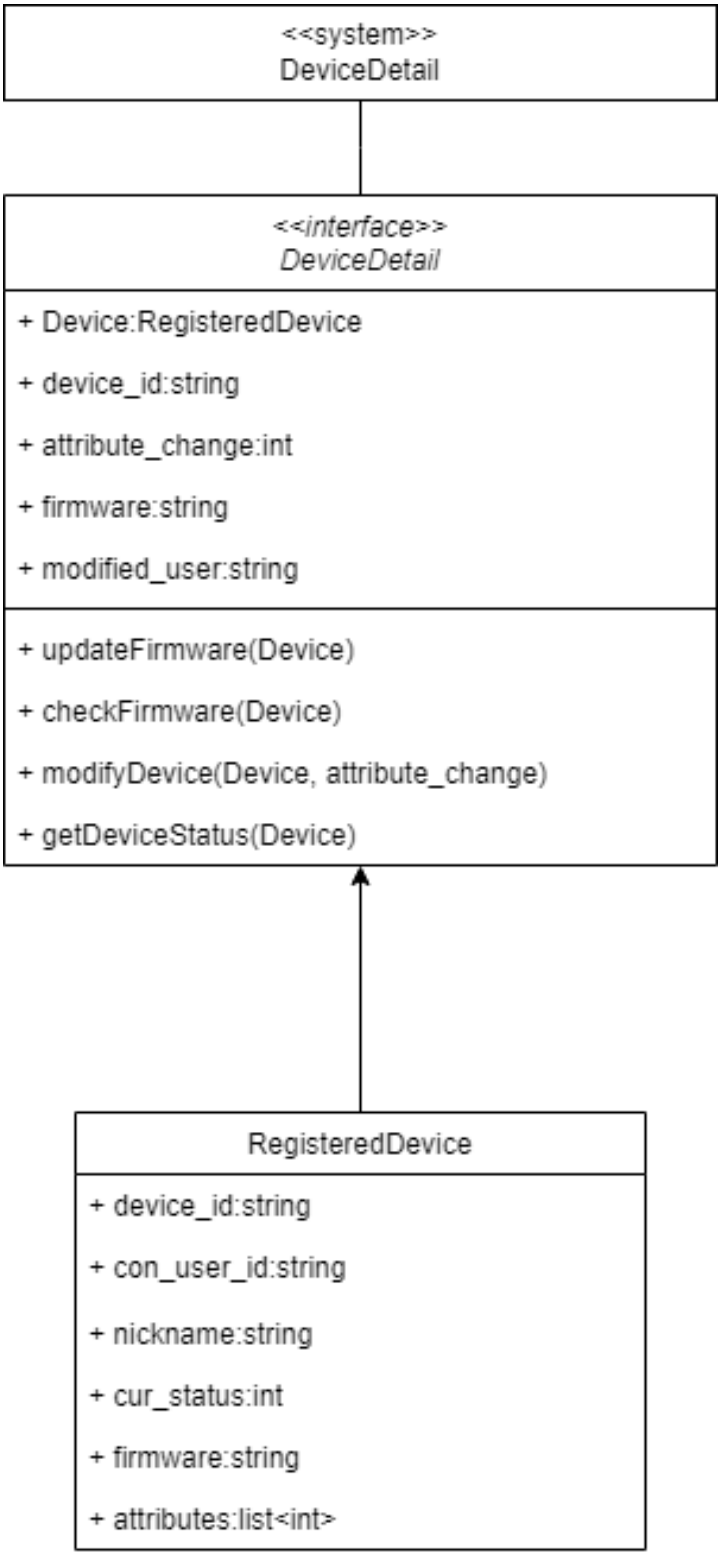

그림 10 DeviceDetail - Class Diagram

4.2.4.4 Sequence Diagram

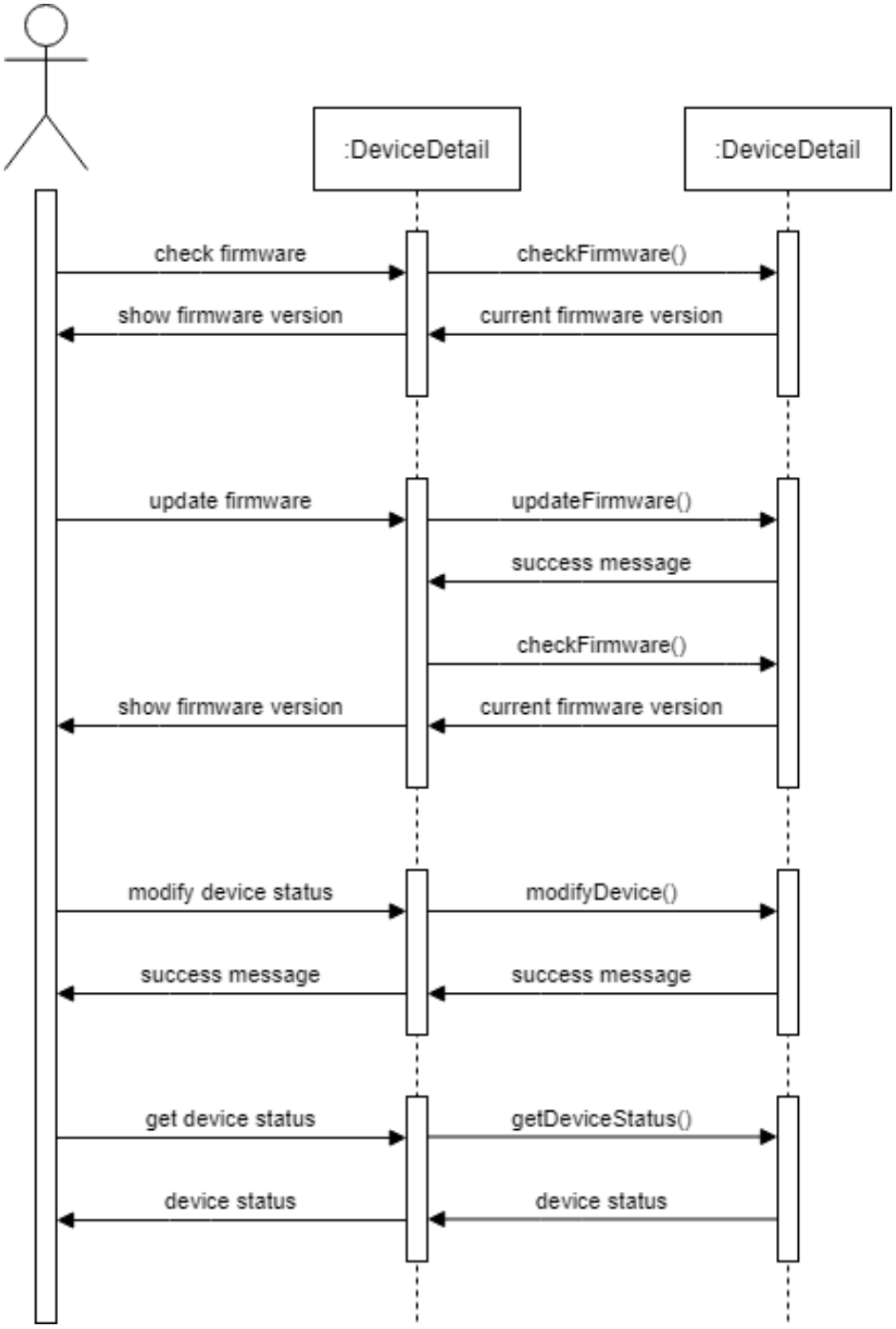

그림 11 DeviceDetail - Sequence Diagram

4.2.5 PresetPage

현재 사용자의 모든 프리셋을 불러오고, 사용자는 목록의 각 프리셋을 상세 조회하거나 복제 및 삭제할 수 있다. 혹은 새 프리셋을 생성하는 페이지로 진입할 수 있다.

4.2.5.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- preset_list: 프리셋들의 목록이다.
- current_page: 생성된 현재 페이지 개체이다.

4.2.5.2 Methods

- generatePage(user_id, List<Preset>): 프리셋 목록을 토대로 현재 페이지를 생성한다.
- openPresetDetailPage(user_id, preset_id): 선택한 프리셋의 상세 페이지를 연다.
- openNewPresetPage(user_id): 새 프리셋 생성 페이지를 연다.
- copyPreset(preset_id): 해당 프리셋을 복제하고, 성공 여부를 확인하여 알린다.
- deletePreset(preset_id): 해당 프리셋을 삭제하고, 성공 여부를 확인하여 알린다.

4.2.5.3 Class Diagram

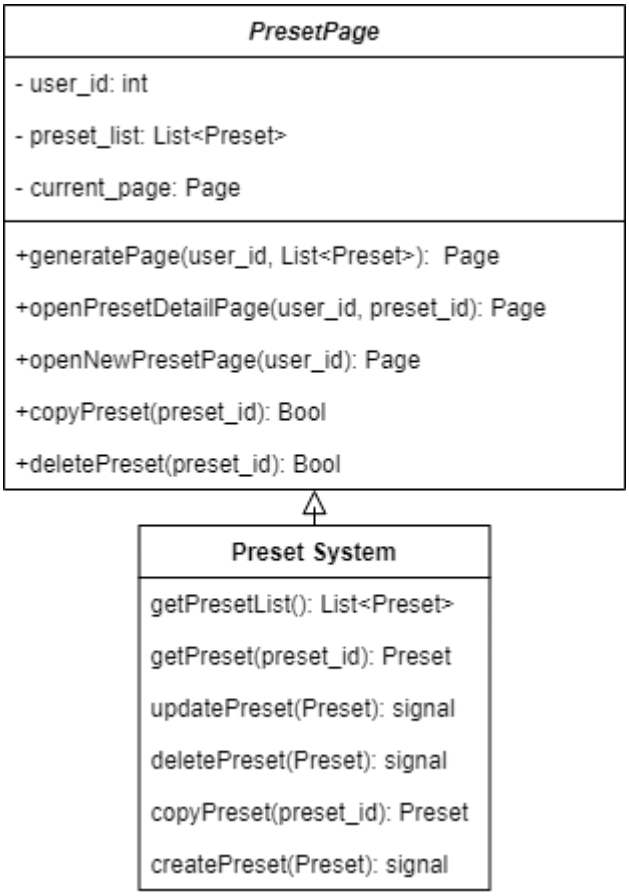

그림 12 PresetPage - Class Diagram

4.2.5.4 Sequence Diagram

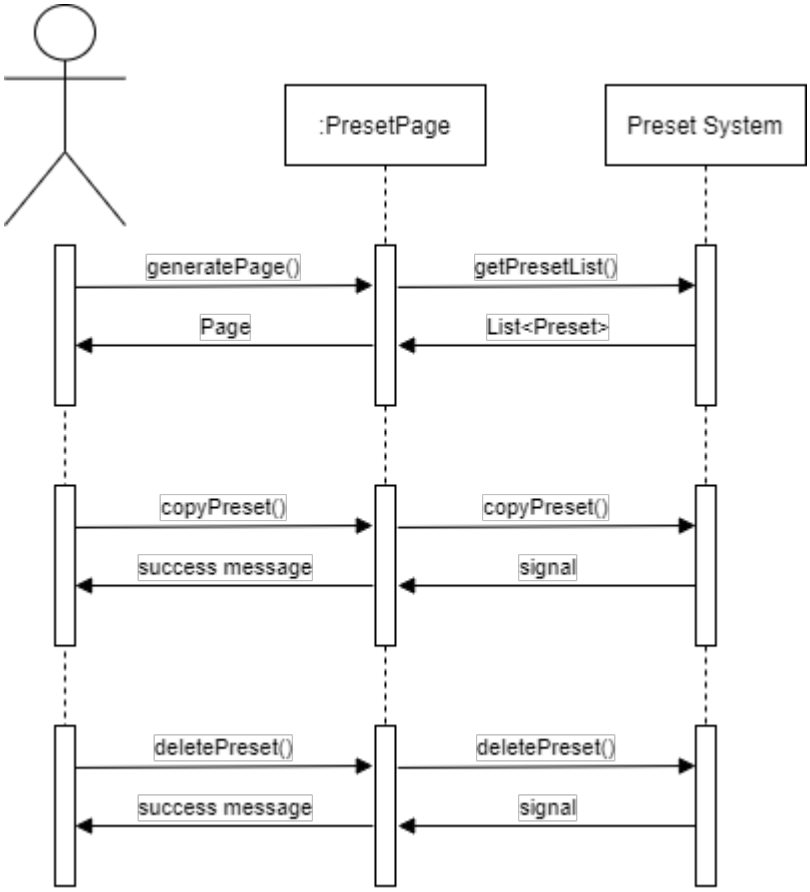

그림 13 PresetPage - Sequence Diagram

4.2.6 PresetDetailPage

선택한 프리셋의 상세 정보를 불러오고, 사용자는 이 페이지에서 프리셋을 편집하거나 삭제할 수 있다.

4.2.6.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- preset_id: 프리셋의 고유번호이다.
- iot_device_id: 프리셋이 제어할 IoT 기기의 고유번호이다.
- attr_control: 프리셋이 제어할 IoT 기기의 속성이다.
- ctrl_degree: 프리셋이 IoT 기기 속성에 대입할 값이다.
- color_code: 프리셋이 화면에 표시될 색상이다.
- current_page: 생성된 현재 페이지 개체이다.

4.2.6.2 Methods

- generatePage(user_id, preset_id): 해당 프리셋의 상세 페이지를 생성한다.
- updatePreset(preset_id): 해당 프리셋을 갱신하고, 성공 여부를 확인하여 알린다.
- deletePreset(preset_id): 해당 프리셋을 삭제하고, 성공 여부를 확인하여 알린다.

