

반려동물 원격 케어를 위한 스마트홈 시스템

Smart Home Pet Care



차현묵, 곽재원, 김준식, 양승찬, 유새하, 이경돈, 최하늘

개요

시스템 구조

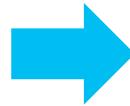
SRS

SDS

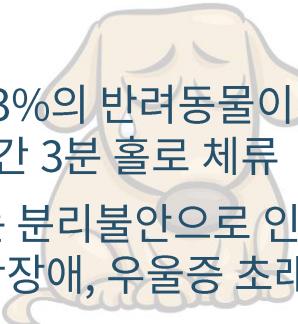
1. 개요

반려 동물의 분리불안

- 반려동물의 혼자 있는 시간 증가
- 혼자 있는 반려 동물의 분리불안

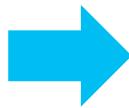
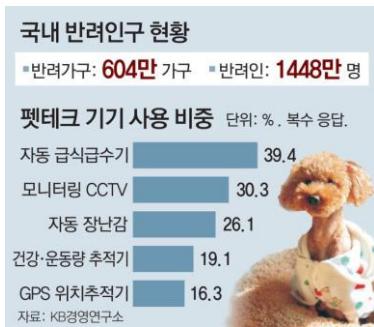


- 84.3%의 반려동물이 평균 6시간 3분 홀로 체류
- 이는 분리불안으로 인한 공황장애, 우울증 초래



펫케어 시장 가능성

- 반려동물 보유 가구 증가
- 펫케어 시장 규모 증가



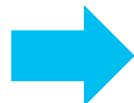
- 높은 시장 성장률
- 기술의 지속적인 발전
- 시장의 지속적인 확대

국내외 상용 서비스

- LG유플러스 스마트홈 펫케어 서비스
- 펫맘 스마트 정수기
- 놀아주는 스마트 펫카메라 러붐 T20
- AI 급식기 스마트 사료 그릇 Mookki

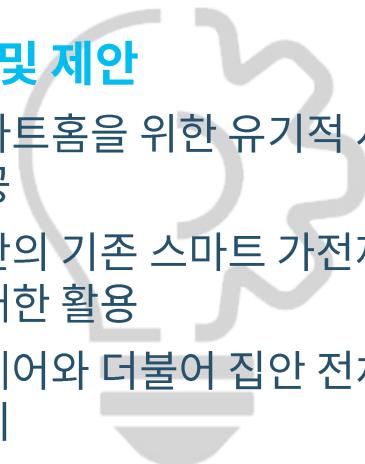
문제점

- 주력 기능에만 집중된 제품
- 구성전용 스마트 기기 필요



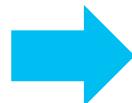
개선 및 제안

- 스마트홈을 위한 유기적 시스템 제공
- 집안의 기존 스마트 가전제품을 최대한 활용
- 펫케어와 더불어 집안 전체를 관리



반려동물 돌봄 편의성 증대

- 반려동물과 가정의 상태를 실시간으로 확인
- 상황에 맞는 동작 자동 수행 또는 원격으로 명령 지시
- IoT 기술을 이용해 집안의 환경을 유기적으로 조정

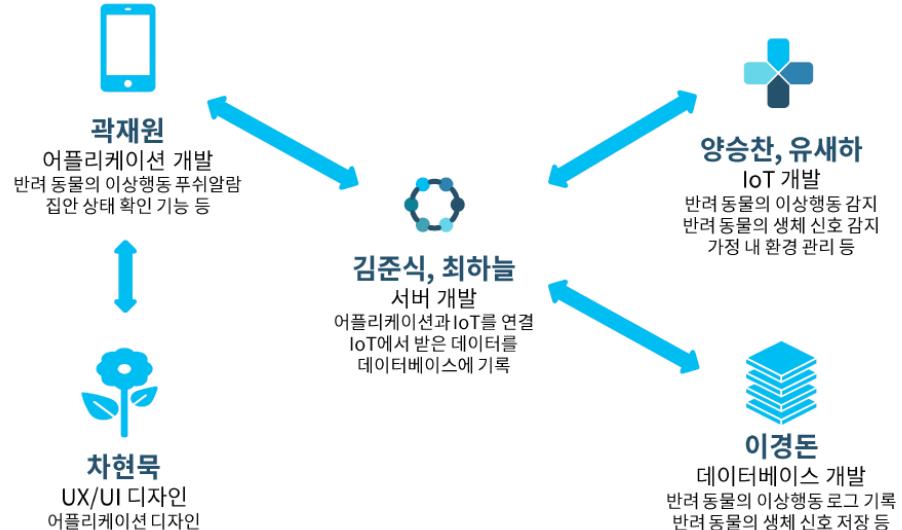


“반려동물과 반려인 모두가
만족할 수 있는
최선의 환경을 제공하는
서비스 구축”