4.2.6.3 Class Diagram

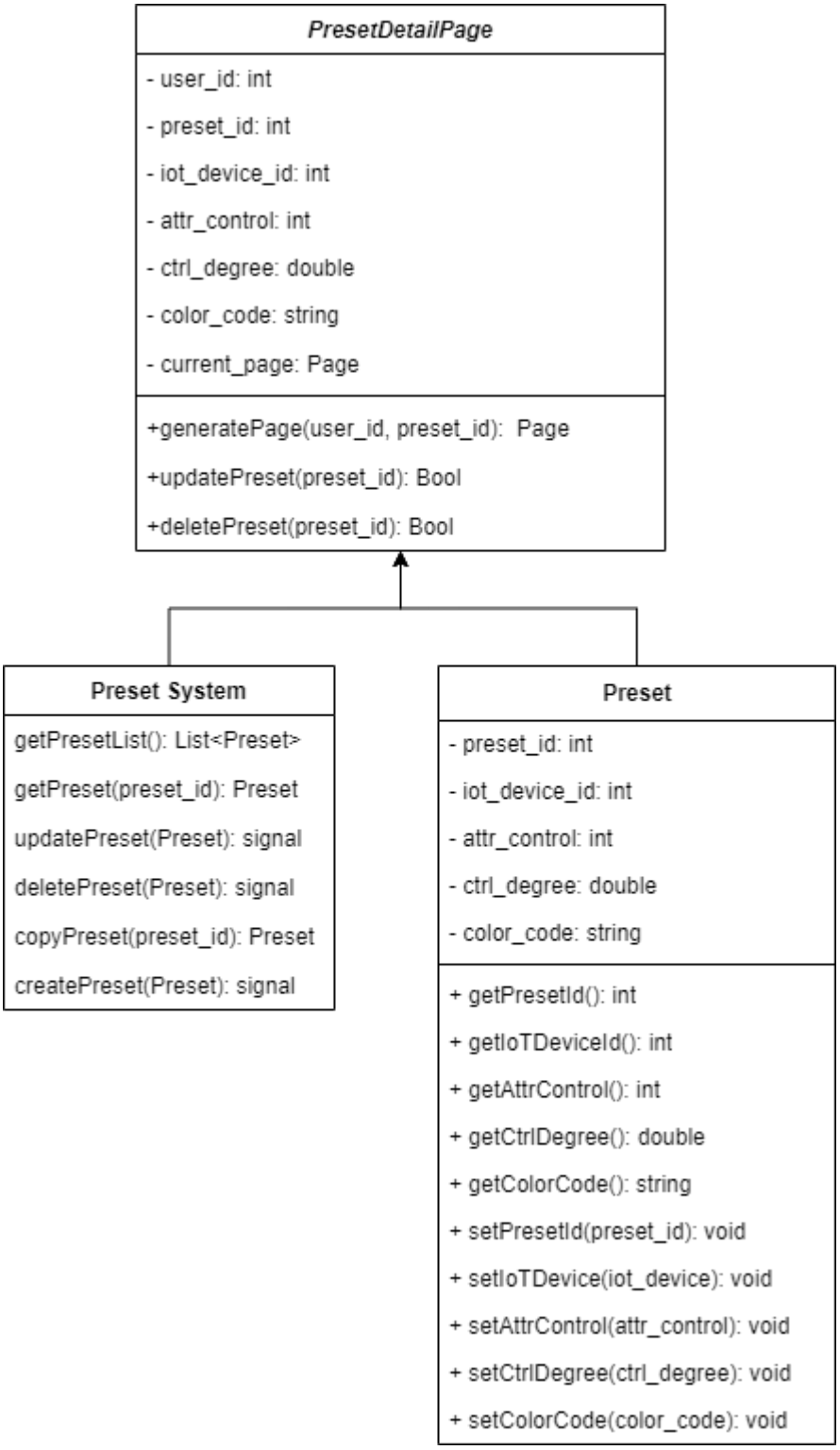

그림 14 PresetDetailPage - Class Diagram

4.2.6.4 Sequence Diagram

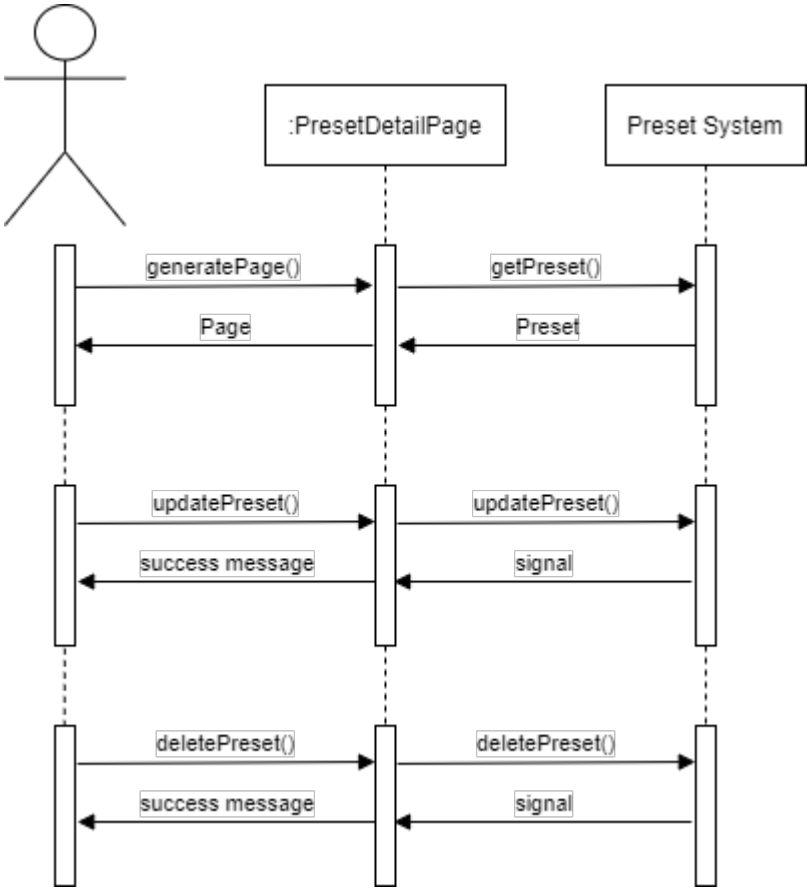

그림 15 PresetDetailPage - Sequence Diagram

4.2.7 NewPresetPage

사용자가 입력한 값들을 토대로 새 프리셋을 추가할 수 있다.

4.2.7.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- preset_id: 프리셋의 고유번호이다.
- iot_device_id: 프리셋이 제어할 IoT 기기의 고유번호이다.
- attr_control: 프리셋이 제어할 IoT 기기의 속성이다.
- ctrl_degree: 프리셋이 IoT 기기 속성에 대입할 값이다.
- color_code: 프리셋이 화면에 표시될 색상이다.
- current_page: 생성된 현재 페이지 개체이다.

4.2.7.2 Methods

- generatePage(user_id): 새 프리셋 페이지를 생성한다.
- createPreset(Preset): 새 프리셋을 생성하고, 성공 여부를 확인하여 알린다.

4.2.7.3 Class Diagram

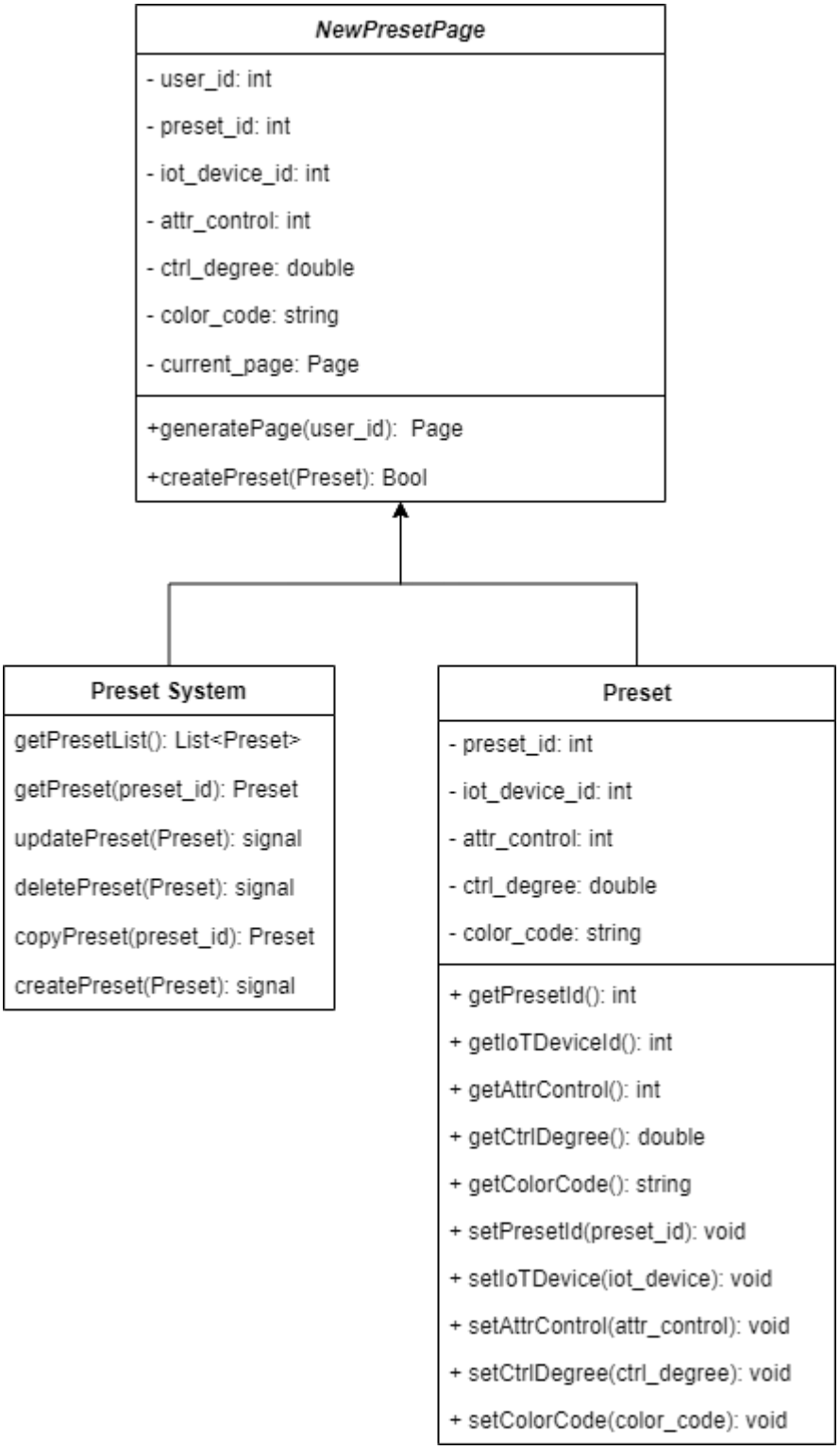

그림 16 NetPresetPage - Class Diagram

4.2.7.4 Sequence Diagram

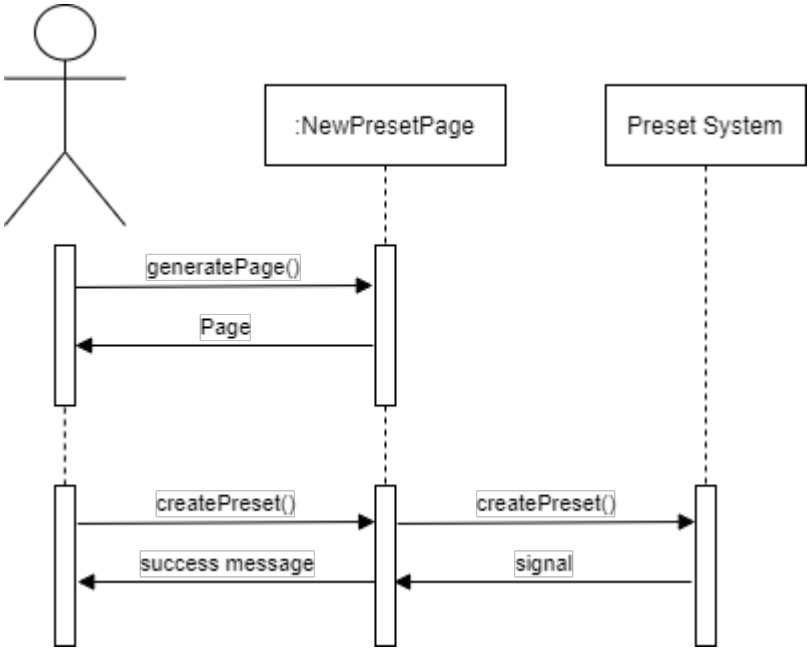

그림 17 NewPresetPage - Sequence Diagram

4.2.8 TimelinePage

현재 사용자의 모든 타임라인을 목록으로 보여주고, 사용자는 각 타임라인을 상세 조회하거나 삭제할 수 있다.

4.2.8.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- timeline_list: 타임라인들의 목록이다.
- current_page: 생성된 현재 페이지 개체이다.

4.2.8.2 Methods

- generatePage(user_id, List<Timeline>): 타임라인 목록을 토대로 타임라인 페이지를 생성한다.
- openTimelineDetailPage(user_id, timeline_id): 해당 타임라인의 상세 페이지를 연다.
- openNewTimelinePage(user_id): 새 타임라인 생성 페이지를 연다.
- deleteTimeline(timeline_id): 해당 타임라인을 삭제하고, 성공 여부를 확인하여 알린다.

4.2.8.3 Class Diagram

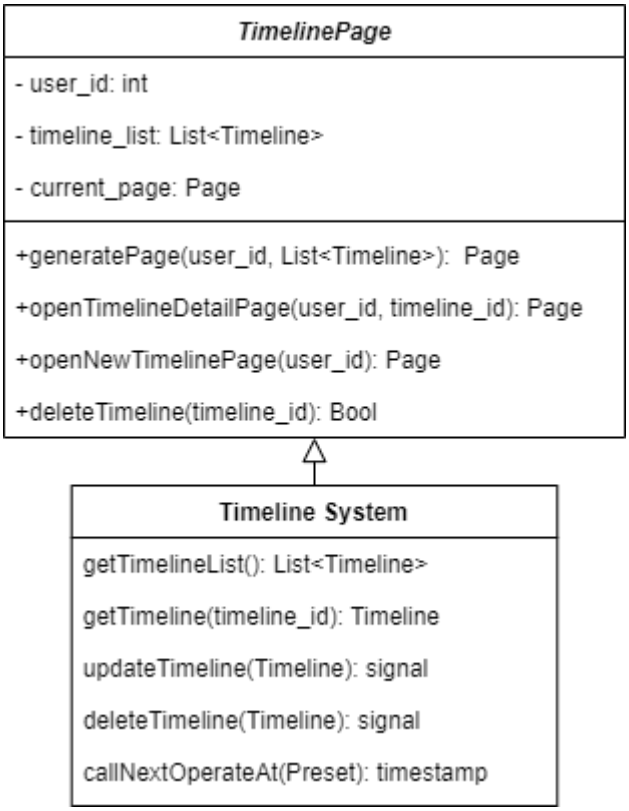

그림 18 TimelinePage - Class Diagram

4.2.8.4 Sequence Diagram

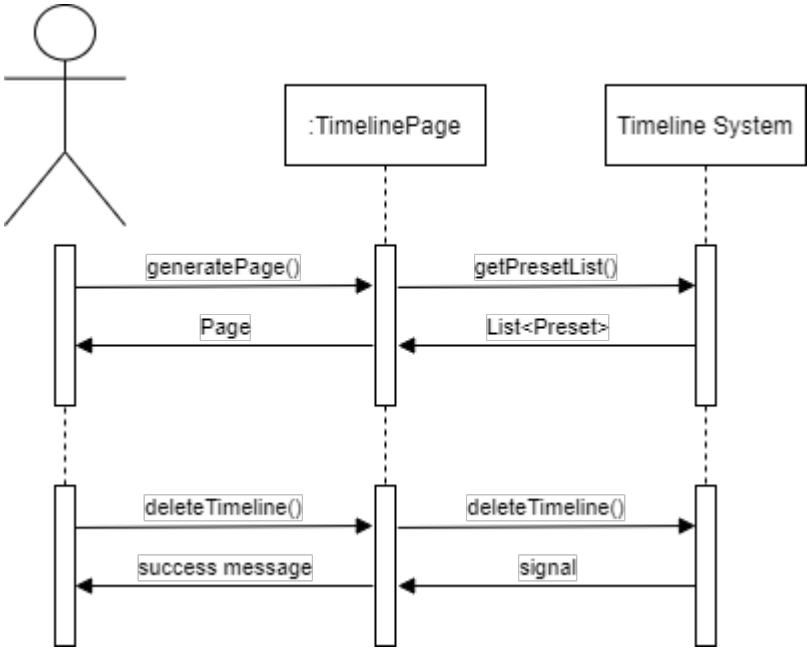

그림 19 TimelinePage - Sequence Diagram

4.2.9 TimelineDetailPage

선택한 타임라인의 상세 정보를 불러오고, 사용자는 이 타임라인을 편집하거나 삭제할 수 있다.

4.2.9.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- timeline_id: 해당 타임라인의 고유번호이다.
- preset_id: 해당 타임라인이 실행할 프리셋의 고유번호이다.
- next_operate_at: 해당 타임라인이 다음으로 실행될 시간이다.
- rtype: 해당 타임라인의 반복 종류(주기)이다.
- duration: 해당 타임라인이 반복될 기간이다.

4.2.9.2 Methods

- generatePage(user_id, timeline_id): 해당 타임라인의 상세 페이지를 생성한다.
- updateTimeline(timeline_id): 해당 타임라인을 갱신하고, 성공 여부를 확인하여 알린다.
- deleteTimeline(timeline_id): 해당 타임라인을 삭제하고, 성공 여부를 확인하여 알린다.

4.2.9.3 Class Diagram

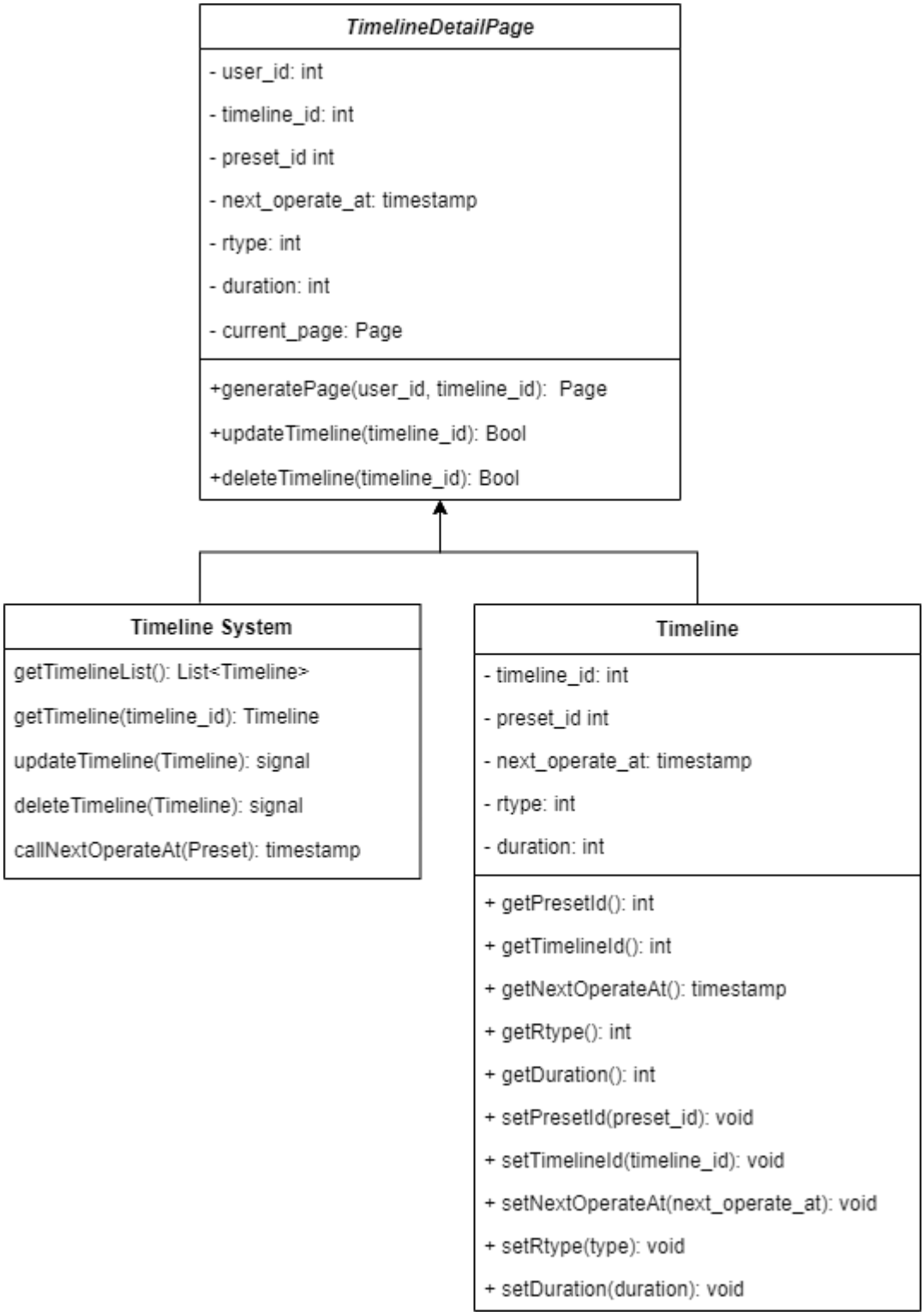

그림 20 TimelineDetailPage - Class Diagram

4.2.9.4 Sequence Diagram

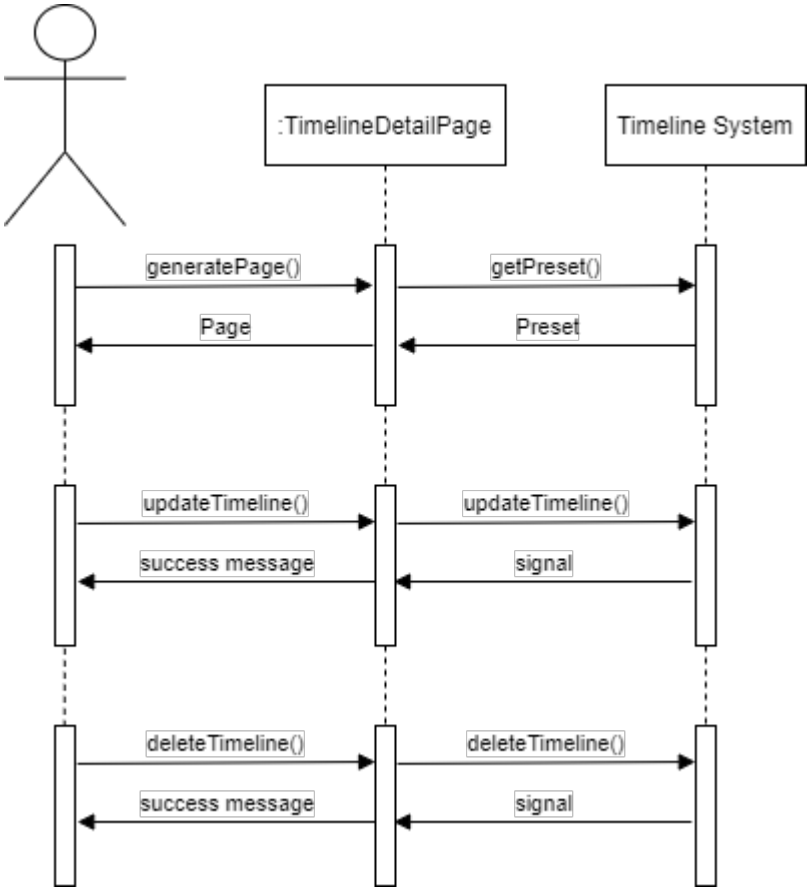

그림 21 TimelineDetailPage - Sequence Diagram

4.2.10 NewTimelinePage

사용자가 입력한 값들을 토대로 새 타임라인을 추가할 수 있다.

4.2.10.1 Attributes

- user_id: 현재 사용자의 고유번호이다.
- timeline_id: 해당 타임라인의 고유번호이다.
- preset_id: 해당 타임라인이 실행할 프리셋의 고유번호이다.
- next_operate_at: 해당 타임라인이 다음으로 실행될 시간이다.
- rtype: 해당 타임라인의 반복 종류(주기)이다.
- duration: 해당 타임라인이 반복될 기간이다.

4.2.10.2 Methods

- generatePage(user_id): 새 타임라인 페이지를 생성한다.
- createTimeline(Timeline): 새 타임라인을 생성하고, 성공 여부를 확인하여 알린다.

4.2.10.3 Class Diagram

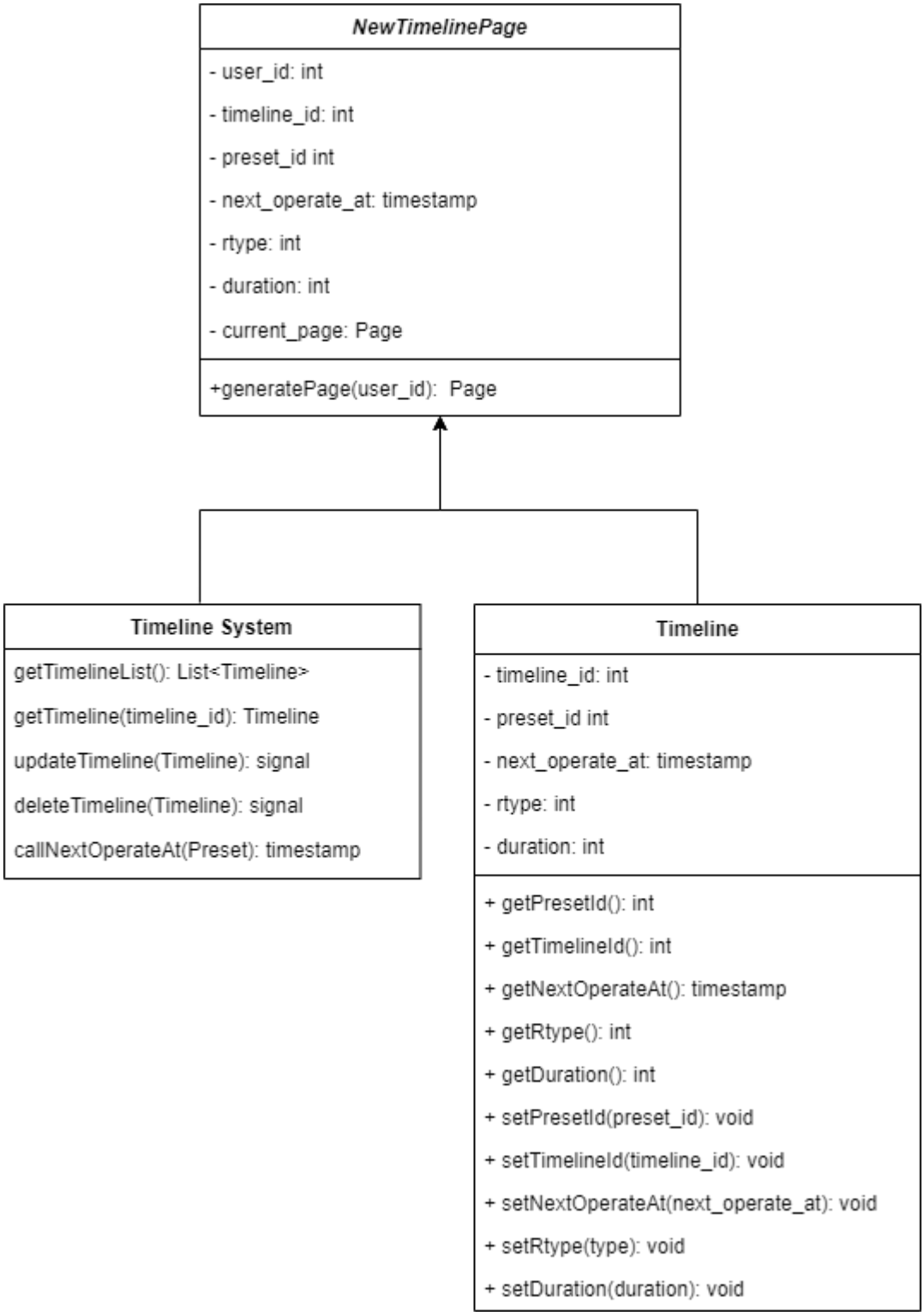

그림 22 NewTimelinePage – Class Diagram

4.2.10.4 Sequence Diagram

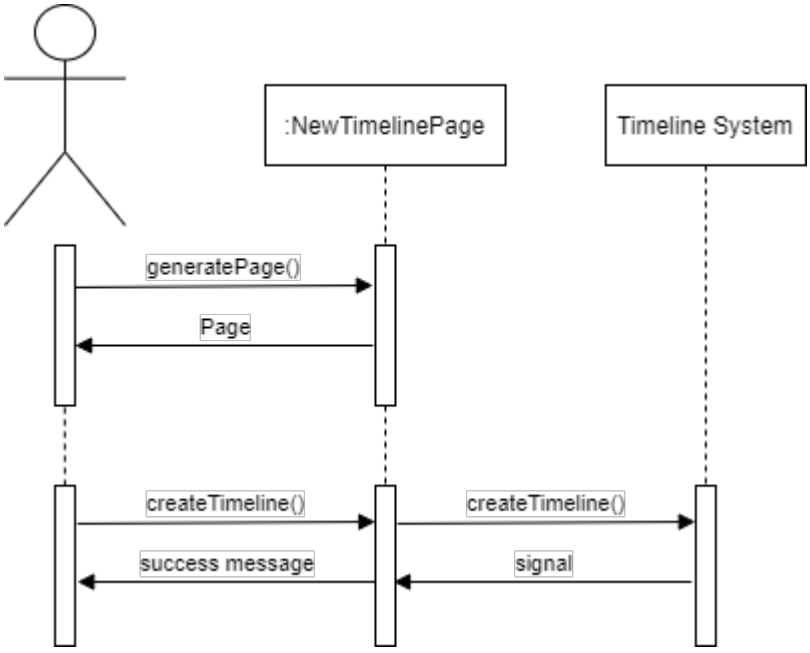

그림 23 NewTimelinePage - Sequence Diagram

5. System Architecture - Backend

5.1 Objectives

이 장에서는 본 시스템에서 Back-end 파트의 기능과 구성 요소 등을 AWS RDS 를 이용하여 구현하는 방법과 구현 구조에 대하여 기술한다.