펫케어 산업시장 진출/확대

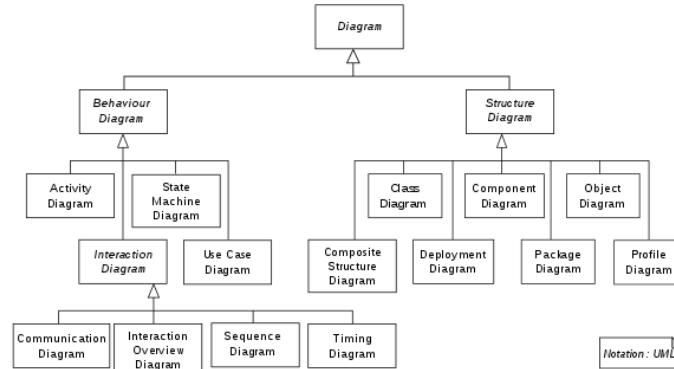
반려동물 방치 문제 해결

스마트 헬스케어 도입



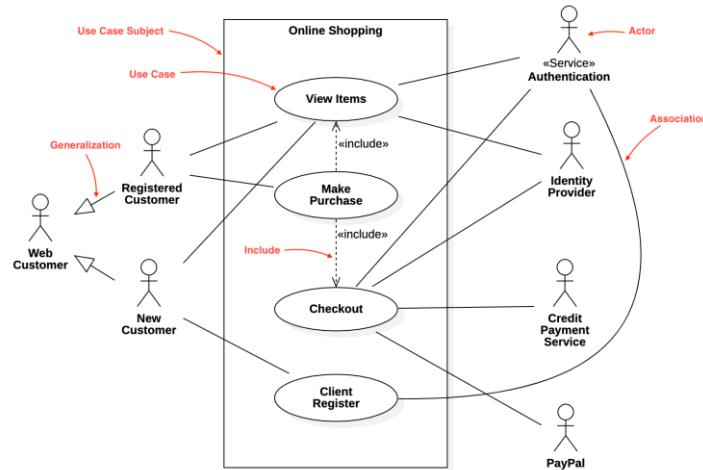
Unified Modeling Language

- 개발자들 간의 의사소통을 용이하게 하기 위해 사용되는 표준화된 모델링 언어
- 시스템의 구조를 모델링하는 과정에서 빠진 부분이나 상충된 부분을 찾기 쉬움
- 프로젝트의 규모와 상관없이 적용 가능



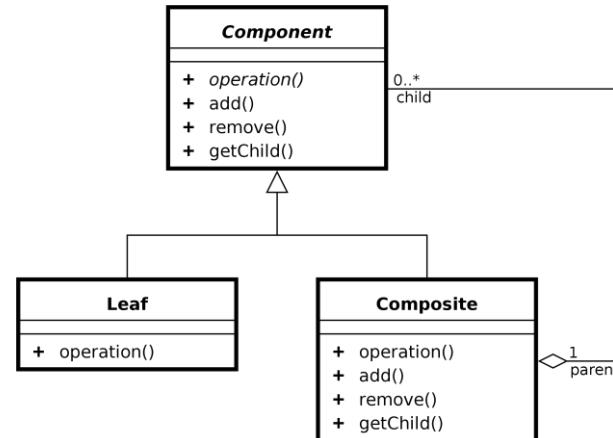
Use Case Diagram

- 시스템이 제공하는 기능적 단위 묘사
- 기능적 요구사항 시각화
- 구성요소
 - Actor : 시스템의 사용자
 - Scope : 시스템의 주요 기능
 - Use Case : 사용자가 실제 시스템을 사용할 때 맞이하는 상황



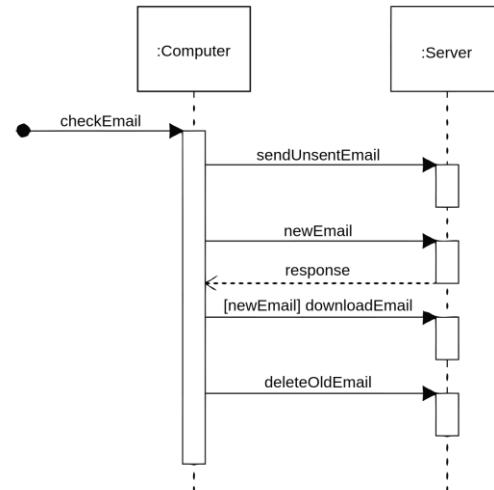
Class Diagram

- 시스템 내에서 객체가 다른 객체와 이루는 관계를 보여줌
- 시스템의 정적인 구조를 보여줌
- 구성요소
 - 1행 : class의 이름
 - 2행 : class의 attribute
 - 3행 : class의 method



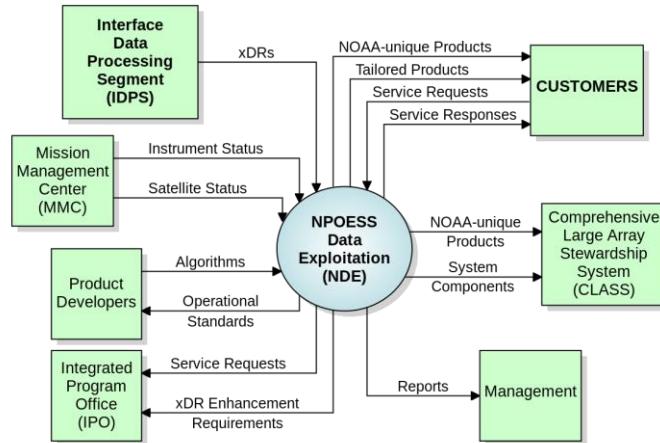
Sequence Diagram

- 구체적인 흐름 묘사
- 객체 간 관계, 다른 객체 호출 과정 구체적 묘사
- 구성요소
 - 세로 축 : 시간 순 확인 가능
 - 가로 축 : 호출/전송 객체 확인 가능
 - 객체 : Diagram의 맨 위쪽에 박스로 묘사



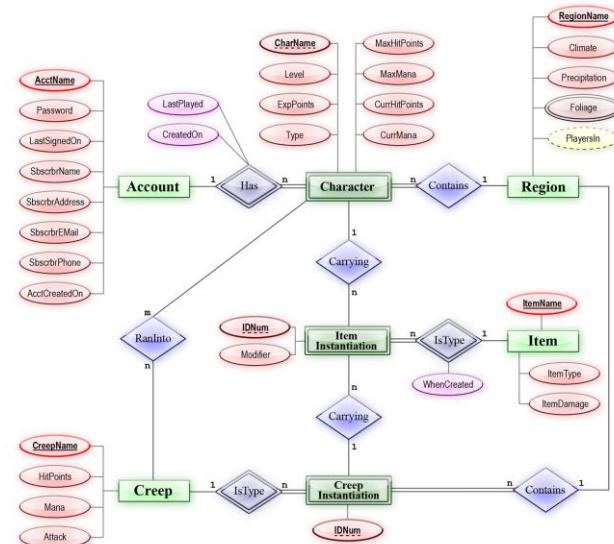
Context Diagram

- 시스템의 경계, 환경, 상호작용 객체 묘사
- 시스템을 표현하는 가장 높은 Level의 Diagram
- 시스템의 정적인 구조 묘사
- 시스템이 고려해야 할 외부적인 요인에 초점
- 모든 이해관계자가 읽어야 하므로, 이해하기 쉬운 언어 사용



Entity-Relationship Diagram

- 각 객체를 이루고 있는 Attribute와 객체관의 관계 묘사
- 구성요소
 - 객체 : 직사각형
 - 객체의 Attribute : 타원형
 - 공유 Attribute : 실선 연결
 - 객체 간의 관계 : 마름모, 화살표