5.1 Overall Architecture

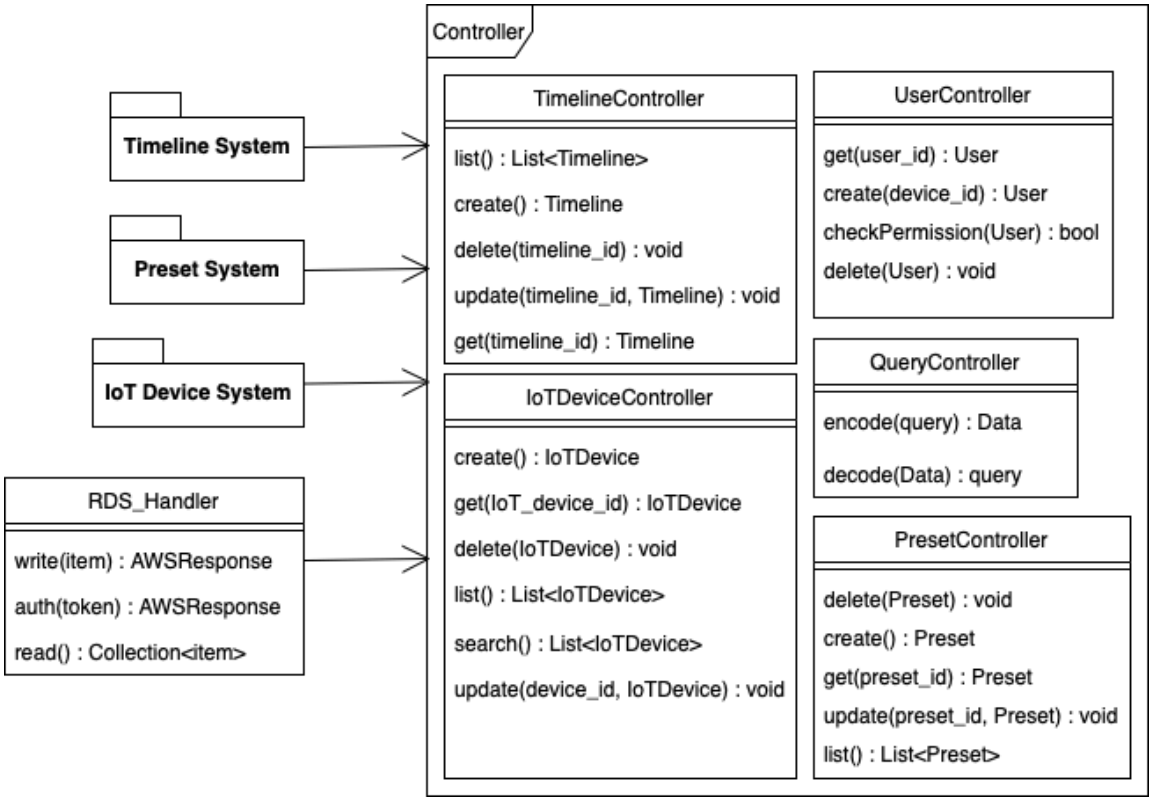

그림 24 Back-end Overall Architecture

위 그림은 전체적인 Back-end 시스템의 구조를 요약하고 있다. AWS RDS 를 이용한 Back-end 시스템은, System, Handler, 그리고 Controller 로 구성되어 있다. 유저는 유니크한 기기 정보를 이용하여, 앱 실행 시에 자동으로 로그인 처리가 된다.

로그인으로 얻게 된 토큰을 이용하여, 유저는 시스템 및 DB 에 접근 권한을 가지게 된다. 로그인 정보는 디바이스에 캐싱되어 있으며, User Controller 의 checkPermission()을 통해 기능 혹은 시스템 접근 시에 검사한다.

5.1.1 System

시스템이란, Homepany 서비스에서 이용할 수 있는 세부 기능을 처리하는 역할을 하며 Preset System, IoT Device System, Timeline System 으로 이루어져 있다.

5.1.2 Handler

Handler 는 AWS RDS Handler 가 있으며, 이는 유저 인증을 통해 AWS RDS 에 접근을 가능하게 하며, 타겟으로 하는 DB 에 read/write 와 같은 기능을 제공하고 operation 의 결과를 반환하는 인터페이스 역할을 한다.

5.1.3 Controller

Controller 는 Handler 와 System 사이에 존재하며, Timeline Controller, Preset Controller, IoT Device Controller, Query Controller, User Controller 가 존재한다.

5.2 Subcomponents

5.2.1 IoT Device System

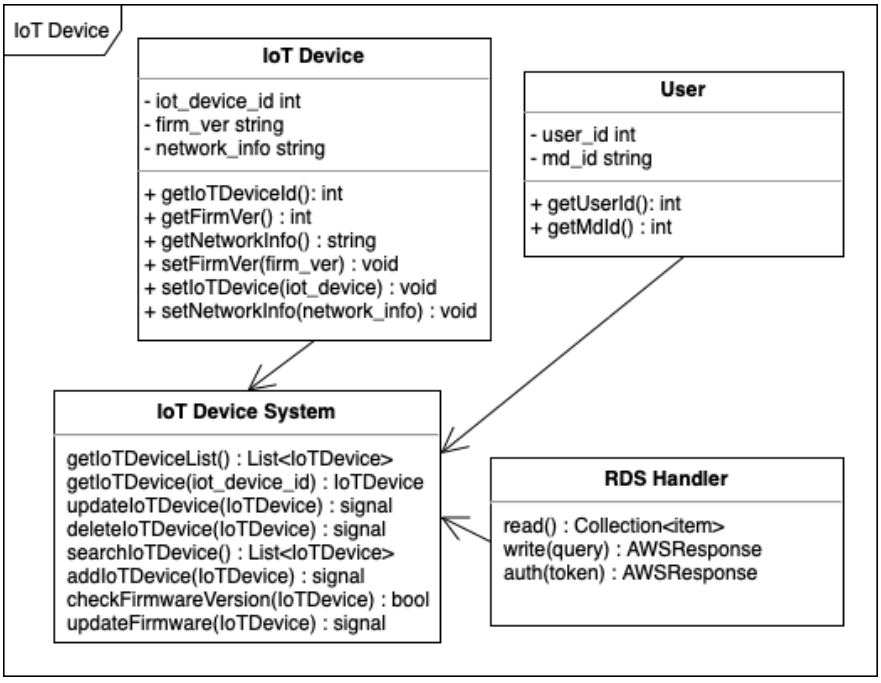

그림 25 IoT Device System – Class Diagram

IoT 기기 시스템은 사용자가 등록한 IoT 기기들을 관리한다. 사용자는 searchIoTDevice()를 통해 네트워크에 존재하는 새로운 IoT 기기를 검색할 수 있으며, 반환된 기기 중 등록을 원하는 IoT 기기는 addIoTDevice()를 통해 DB 에 저장된다.

사용자는 프리셋 등록/변경 시에 등록한 IoT 기기 목록을 getIoTDeviceList()를 통해 불러오게 되는데, 이때 IoT 기기의 펌웨어 버전을 checkFirmwareVersion()을 통해 최신 버전으로 업데이트가 필요한 지 검사한다. 업데이트가 필요할 경우, 서버로부터 반환된 업데이트 정보를 이용하여 기기를 최신 펌웨어로 업데이트한 뒤, updateFirmware()를 통해 변경된 기기 정보를 DB 에 저장한다.

사용자는 deleteIoTDevice()를 통해 등록한 IoT 기기를 삭제할 수 있다.

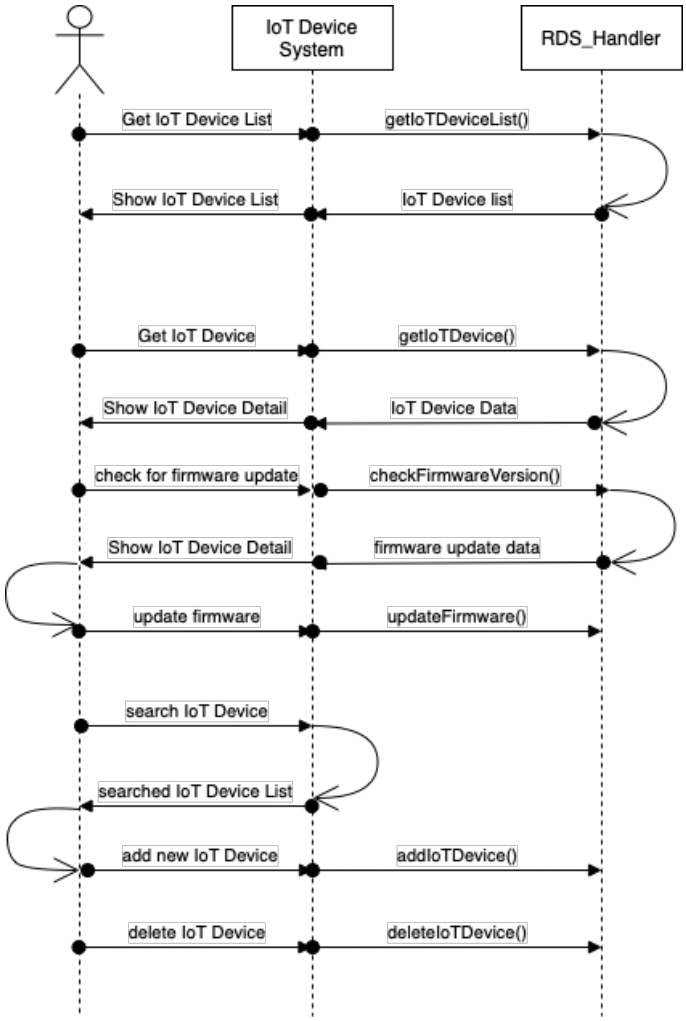

그림 26 IoT Device System - Sequence Diagram

5.2.2 Preset System

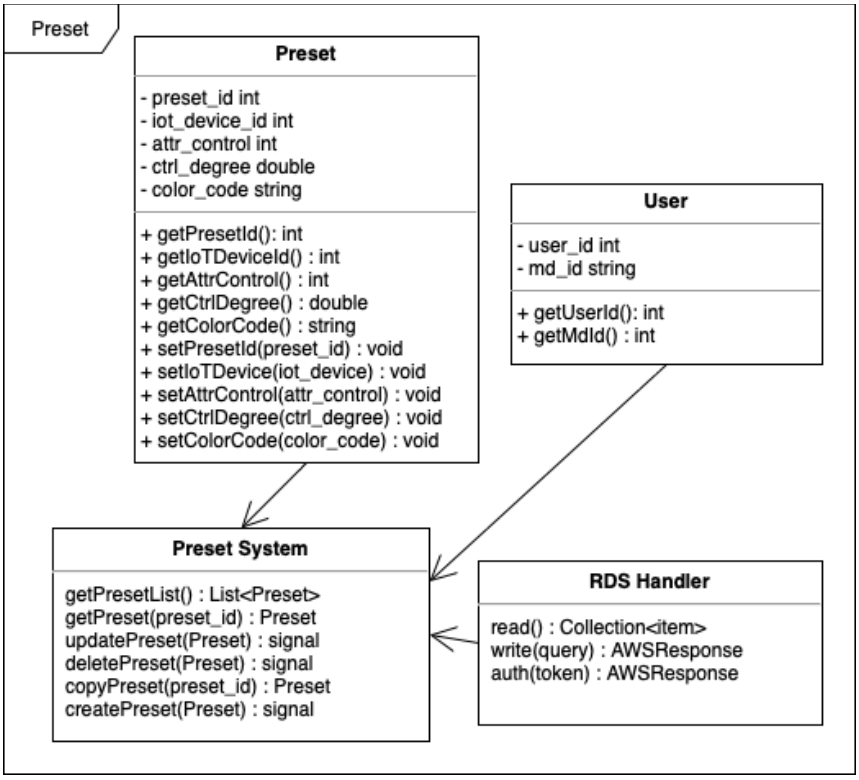

그림 27 Preset System - Class Diagram

사용자는 프리셋 시스템을 통해 등록된 프리셋을 관리할 수 있다. getPresetList()를 통해, 등록된 프리셋 리스트를 가져오고, getPreset()을 통해 원하는 프리셋의 상세 정보를 조회할 수 있다.

이때, 상세 정보에는 프리셋에 등록된 IoT 기기의 정보가 포함되어야 하므로, Preset 클래스의 iot_device_id 를 인자로 한 getIoTDevice()를 통해 IoT 기기의 정보를 불러온다.

상세 정보 창에서는 사용자가 프리셋의 변경을 updatePreset()를 통해 이루어진다. 프리셋의 컬러 코드, 프리셋에 등록된 IoT Device 에 더하여 해당 기기의 기능과 기능의 발현 정도를 조절할 수 있다.

또한, 위 창에서 deletePreset()을 통해 실행할 수 있으며, 현재 프리셋을 copyPreset()을 통해 복사할 수 있다.

프리셋 목록 창에서는 신규 프리셋을 createPreset()을 통해 등록할 수 있다.

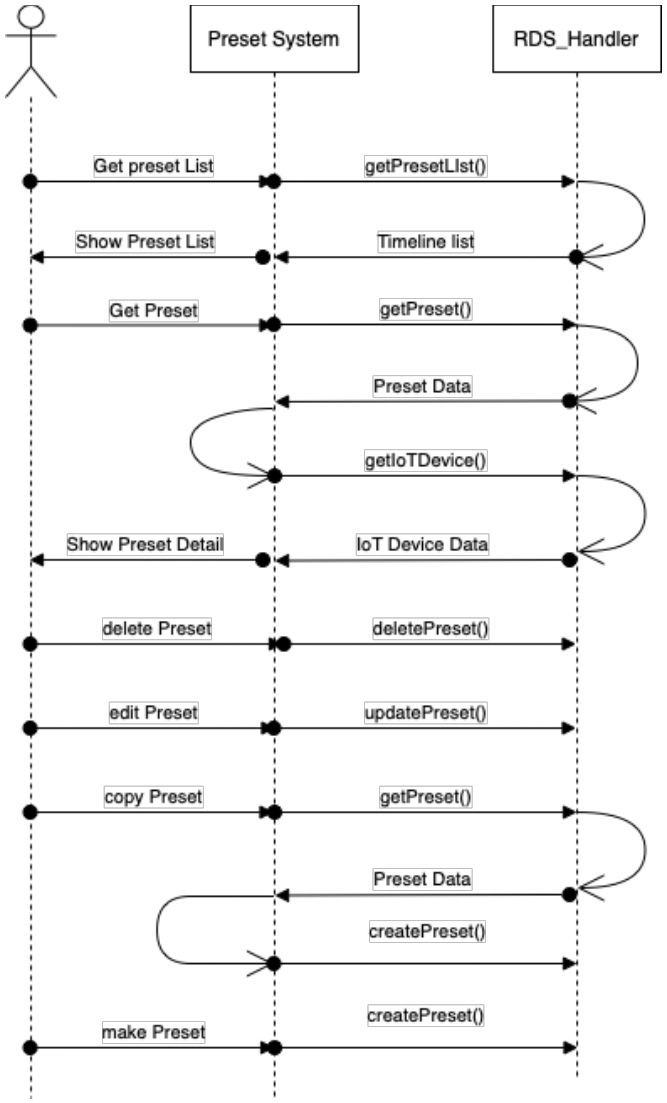

그림 28 Preset System - Sequence Diagram

5.2.3 Timeline System

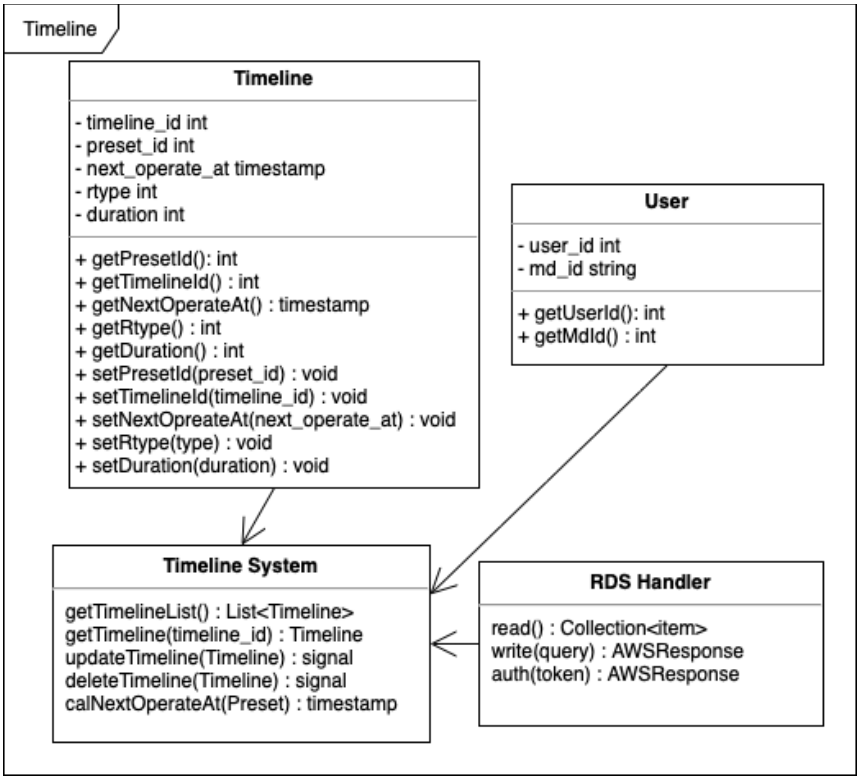

그림 29 Timeline System - Class Diagram

Timeline System 에서는 프리셋 실행 예약 일정을 관리할 수 있다. getTimelineList()를 통해 타임라인 목록이 반환되고, getTimeline()을 통해 원하는 타임라인의 상세 정보를 조회할 수 있다.