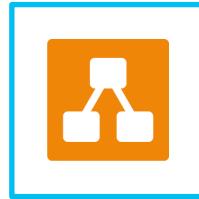




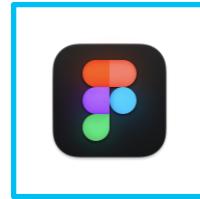
Microsoft Word



Google Docs

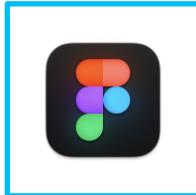


Draw.io

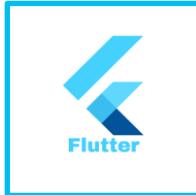


Figma

Frontend



Figma

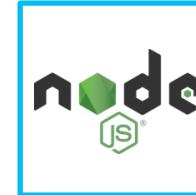


Flutter

Backend



AWS



Node.js



MariaDB

개발 계획표

18

2. 시스템 구조

System
Interface

User
Interface

HW
Interface

SW
Interface

Communication
Interface

Memory
Constraints

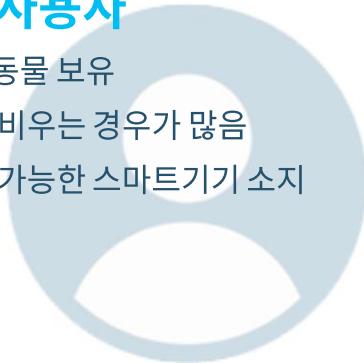
인터페이스 설계

21

System Interface	User Interface	HW Interface
<ul style="list-style-type: none">모든 정보는 기본적으로 서버에 암호화 후 저장IoT 디바이스는 Wi-Fi를 통해 실시간으로 동기화 	<ul style="list-style-type: none">UI는 모바일 기기의 화면을 통해 출력조작은 기기 내장 터치 기능 이용계정 로그인 필요 	<ul style="list-style-type: none">Android 및 iOS 기반 모바일 기기를 대상으로 서비스최소 1.5GB 이상의 RAM1.4GHz 이상의 AP 
<ul style="list-style-type: none">Android 9 Pie 또는 iOS 14이상 	<ul style="list-style-type: none">HTTP 프로토콜을 이용해 통신수치 데이터 : JSON영상 데이터 : MPEG (via HTTP Live Streaming) 	<ul style="list-style-type: none">지속적인 백그라운드 작업최소 3GB 이상의 RAM 용량최소 500MB 이상의 내부 스토리지 용량 
SW Interface	Communication Interface	Memory Constraints

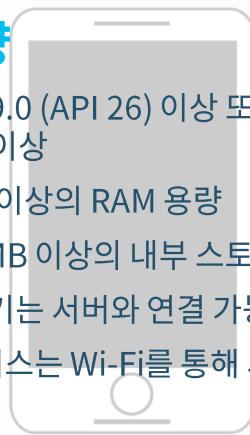
타깃 사용자

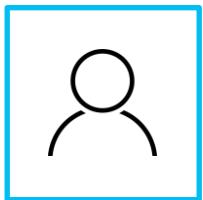
- 반려동물 보유
- 집을 비우는 경우가 많음
- 휴대 가능한 스마트기기 소지



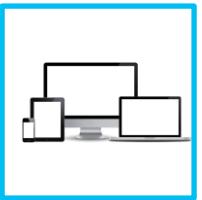
기기 사양

- Android 9.0 (API 26) 이상 또는 iOS 14.0 이상
- 최소 3GB 이상의 RAM 용량
- 최소 500MB 이상의 내부 스토리지 용량
- 스마트 기기는 서버와 연결 가능
- IoT 디바이스는 Wi-Fi를 통해 서버와 연결 가능