상세 정보 창에서는 updateTimeline()을 통해 원하는 일정의 프리셋, 예약 시간, 반복 주기, 지속 시간 등을 변경할 수 있다.

프리셋의 예약 시간이 되면, 다음 실행 시간이 예약 시간이 다른 일정과 겹치는 것을 방지하기 위해, calNextOperateAt()을 통해, 변경 시에 해당 예약 시간이 다른 시간과 겹치지 않으면 반복 주기에 따라 다음 실행 시간을 지정해 준다.

사용자는 일정을 deleteTimeline()을 통해 삭제 가능하다.

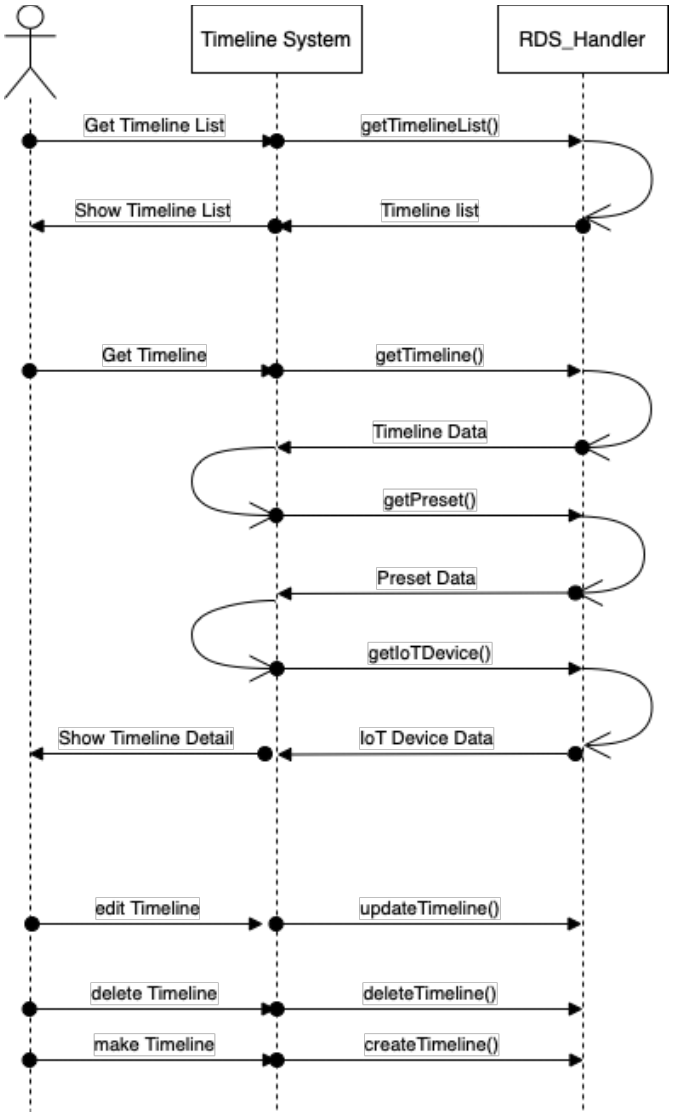

그림 30 Timeline System - Sequence Diagram

6. Protocol Design

6.1 Objectives

이 장에서는 본 시스템의 통신 프로토콜에 대한 내용을 다룬다. 원격 서버, 데이터베이스, Smart Homepany 소프트웨어간 데이터 통신이 어떠한 방식 및 형식으로 이루어지는지 설명하며 이를 준수하는 REST (Representational State Transfer) API 에 대해 기술한다.

6.2 Transfer Protocol: HTTP

상술하였듯이 본 시스템은 REST 아키텍처를 따르며, 이를 통해 설계된 API 를 이용하여 소프트웨어와 Backend 서버 간 데이터 통신을 제공한다. 구체적으로 REST 는 HTTP URI(Uniform Resource Identifier)를 통해 주고 받을 자원(Resource)을 명시하고, HTTP 메소드(i.e., GET, POST, PUT, DELETE)를 통해 해당 자원에 대한 CRUD (Create;Read;Update;Delete) 오퍼레이션을 적용한다. 따라서 Smart Homepany 시스템은 기본적으로 HTTP (HyperText Transfer Protocol)를 그대로 활용한다.

HTTP 는 클라이언트와 서버 사이에 이루어지는 요청 및 응답(Request/Response) 프로토콜이다. 서버는 클라이언트가 요청(Request)한 정보를 보고 알맞은 응답(Response)을 제공한다.

6.3 Data type: JSON

서버와 주고 받는 데이터의 형식은 일반적으로 가장 널리 사용되는 JSON(Javascript Obejct Notation)을 따른다. JSON 은 원하는 데이터에 대한 parsing 작업에 쉬우며 인간과 기계 모두 쉽게 읽을 수 있어 범용적인 활용에 용이하다.

6.4 Account (User, Mobile device)

6.4.1 신규 사용자 등록하기

기능	모바일 디바이스 정보를 기반으로 새로운 사용자를 추가한다.		
URL	{{url}}/user	**METHOD**	POST

파라미터	user_id	string	유저가 설정한 ID (Nickname)
	md_id	string	모바일 디바이스 고유 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	이미 중복되는 user_id 가 존재할 경우	ResultCode: 400	
	이미 중복되는 md_id 가 존재할 경우	ResultCode: 401	
성공 반환 JSON 예시	{ "ResultCode":200, }		
실패 반환 JSON 예시	{ "ResultCode":400 }		

표 1 신규 사용자 등록하기

6.4.2 기존 사용자 식별하기 (로그인 및 권한 확인)

기능	사용자 정보가 서버에 존재하는지 확인한다.		
URL	{{url}}/user	**METHOD**	GET
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	md_id	string	모바일 디바이스 고유 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	400
에러 종류	매칭되는 사용자 정보가 없는 경우	ResultCode: 400	

성공 반환 JSON 예시	{
	"ResultCode":200,
	}

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 2 기존 사용자 식별하기

6.4.3 기기 정보 업데이트

기능	특정 사용자의 기기 정보를 업데이트한다.		
URL	{{url}}/user	**METHOD**	PUT
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	md_id	string	모바일 디바이스 고유 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	매칭되는 user_id 가 없는 경우	ResultCode: 400	
	md_id 가 이미 등록되어 있는 경우	ResultCode: 401	

성공 반환 JSON 예시	{
	"ResultCode":200,
	}		

실패 반환 JSON 예시	{
	"ResultCode":400
	}		

표 3 기기 정보 업데이트

6.4.4 사용자 정보 삭제

기능	기기 정보 및 이에 대응되는 사용자 정보를 삭제한다.		
URL	{{url}}/user	**METHOD**	DELETE
파라미터	md_id	string	모바일 디바이스 고유 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	400
에러 종류	매칭되는 md_id 가 없는 경우	ResultCode: 400	
성공 반환 JSON 예시	{ "ResultCode":200, }		
실패 반환 JSON 예시	{ "ResultCode":400 }		

표 4 사용자 정보 삭제

6.5 IoT device

6.5.1 신규 IoT 기기 등록하기

기능	신규 IoT 기기를 등록한다.		
URL	{{url}}/iot	**METHOD**	POST
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	iot_id	string	IoT 기기 ID
	firm_ver	string	펌웨어 버전

	net_info	string	네트워크 정보
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	이미 중복되는 IoT 기기가 존재할 경우	ResultCode: 400	
	네트워크 정보가 유효하지 않을 경우	ResultCode: 401	
성공 반환 JSON 예시	{ "ResultCode":200, }		
실패 반환 JSON 예시	{ "ResultCode":400 }		

표 5 신규 IoT 기기 등록하기

6.5.2 IoT 기기 정보 불러오기

기능	사용자에게 권한이 있는 IoT 기기의 목록을 불러온다.		
URL	{{url}}/iot/list	**METHOD**	GET
파라미터	user_id	string	유저가 설정한 ID (Nickname)
성공 반환	ResultCode	HTTP status code	200
	IoTList	리스트	사용자의 IoT 기기 목록
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
성공 반환 JSON 예시	{ "ResultCode":200, "IoTList":[		

	<pre>{
	"iot_id":"E06C3823FC827F63E1",
	"firm_ver":"0.0.1",
	"net_info":"D00123043340"
	},
	{
	"iot_id":"842A714A31F8D9AD4",
	"firm_ver":"1.4.1",
	"net_info":"E054363F0340"
	}
	]
	}</pre>

실패 반환 JSON 예시	<pre>{
	"ResultCode":400
	}</pre>

표 6 IoT 기기 정보 불러오기

6.5.3 IoT 기기 목록 불러오기

기능	사용자에게 권한이 있는 IoT 기기의 목록을 불러온다.		
URL	{{url}}/iot/list	**METHOD**	GET
파라미터	user_id	string	유저가 설정한 ID (Nickname)
성공 반환	ResultCode	HTTP status code	200
	IoTList	리스트	사용자의 IoT 기기 목록
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	

성공 반환 JSON 예시	{
	"ResultCode":200,
	"IoTList":[
	{
	"iot_id":"E06C3823FC827F63E1",
	"firm_ver":"0.0.1",
	"net_info":"D00123043340"
	},
	{
	"iot_id":"842A714A31F8D9AD4",
	"firm_ver":"1.4.1",
	"net_info":"E054363F0340"
	}
	]
	}

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 7 IoT 기기 목록 불러오기

6.5.4 IoT 기기 펌웨어 버전 정보 업데이트

기능	특정 IoT 기기의 펌웨어 버전 정보를 업데이트한다.		
URL	{{url}}/iot/firmware	**METHOD**	PUT
파라미터	iot_id	string	IoT 기기 ID
	firm_ver	string	펌웨어 버전
성공 반환	ResultCode	HTTP status code	200

실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 iot_id 에 매칭되는 IoT 기기가 없는 경우	ResultCode: 401	
성공 반환 JSON 예시	{ "ResultCode":200, }		
실패 반환 JSON 예시	{ "ResultCode":400 }		

표 8 IoT 기기 펌웨어 버전 정보 업데이트

6.5.5 IoT 기기 네트워크 정보 업데이트

기능	특정 IoT 기기의 펌웨어 버전 정보를 업데이트한다.		
URL	{{url}}/iot/net-info	**METHOD**	PUT
파라미터	iot_id	string	IoT 기기 ID
	net_info	string	네트워크 정보
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 iot_id 에 매칭되는 IoT 기기가 없는 경우	ResultCode: 401	
성공 반환 JSON 예시	{ "ResultCode":200, }		

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 9 IoT 기기 네트워크 정보 업데이트

6.5.6 IoT 기기 삭제

기능	특정 IoT 기기를 삭제한다.		
URL	{{url}}/iot/net-info	**METHOD**	DELETE
파라미터	iot_id	string	IoT 기기 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 iot_id 에 매칭되는 IoT 기기가 없는 경우	ResultCode: 401	

성공 반환 JSON 예시	{
	"ResultCode":200,
	}		

실패 반환 JSON 예시	{
	"ResultCode":400
	}		

표 10 IoT 기기 삭제

6.6 Preset (Registry, etc.)

6.6.1 프리셋 목록 불러오기

기능	사용자가 등록한 프리셋의 목록을 불러온다.		
URL	{{url}}/preset/list	**METHOD**	GET
파라미터	user_id	string	유저가 설정한 ID (Nickname)
성공 반환	ResultCode	HTTP status code	200
	PresetList	리스트	사용자의 프리셋 목록
실패 반환	ResultCode	HTTP status code	400
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	

성공 반환 JSON 예시	{
	"ResultCode":200,
	"PresetList":[
	{
	"preset_id":1,
	"iot_id":"E06C3823FC827F63C3",
	"attr_control":2,
	"ctrl_degree":3.0,
	"color_code":"0xffffff"
	},
	{
	"preset_id":2,
	"iot_id":"E06C3823FC827F63E3",
	"attr_control":33,
	"ctrl_degree":133.0,
	"color_code":"0xfecd2f"
	}		

	] }

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 11 프리셋 목록 불러오기

6.6.2 특정 프리셋 가져오기

기능	사용자의 프리셋 중, 특정 preset_id 에 해당하는 프리셋을 반환한다.		
URL	{{url}}/preset	**METHOD**	GET
파라미터	preset_id	int	특정 프리셋 id
성공 반환	ResultCode	HTTP status code	200
	Preset	Preset Class	사용자의 특정 프리셋
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 프리셋 ID 에 매핑되는 프리셋이 존재하지 않을 경우	ResultCode: 401	

성공 반환 JSON 예시	{
	"ResultCode":200,
	"Preset": {
	"preset_id":1,
	"iot_id":"E06C3823FC827F63E1",
	"attr_control":2,
	"ctrl_degree":3.0,
	"color_code":"0xffffff"
	}
	}		

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 12 특정 프리셋 가져오기

6.6.3 프리셋 수정하기

기능	사용자의 프리셋 중, 특정 preset_id 에 해당하는 프리셋을 수정한다.		
URL	{{url}}/preset	**METHOD**	PUT
파라미터	preset_id	int	특정 프리셋 ID
	iot_id	string	프리셋에 설정된 IoT 기기 ID
	attr_control	int	프리셋에 설정된 IoT 기기의 기능 번호
	ctrl_degree	double	프리셋에 설정된 IoT 기기의 기능 발현 정도
	color_code	string	프리셋 컬러 코드
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 프리셋 ID 에 매핑되는 프리셋이 존재하지 않을 경우	ResultCode: 401	
	iot_id 에 매핑되는 IoT 기기가 존재하지 않을 경우	ResultCode: 402	
	IoT 기기에 attr_control 에 해당하는 기능이 존재하지 않을 경우	ResultCode: 403	
	ctrl_degree 의 값이 유효하지 않은 경우	ResultCode: 404	
	color_code 가 색상 값 형식에 맞지 않는 경우	ResultCode: 405	
성공 반환 JSON 예시	{ "ResultCode":200		

	}

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 13 프리셋 수정하기

6.6.4 프리셋 삭제하기

기능	사용자의 프리셋 중, 특정 preset_id 에 해당하는 프리셋을 삭제한다.		
URL	{{url}}/preset	**METHOD**	DELETE
파라미터	preset_id	int	특정 프리셋 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 프리셋 ID 에 매핑되는 프리셋이 존재하지 않을 경우	ResultCode: 401	

성공 반환 JSON 예시	{
	"ResultCode":200
	}		

실패 반환 JSON 예시	{
	"ResultCode":400
	}		

표 14 프리셋 삭제하기

6.6.5 프리셋 등록하기

기능	사용자의 지정한 값에 따라 새로운 프리셋을 등록한다.

URL	{{url}}/preset	**METHOD**	POST
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	iot_id	string	프리셋에 설정할 IoT 기기 ID
	attr_control	int	프리셋에 설정할 IoT 기기의 기능 번호
	ctrl_degree	double	프리셋에 설정할 IoT 기기의 기능 발현 정도
	color_code	string	프리셋 컬러 코드
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	iot_id 에 매핑되는 IoT 기기가 존재하지 않을 경우	ResultCode: 401	
	IoT 기기에 attr_control 에 해당하는 기능이 존재하지 않을 경우	ResultCode: 402	
	ctrl_degree 의 값이 유효하지 않은 경우	ResultCode: 403	
	color_code 가 색상 값 형식에 맞지 않는 경우	ResultCode: 404	

성공 반환 JSON 예시	{
	"ResultCode":200
	}		

실패 반환 JSON 예시	{
	"ResultCode":400
	}		

표 15 프리셋 등록하기

6.7 Timeline (Registry, etc.)

6.7.1 타임라인 목록 가져오기

<table>
<tr><td>기능</td><td colspan="3">사용자가 등록한 타임라인(프리셋 예약 목록)을 불러온다.</td></tr>
<tr><td>URL</td><td>{{url}}/timeline/list</td><td>METHOD</td><td>GET</td></tr>
<tr><td>파라미터</td><td>user_id</td><td>string</td><td>유저가 설정한 ID (Nickname)</td></tr>
<tr><td rowspan="2">성공 반환</td><td>ResultCode</td><td>HTTP status code</td><td>200</td></tr>
<tr><td>TimelineList</td><td>리스트</td><td>사용자의 타임라인 목록</td></tr>
<tr><td>실패 반환</td><td>ResultCode</td><td>HTTP status code</td><td>400</td></tr>
<tr><td>에러 종류</td><td>사용자가 인증이 되어 있지 않은 경우</td><td colspan="2">ResultCode: 400</td></tr>
<tr><td>성공 반환 JSON 예시</td><td colspan="3"><pre>
{
    "ResultCode":200,
    "TimelineList":[
        {
            "timeline_id":1,
            "preset_id":2,
            "next_operate_at":1651959083,
            "rtype":2,
            "duration":100
        },
        {
            "timeline_id":2,
            "preset_id":3,
            "next_operate_at":-1,
            "rtype":0,
            "duration":210
</pre></td></tr>
</table>

	}] }

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 16 타임라인 목록 가져오기

6.7.2 특정 타임라인 가져오기

기능	사용자가 등록한 타임라인(프리셋 예약 목록)을 불러온다.		
URL	{{url}}/timeline	**METHOD**	GET
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	timeline_id	int	특정 타임라인 ID
성공 반환	ResultCode	HTTP status code	200
	Timeline	Timeline Class	사용자의 특정 타임라인
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	timeline_id 에 해당하는 타임라인이 존재하지 않을 경우	ResultCode: 401	

성공 반환 JSON 예시	{
	"ResultCode":200,
	"Timeline":{
	"timeline_id":1,
	"preset_id":2,
	"next_operate_at":1651959083,
	"rtype":2,
	"duration":100
	}		

	}

실패 반환 JSON 예시	{
	"ResultCode":400
	}

표 17 특정 타임라인 가져오기

6.7.3 타임라인 수정하기

기능	사용자의 타임라인 중, 특정 timeline_id 에 해당하는 타임라인을 수정한다.		
URL	{{url}}/timeline	**METHOD**	PUT
파라미터	timeline_id	int	특정 타임라인 ID
	preset_id	int	타임라인에 설정한 프리셋 ID
	next_operate_at	timestamp	타임라인이 예약한 시간
	rtype	int	타임라인의 반복 여부 (0: 반복하지 X, 1: 매일 반복, 2: 매주 반복, 3: 매달 반복)
	duration	int	타임라인 예약 지속시간(분)
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	해당 타임라인 ID 에 매핑되는 타임라인이 존재하지 않을 경우	ResultCode: 401	
	preset_id 에 해당하는 프리셋이 존재하지 않을 경우	ResultCode: 402	
	반복 여부 인자가 유효하지 않을 경우	ResultCode: 403	
	계산된 다음 예약 시간이 다른 일정과 겹치는 경우	ResultCode: 404	

	지속 시간 인자가 유효하지 않을 경우	ResultCode: 405

성공 반환 JSON 예시	{
	"ResultCode":200
	}	

실패 반환 JSON 예시	{
	"ResultCode":400
	}	

표 18 타임라인 수정하기

6.7.4 타임라인 생성하기

기능	사용자의 지정한 값에 따라 새로운 타임라인을 등록한다.		
URL	{{url}}/timeline	**METHOD**	POST
파라미터	user_id	string	유저가 설정한 ID (Nickname)
	preset_id	int	타임라인에 설정할 프리셋 ID
	next_operate_at	timestamp	타임라인을 예약할 시간
	rtype	int	타임라인의 반복 여부 (0: 반복하지 X, 1: 매일 반복, 2: 매주 반복, 3: 매달 반복)
	duration	int	타임라인 예약 지속시간(분)
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	preset_id 에 해당하는 프리셋이 존재하지 않을 경우	ResultCode: 401	
	반복 여부 인자가 유효하지 않을 경우	ResultCode: 402	

	계산된 다음 예약 시간이 다른 일정과 겹치는 경우	ResultCode: 403
	지속 시간 인자가 유효하지 않을 경우	ResultCode: 404
성공 반환 JSON 예시	{ "ResultCode":200 }	
실패 반환 JSON 예시	{ "ResultCode":400 }	

표 19 타임라인 생성하기

6.7.5 타임라인 삭제하기

기능	사용자의 타임라인 중, 특정 timeline_id 에 해당하는 타임라인을 삭제한다.		
URL	{{url}}/timeline	**METHOD**	DELETE
파라미터	preset_id	int	타임라인에 설정할 프리셋 ID
성공 반환	ResultCode	HTTP status code	200
실패 반환	ResultCode	HTTP status code	40X
에러 종류	사용자가 인증이 되어 있지 않은 경우	ResultCode: 400	
	timeline_id 에 해당하는 타임라인이 존재하지 않을 경우	ResultCode: 401	
성공 반환 JSON 예시	{ "ResultCode":200 }		
실패 반환 JSON 예시	{ "ResultCode":400 }		

표 20 타임라인 삭제하기

7. Database Design

7.1 Objectives

이 장에서는 본 시스템에서 사용되는 데이터 클래스의 속성 및 구조에 대하여 기술하였다. 시스템에서 운용하는 데이터들의 구조와 이들이 데이터베이스에서 어떻게 표현되는지 상세하게 설명한다. 이 항목에서는 서로 다른 데이터 클래스들 간의 관계를 ER-Diagram (Entity Relationship Diagram) 및 Relational schema 로 나타내며, 해당 데이터베이스를 실제로 구현할 수 있는 SQL DDL 을 보여준다.

7.2 ER Diagram

본 시스템은 총 5 개의 대표적인 개체(Entity)로 구성된다: 유저, 유저의 모바일 디바이스, IoT 기기, Timeline(예약 정보), 프리셋.

각 개체 간의 관계를 아래와 같은 ER-Diagram 로 나타낼 수 있다. ER-Diagram 에서 직사각형은 5 개의 개체들을 나타내며, 타원은 각 개체의 속성들을 나타낸다. 복선으로 나타내어진 타원은 다중 값 속성(Multi-valued attribute)을 나타내며, Preset 개체와 Timeline 개체에는 복합 속성(Composite attribute)이 존재한다. 마름모는 서로 다른 두 개체 간 관계 정보를 나타낸다. 각 개체는 선으로 이어져 있는데, 선 위의 숫자는 관계 타입(M:N)을 세부적으로 나타내고 있다. 복선(이중 선)은 전체 참여 제약조건(Total participation constraint)을 의미한다.

7.2.1 ER Diagram

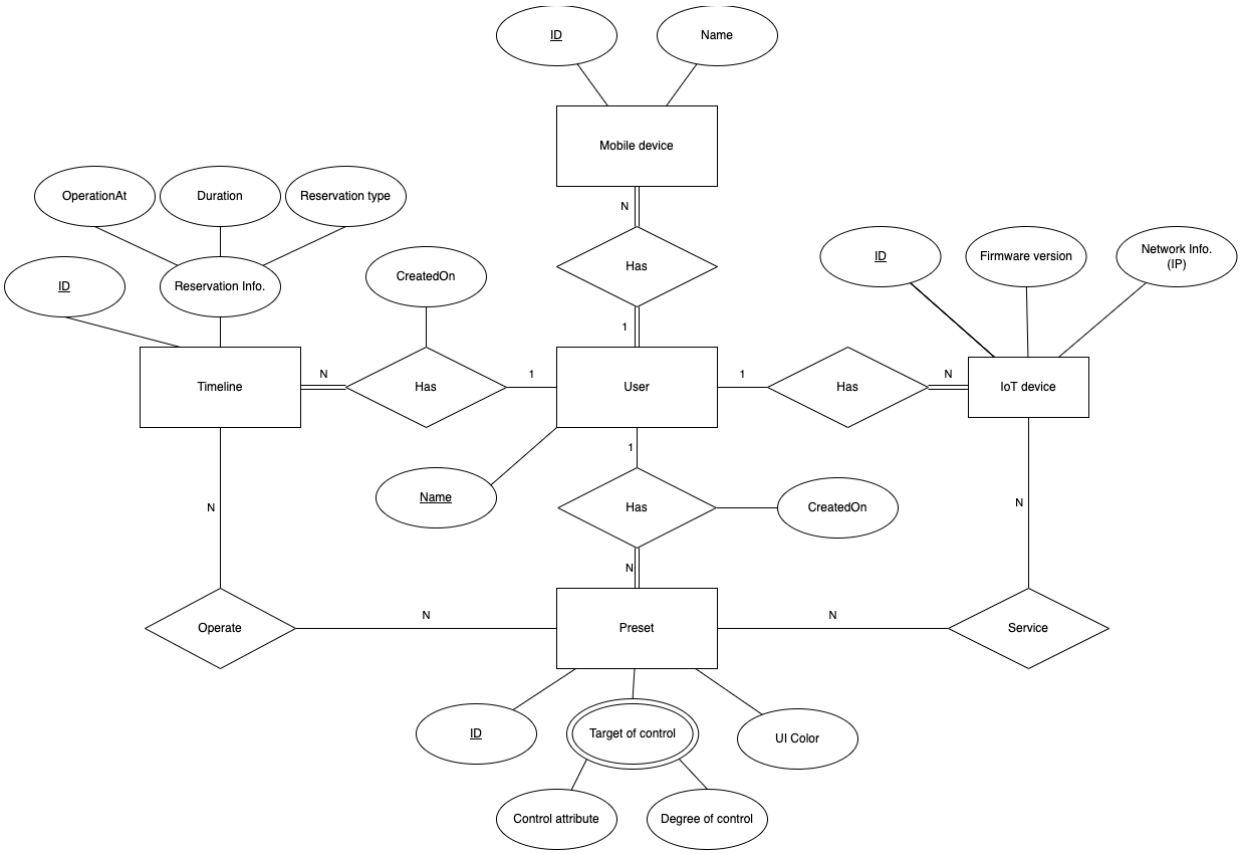

그림 31 ER Diagram

7.3 ER Diagram – Entities

7.3.1 User 개체

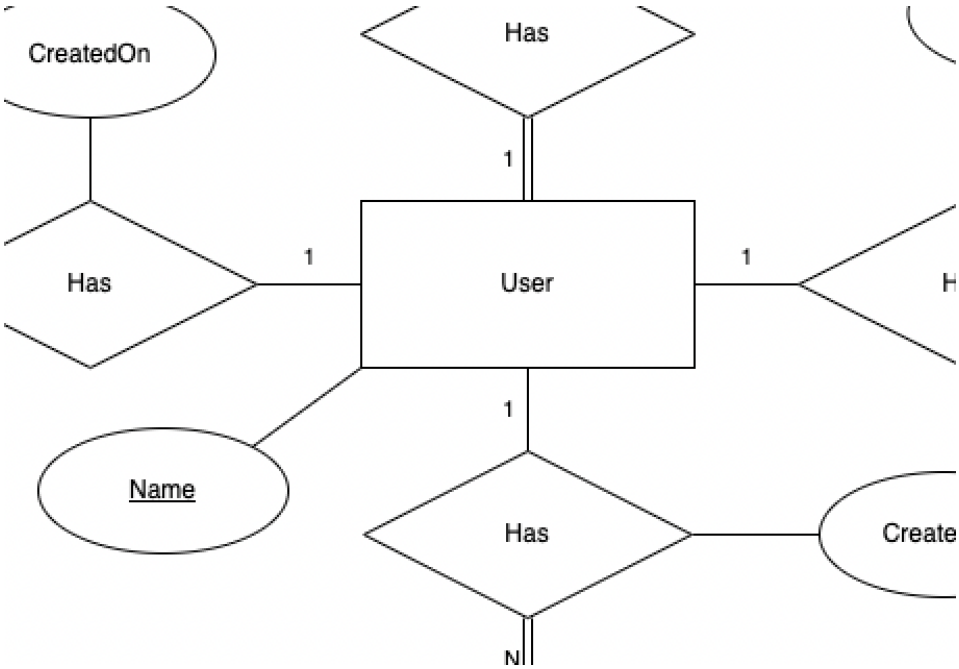

그림 32 ER Diagram - User

User 개체는 Smart Homepany 시스템의 실사용 유저의 정보를 담고있다. 독립적으로 존재하는 속성은 기본 키(Primary)로 쓰일 Name 속성 뿐이다. 따라서 실제 데이터베이스에서 User 개체의 속성은 대부분이 외래 키(Foreign key)에 해당될 것으로 예상된다. 본 소프트웨어는 기기의 고유 정보를 수집하여 이를 서버 접속 시 활용하므로 별도의 아이디 혹은 비밀번호 등의 값을 필요로 하지 않는다.

본 소프트웨어는 추후 소프트웨어 진화 단계에서 사용자의 나이 및 성별 정보를 수집하여 프리셋에 대한 추천 시스템을 구축할 수 있을 것으로 예상된다. 이와 같은 소프트웨어 진화가 동반된다면 User 개체의 속성도 변화할 수 있을 것으로 예상된다.

모든 유저는 Mobile device 와 반드시 관계를 형성하며 (total participation), IoT device, , Preset, Timeline 개체와 관계를 형성할 수 있다. (partial participation) User 개체는 다른 모든 개체와 일대다(1:N)의 관계를 형성한다.

7.3.2 Mobile device 개체

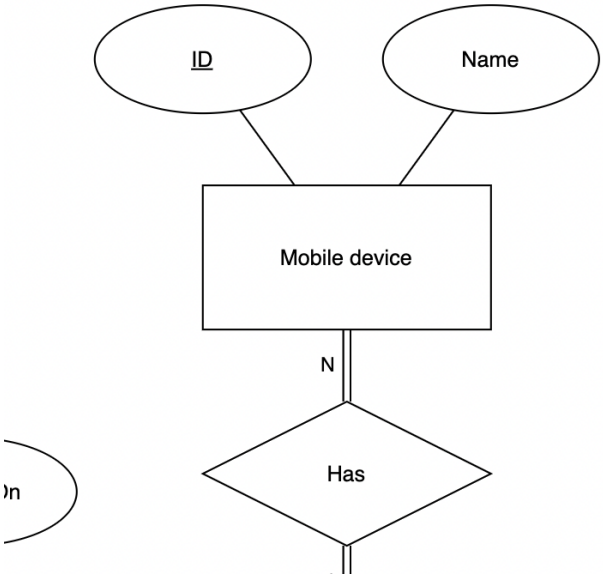

그림 33 ER Diagram - Mobile device

Mobile device 개체는 사용자의 모바일 디바이스의 아이디(ID)와 닉네임(Name)을 속성으로 가진다. 여기에서 ID 속성은 앞서 기술하였듯이 기기에 대한 고유 정보를 담고 있다. 기기에 대한 고유 정보는 사용자가 Smart Homepany 에 최초 로그인을 시도할 때 자동으로 수집 및 암호화되어 저장된다. 모든 Mobile device 개체는 반드시 하나의 User 개체와 관계를 형성한다.

7.3.3 IoT device 개체

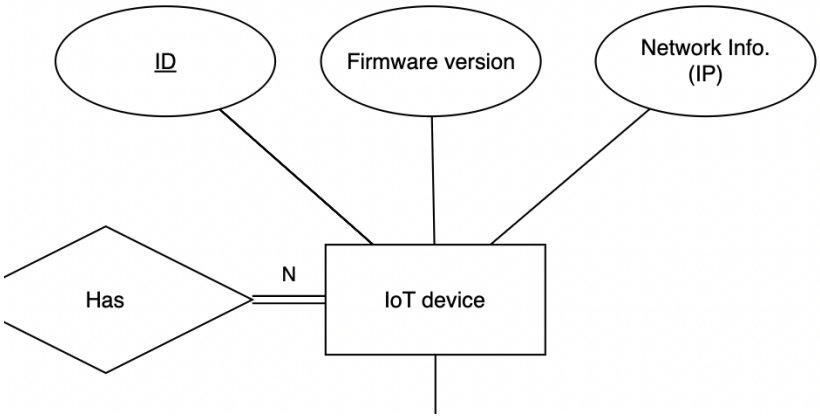

그림 34 ER Diagram - IoT device

IoT device 개체는 사용자의 IoT 기기에 대한 정보를 저장한다. 앞서 모바일 디바이스 개체처럼, IoT device 에 대한 데이터베이스 테이블에서도 사용자가 등록한 IoT 기기의 고유 정보를 수집 및 암호화하여 데이터베이스에 저장하게 된다. 또한 IoT 기기의 펌웨어 버전 업데이트 필요 여부를 주기적으로 확인하기 위해 Firmware version 속성을 가진다. 마지막으로 서버에서 기기를 제어할 때 필요할 기기의 네트워크 연결 정보(IP 정보로 잠정 예정)를 암호화하여 저장한다. 모든 IoT device 개체는 반드시 하나의 User 개체와 관계를 형성한다. 또한 모든 IoT device 는 Preset 개체와 다대다(N:N)의 관계를 형성한다.

7.3.4 Preset 개체

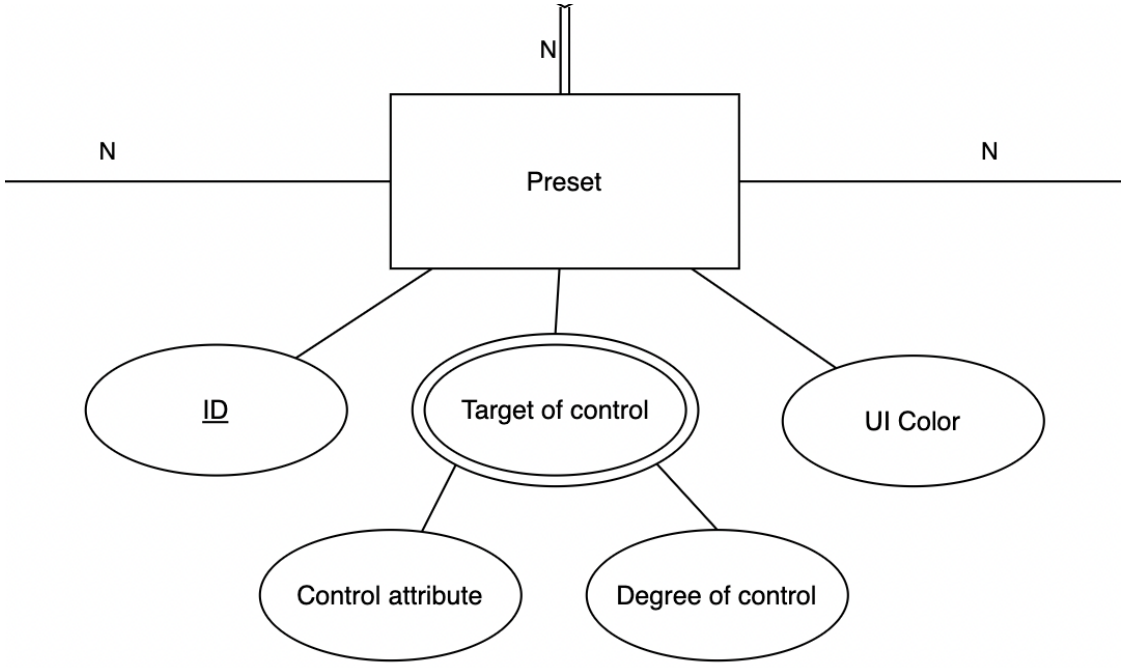

그림 35 ER Diagram - Preset

Preset 개체는 고유 ID 속성과 IoT 기기에 대한 제어 대상 정보를 담고 있다. 유저는 특정 IoT 기기에 대한 제어 대상 기능 및 제어 정도를 조정할 수 있다. 예를 들어 스마트 조명 IoT 기기를 제어하고자 한다면, 조명의 색과 명도를 각각 조절할 수 있는데 이것이 Control attribute 에 해당되며, 구체적으로 색을 파란색, 명도를 70%로 조절하고자 한다면 이 정보가 Degree of control 에 해당된다. Control attribute 속성 값은 대상 IoT 기기의 유형에 따라 그 값이 제한될 수 있으며, 이를 위해 서버단에서 추가적인 예외처리가 필요하다. 예를 들어 커튼의 경우 조명과 달리 제어 대상이 [커튼이 열린 정도] 밖에 없으므로 이 경우 Control attribute 속성은 반드시 0 의 값만 가질 수 있다. 추가적으로 UI 상에 특정 프리셋이 어떤 색으로 표시될 지에 대한 정보 또한 데이터베이스에 저장된다. (색 정보는 유저에게 제어 권한이 있으므로 유저가 색을 설정하는 대로 데이터베이스에

반영된다.) 모든 Preset 개체는 반드시 하나의 User 개체와 관계를 형성한다. 또한 모든 Preset 개체는 IoT device 개체 및 Timeline 개체와 다대다(N:N) 관계를 형성한다.

7.3.5 Timeline 개체

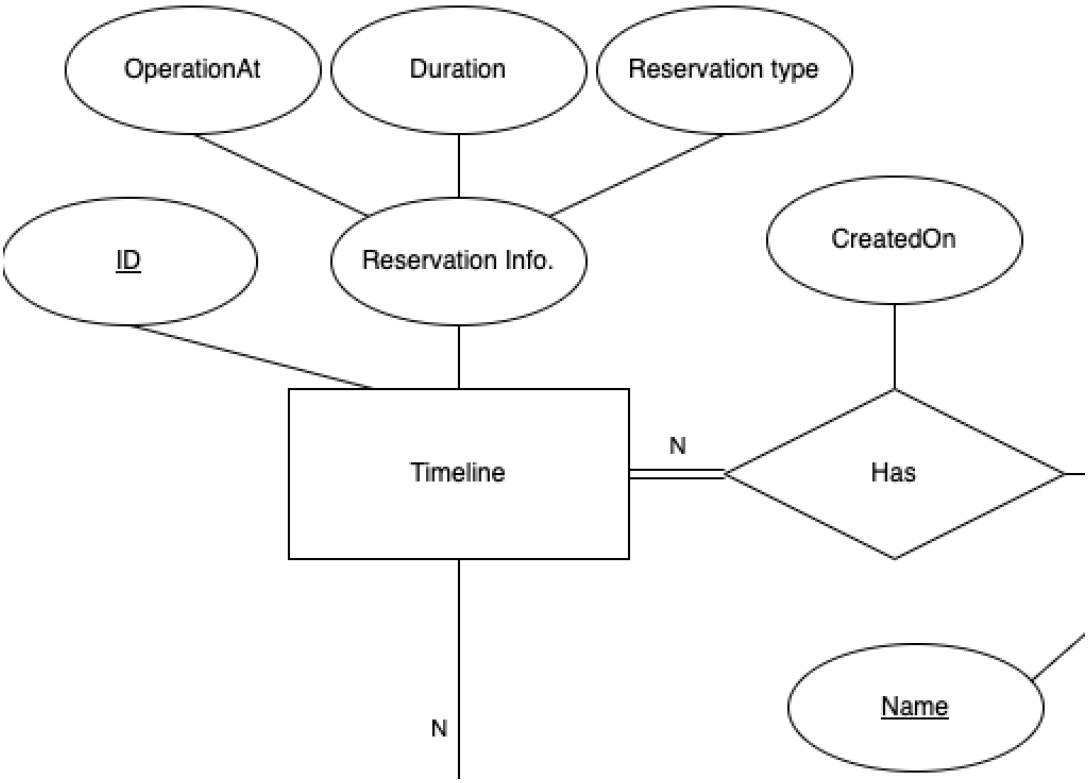

그림 36 ER Diagram - Timeline

Timeline 개체는 각 Preset 을 언제 실행하고 언제 종료할지에 대한 시간 정보를 담고 있다. 실제 서비스 단계에서는 Timeline 개체의 정보를 참조하여 특정 Preset 의 실행 여부 및 종료 여부를 실시간으로 처리한다. 특정 프리셋은 해당 프리셋과 대응되는 타임라인의 OperationAt 에 Duration 만큼 실행되며, 실행이 끝난 뒤에는 Reservation type 에 따라 OperationAt 속성의 값을 적절하게 업데이트하게 된다. Reservation type 은 Enumerate 한 특성을 가지고 현재는 잠정적으로 0 부터 3 까지의 값을 가질 수 있으며 각 값은 (‘none’, daily', 'weekly', 'monthly')를 나타낸다. 예를 들어 Reservation type 의 값이 1 일 경우 해당 타임라인은 실행을 마친 뒤 OperationAt 의 값이 다음 날 같은 시간으로 갱신된다. None 은 일회성 예약을 나타내며 실행을 마친 뒤 해당 타임라인은 소멸된다. Reservation type 이 가질 수 있는 값은 추후 소프트웨어 진화 단계에서 변경 및 추가 될 수 있다. 모든 Timeline 개체는 반드시 하나의 User 개체와 관계를 형성한다. 또한 모든 Timeline 개체는 Preset 개체와 다대다(N:N) 관계를 형성한다.

7.4 Relational Schema

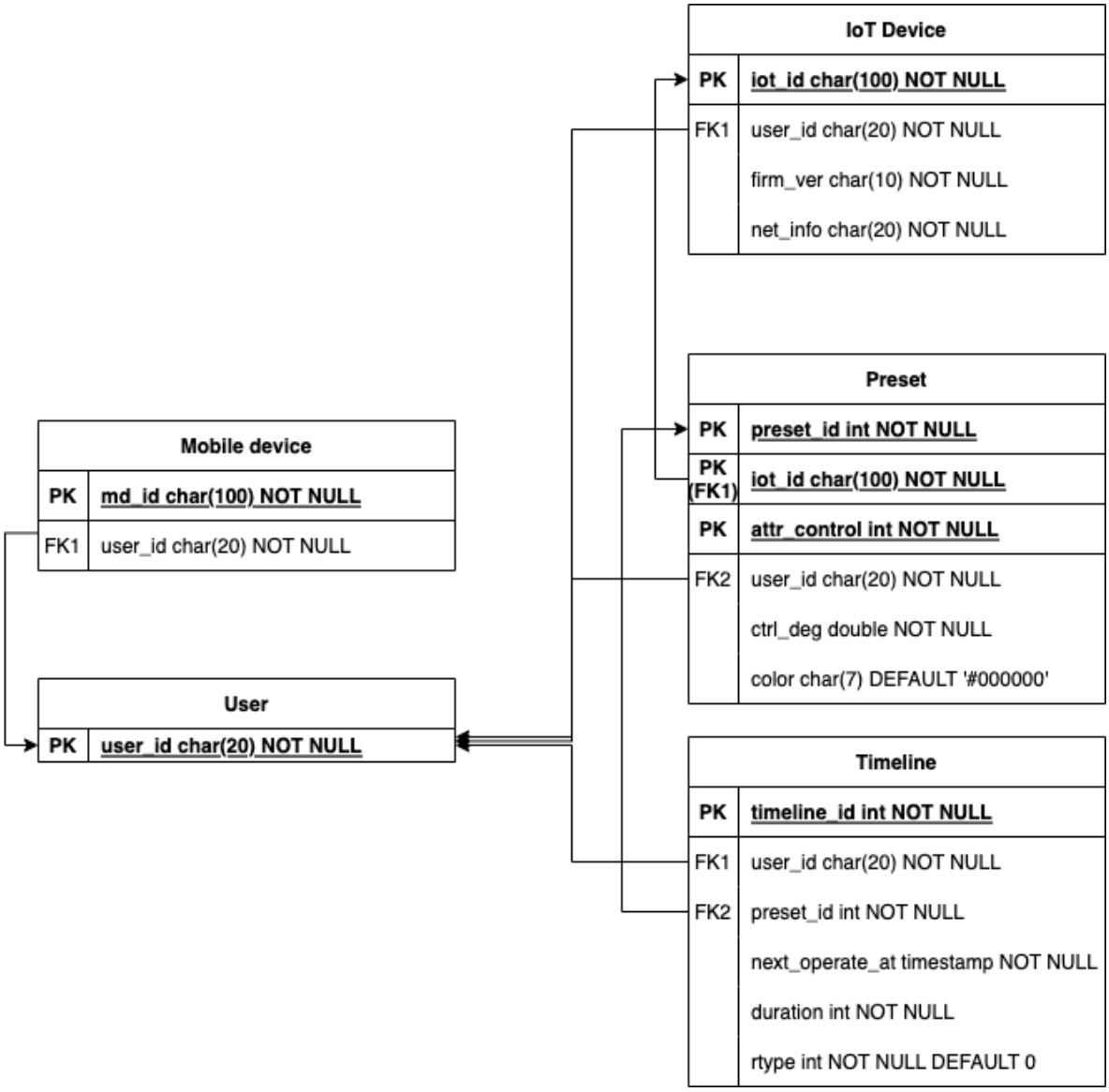

그림 37 Relational Schema

7.5 SQL DDL

7.5.1 User

```
CREATE TABLE `User` (
    `user_id` CHAR(20) NOT NULL,
    PRIMARY KEY (`user_id`)
);
```

그림 38 SQL DDL - User

7.5.2 Mobile device (Mobile_device)

```
CREATE TABLE `Mobile_device` (
    `md_id` CHAR(100) NOT NULL,
    `user_id` CHAR(20) NOT NULL,
    PRIMARY KEY (`md_id`),
    FOREIGN KEY (`user_id`) REFERENCES
User(user_id),
);
```

그림 39 SQL DDL - Mobile device

7.5.3 IoT device (Iot_device)

```
CREATE TABLE `Iot_device` (
    `iot_id` CHAR(100) NOT NULL,
    `user_id` CHAR(20) NOT NULL,
    `firm_ver` CHAR(10) NOT NULL,
    `net_info` CHAR(20) NOT NULL,
    PRIMARY KEY (`iot_id`),
    FOREIGN KEY (`user_id`) REFERENCES
User(user_id),
);
```

그림 40 SQL DDL - IoT device

7.5.4 Preset

```
CREATE TABLE `Preset` (
    `preset_id` INT NOT NULL AUTO_INCREMENT,
    `iot_id` CHAR(20) NOT NULL,
    `attr_control` INT NOT NULL,
    `user_id` CHAR(20) NOT NULL,
    `ctrl_deg` DOUBLE NOT NULL,
    `color` CHAR(7) DEFAULT `#000000`,
    PRIMARY KEY (`preset_id`, `iot_id`,
`attr_control`),
    FOREIGN KEY (`iot_id`) REFERENCES
Iot_device(iot_id),
    FOREIGN KEY (`user_id`) REFERENCES
User(user_id)
);
```

그림 41 SQL DDL - Preset

7.5.5 Timeline

```
CREATE TABLE `Timeline` (
    `timeline_id` INT NOT NULL
AUTO_INCREMENT,
    `user_id` CHAR(20) NOT NULL,
    `preset_id` INT NOT NULL,
    `next_operate_at` TIMESTAMP NOT NULL,
    `duration` INT NOT NULL,
    `rtype` INT NOT NULL DEFAULT '0',
    PRIMARY KEY (`timeline_id`),
    FOREIGN KEY (`user_id`) REFERENCES
User(user_id),
    FOREIGN KEY (`preset_id`) REFERENCES
Preset(preset_id)
);
```

그림 42 SQL DDL - Timeline

8. Testing Plan

8.1 Objectives

이 장에서는 개발, 배포, 사용자 테스트에 대한 계획에 대해 설명한다. 이 테스트들은 프로그램의 오류를 감지하고 결함을 보완하여 프로그램 완성도를 높이고 안정적인 배포를 가능하게 하는 데에 중요한 역할을 한다. 또한 프로젝트를 테스트할 때 테스트 중인 프로그램의 품질을 확인하는 데에 필요한 노력을 결정하는 것에 도움이 된다.

8.2 Testing Policy

8.2.1 Development Test

Development Test 는 개발 단계에서 이루어지는 시험으로, 프로그램 안정화를 주된 목표로 한다. 이 단계는 프로그램에 발생할 수 있는 위협과 문제사항들을 예방하고 발견하는 전략으로써, 코드가 설계 명세서의 내용을 만족하는지 테스트하기 위한 정책을 제시한다. 정적 코드 분석, 데이트 흐름 분석, 피어 코드 검토, 단위 테스팅 등의 테스트를 통해서 프로그램의 성능, 신뢰도, 보안 수준을 평가하고 상승시킨다.

8.2.1.1 Performance

Smart Homepany 가 기존의 스마트 홈 소프트웨어와 가장 큰 차이를 보이는 부분이 IoT 기기들의 상태를 프리셋을 이용하여 자동으로 조작할 수 있는 것이기 때문에, 성능 문제로 인하여 예약된 시간에 지정한 프리셋이 제대로 실행되지 않거나, 프리셋이 실행되었음에도 기기의 상태가 의도대로 변화하지 않는 상황이 발생해서는 안 된다.

IoT 기기와 정상적으로 연결되었고, 통신상태가 원활하여 서버와 문제없이 연결되었다는 전제 하에, 앱 최초 실행 과정에서 메인 화면으로의 연결은 2 초 이내에 수행되어야 한다. 메인 화면에서 각 기능에 따른 화면 전환과 데이터의 표시는 1 초 이내에 수행되어야 한다.

프리셋의 등록, 편집 및 예약은 기기에서의 수행 이후 1 초 이내의 딜레이로 서버에 적용되어야 한다. 빠른 속도로 많은 양의 프리셋을 등록하고 예약하였을 때 그 결과가 서버에 반영되고 다시 앱에 표시되기까지의 시간을 측정하여 최소화하기 위한 테스트가 있을 것이다.

프리셋에 설정해둔 기기의 옵션은 실행 이후 0.1 초 이내의 딜레이로 기기에 반영되어야 한다. 사용자가 업무 시간으로 설정된 시간대에는 즉시 인터폰 알람이 울리지 않도록 제어해야 한다. 조명,

커튼의 열린 정도, 온도 또한 지정해둔 옵션이 정확하게 반영되는지를 테스트할 것이다. 다양한 조건과 네트워크 환경에 대한 테스트 사례를 준비하고 앱과 서버와의 통신, 기기와의 상호작용에 대한 코드 흐름을 개선할 것이다.

8.2.1.2 Reliability

신뢰도 또한 프로젝트에서 평가되어야 하는 중요한 요소이다. 테스트를 하는 도중 어떠한 오작동이나 오류로 인하여 프로그램이 종료되거나 테스트 절차를 진행할 수 없는 상황을 피해야 하기 때문이다. 프로그램이 원활하게 작동하기 위해서는 이것을 구성하고 있는 서브 컴포넌트들과 장치들이 올바르게 연결되고 작동해야 한다. 따라서 development test 는 unit test 이후부터, 반복적으로, 신중하게 진행되어야 한다.

8.2.1.3 Security

사용자의 개인정보를 보호하는 것은 개발 과정에서 무조건 고려해야 하는 부분이다. 가정용 스마트 IoT 기기는 보안 취약점 점검을 하지 않는 어플리케이션을 사용하거나, 낮은 강도의 암호화를 사용하기 때문에 데이터가 외부에 노출되기 쉬운 구조이다. Smart Homepany 는 외부에서 IoT 기기로 발생할 수 있는 침입을 고려하여 불필요한 사용자 정보를 저장하지 않고, 저장한 정보는 암호화하여 정보 유출에 대비해야 할 필요가 있다.

8.2.2 Release Test

Release Test 는 개발 팀 외부에서 사용할 수 있도록 설계된 프로그램의 배포 버전을 테스트하는 프로세스이다. 이 단계의 주된 목표는 프로그램의 배포가 진행되어 고객에게 전달된 프로그램이 계획했던 대로 사용하기에 충분한지를 테스트하는 것이다. 테스트는 일반적으로 소프트웨어의 정식 배포 전에 실행되어야 하며, 기본적인 구현이 완료되면 첫 번째 버전부터 시작되어야 한다. 이를 알파 테스트라고 한다. 알파 테스트 단계에서 사용자의 반응을 살펴보고 얻은 결과에 따라 프로그램의 약점이라고 파악된 부분을 수정하여 베타 버전에 반영한다. 이렇게 새롭게 배포된 베타 버전의 테스트에서는 실제 사용자들과 함께 테스트를 진행하게 되고, 베타 테스터들의 의견을 수렴하여 정리한 수정 사항을 새로운 버전에 반영할 수 있다.

8.2.3 User Test

사용자 테스트를 진행하기 위해서 실제로 사용자가 처할 것으로 예상되는 시나리오와 현실적인 상황을 설정할 필요가 있다. 자택에 IoT 기능을 제공하는 인터폰, 조명, 커튼, 보일러를 비롯한 다양한 기기들이 설치되어 있는 사용자를 대상으로 테스트를 진행할 수 있다. 사용자들에게 베타 버전을 배포하고 리뷰를 수집하여 Smart Homepany 개선에 이용할 수 있다.

8.2.4 Test Case

Test case 는 성능, 신뢰성 및 보안의 3 가지 기본 측면에 따라 설정되어야 한다. 프로그램이 기획했던 의도대로 사용할 수 있는지, 사용하는 과정에서 발생할 수 있는 예기치 못한 오류는 무엇이 있는지, 실제 환경에서 소프트웨어를 사용할 때 생기는 문제점에는 무엇이 있는지를 파악하여 test case 를 설정하고 테스트를 진행해야 한다.

9. Development Plan

9.1 Objectives

이 장에서는 시스템 개발에 사용된 개발 환경과 도구, 그리고 시스템에 대한 제약조건, 가정 및 의존성을 기술한다.

9.2 Frontend Environment

9.2.1 Flutter

그림 43 Flutter 로고

Flutter 는 구글이 출시한 오픈 소스 크로스 플랫폼 GUI 기반 어플리케이션 프레임워크로 Android, iOS, Windows, Linux 및 웹 브라우저용 소프트웨어를 개발할 수 있다. 본 프로젝트에서는 Android, iOS 용 Smart Homepany 사용자 어플리케이션을 개발하는 데 사용되었다.

9.2.2 Android Studio

그림 44 Android Studio 로고

Android Studio 는 Android 어플리케이션 개발을 위한 공식 통합 개발 환경이다. 본 프로젝트에서는 Android 용 Smart Homepany 사용자 어플리케이션을 개발하는 데 사용되었다.

9.2.3 Xcode

그림 45 XCode 로고

Xcode 는 Apple 이 개발한 macOS, iOS 용 소프트웨어 개발을 위한 통합 개발 환경이다. 본 프로젝트에서는 iOS 용 Smart Homepany 사용자 어플리케이션을 개발하는 데 사용되었다.

9.3 Backend Environment

9.3.1 Amazon EC2

그림 46 Amazon EC2 로고

Amazon Elastic Compute Cloud(Amazon EC2)는 Amazon Web Services(AWS) 클라우드에 기반하여 가상 컴퓨팅 파워를 통해 서버를 구축할 수 있도록 하는 서비스이다. 본 프로젝트에서는 Smart Homepany 서버를 구축하는 데 사용되었다.

9.3.2 Node.js

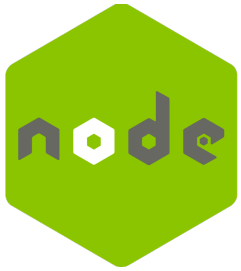

그림 47 Node.js 로고

Node.js 는 브라우저 기반 서버 사이드 네트워크 어플리케이션 개발을 위한 소프트웨어 플랫폼이다. 본 프로젝트에서는 Smart Homepany 서버 어플리케이션을 개발하는 데 사용되었다.

9.3.3 Amazon RDS

그림 48 Amazon RDS 로고

Amazon Relational Database Service(Amazon RDS)는 AWS 클라우드에서 관계형 데이터베이스를 구축할 수 있도록 하는 서비스이다. 본 프로젝트에서는 Smart Homepany 서버의 데이터베이스로 사용되었다.

9.3.4 SQLite

그림 49 SQLite 로고

SQLite 는 어플리케이션에 내장하여 사용하는, C 기반의 가벼운 데이터베이스 관리 시스템이다. 본 프로젝트에서는 Smart Homepany 서버 데이터베이스의 DBMS 로 사용되었다.

9.4 Constraints

Smart Homepany 는 본 디자인 명세서에 기반하여 구현된다. 이외의 부분은 아래의 제약사항을 따라야 하며, 제약사항을 만족한다면 나머지는 개발자인 팀 1 에서 선호하는 방향으로 구현될 수 있다.

- 요구를 만족하는 오픈소스 소프트웨어가 있다면 적극 활용한다.
- 용이한 유지보수를 위해 소스코드는 알기 쉽게 작성한다.
- 어플리케이션의 용량은 가능한 한 작게 한다.
- 어플리케이션 성능을 최적화하는 방향으로 개발한다.
- 직관적인 UI 를 제공하여 사용이 쉽도록 한다.
- Smart Homepany 가 Android 10 Queen Cake (API 29) 및 iOS 13.0 이상의 운영체제에서 오류 없이 동작하도록 한다.
- 1GB RAM 용량과 1.2GHz 의 CPU 클럭 속도를 지원하는 디바이스와 2.4GHz 의 Wi-Fi 네트워크 환경에서 IoT 기기는 3 초 이내에 사용자의 입력에 대응되는 액션을 수행한다.
- 데이터 수집에 대한 사용자의 동의 하에서만 사용자의 프리셋 및 모드 설정, 타임테이블에 관한 데이터를 수집할 수 있다.

9.5 Assumptions and Dependencies

Smart Homepany 는 1GB 의 RAM 용량 및 1.2GHz 의 CPU 클럭 속도를 지원하는 하드웨어와 Android 10 Queen Cake (API 29) 혹은 iOS 13.0 운영체제를 최소 사양으로 요구하며, 2GB 이상의 RAM 용량 및 1.6GHz 이상의 CPU 클럭 속도를 지원하는 하드웨어와 Android 12 Snow Cone (API 31) 이상 혹은 iOS 15.0 이상의 운영체제를 권장 사양으로 요구한다. 그리고 모든 IoT 기기는 Smart Homepany 에서 지원하는 규격의 모델이어야 하며 2.4GHz 대역의 Wi-Fi 네트워크에서만 동작하므로 2.4GHz 와 5GHz 를 모두 지원하는 네트워크를 사용하는 경우 사용자는 2.4GHz 를 선택해야 한다.

사용자는 업무 알람 제어 및 주거 알람 제어 기능을 온전히 활용하기 위해 본인의 업무 관련 어플리케이션과 자택 내 인터폰 등을 제어 대상으로 지정해야 한다. 또한 Smart Homepany 는 사용자의 디바이스에 Bluetooth 및 위치 정보, 푸쉬 및 팝업 알람에 대한 권한을 요구할 수 있으며 사용자는 이에 대해 권한을 허용 여부를 선택할 수 있으나, 기능의 완전한 사용을 위해 기본적으로 모든 요구 권한을 허용하는 것을 권장한다. 상기 사항들이 지켜지지 않을 경우 Smart Homepany 는 개발자 및 사용자가 의도한 대로 동작하지 않을 수 있으며, 서비스 이용에 차질이 있을 수 있다.

10. Supporting Information

10.1 Software Design Specification

본 소프트웨어 설계 명세서 (Software Design Specification, SDS)는 IEEE Recommendation (IEEE Recommended Practice for Software Design Descriptions, IEEE-Std-1016-2009)에 의거하여 작성되었다.

10.2 Document History

Date	Version	Description	Writer
2022/05/03	0.0.0	문서 작업 시작	전원
2022/05/06	0.0.1	7	김종헌
2022/05/06	0.1.0	4.1-4.2.4	임세창
2022/05/07	0.1.1	4.2.5-4.2.10	홍권
2022/05/07	0.2.0	6, 7	김종헌
2022/05/07	0.3.0	5	오세훈
2022/05/08	0.3.1	6.6-6.7	오세훈
2022/05/08	0.3.2	6.4-6.5	김종헌
2022/05/08	0.3.3	4.2.5-4.2.10	홍권
2022/05/09	0.3.4	4.2.5-4.2.10	홍권
2022/05/09	0.4.0	1, 9, 10	김민현

2022/05/11	1.2.0	문서 수합 및 서식 통일	이미소
2022/05/13	2.0.0	오타 점검, 어체 정리	전원
2022/05/15	2.0.1	최종 검토 및 수정	전원