계정 관리



IoT
디바이스
등록/설정



이상행동
PUSH 알림



실시간 위치
파악



행동 로그



건강 상태
확인/분석

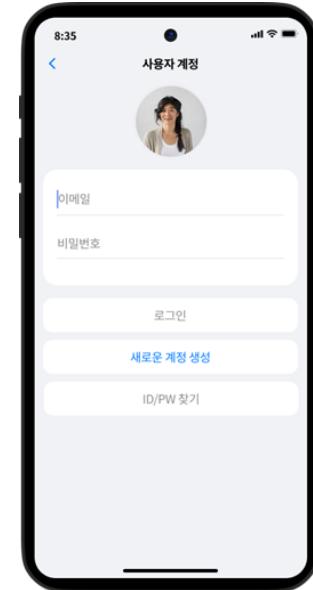


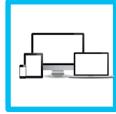
설정



계정 관리

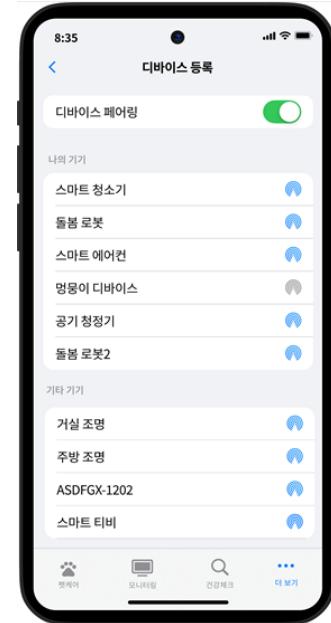
- 앱 실행 후 맨 처음 보는 화면
- 기존 이용자의 경우:
DB 서버에 등록되어 있는 본인의 계정으로 로그인
- 신규 이용자의 경우:
계정 생성 후 로그인
- ID/비밀번호 분실 시 찾는 기능 제공





IoT 디바이스 등록 및 설정

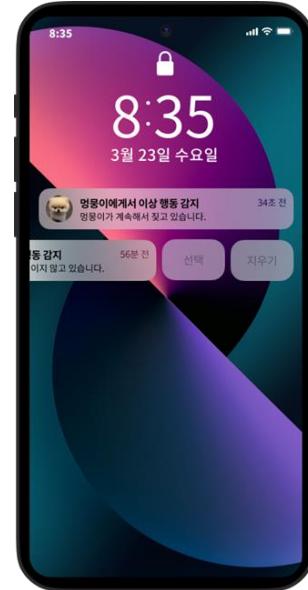
- 신규 IoT 디바이스의 연결 수행
- 연결된 IoT 디바이스의 관리
- 사용중인 IoT 디바이스의 연결 해제
- 사용중인 IoT 디바이스의 기능 설정





반려동물 이상행동 PUSH 알림

- IoT 디바이스의 카메라를 이용한 영상 분석
- AI로 이상행동이 감지되면 사용자에게 즉시 PUSH 알림 전송
- 이상행동
 - 일정 시간 동안 특이 행동을 반복
 - 일정 시간 움직임이 없음





반려동물 실시간 위치 파악

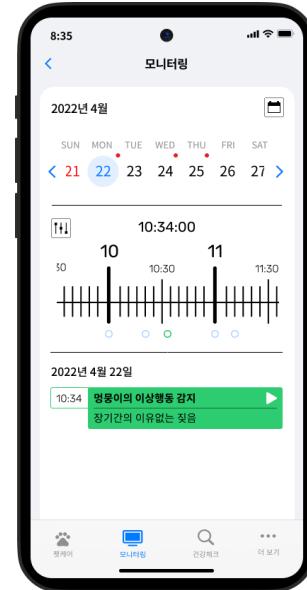
- IoT 디바이스를 이용해 반려동물 위치 파악
- 어플리케이션 화면에 지도 형태로 표시
- 설정해둔 위치를 벗어날 경우, PUSH 알림 전송





반려동물 행동 로그

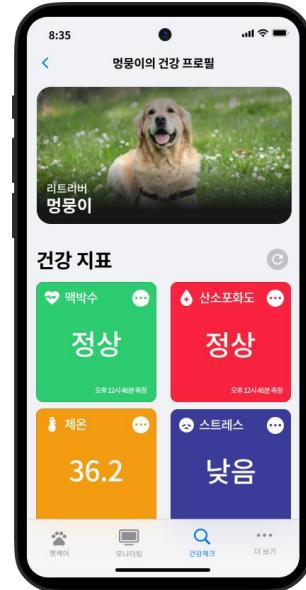
- 미처 확인하지 못한 시간대의 반려동물 행동 확인 가능
- 저장 내용 : 1개월간 보관
 - 반려동물 이상행동
 - 위치
 - 맞춤 케어 시스템의 장비 작동 여부
 - 발생 시간
 - 발생 위치
 - 1분 이내의 영상





반려동물 건강상태 획득/분석

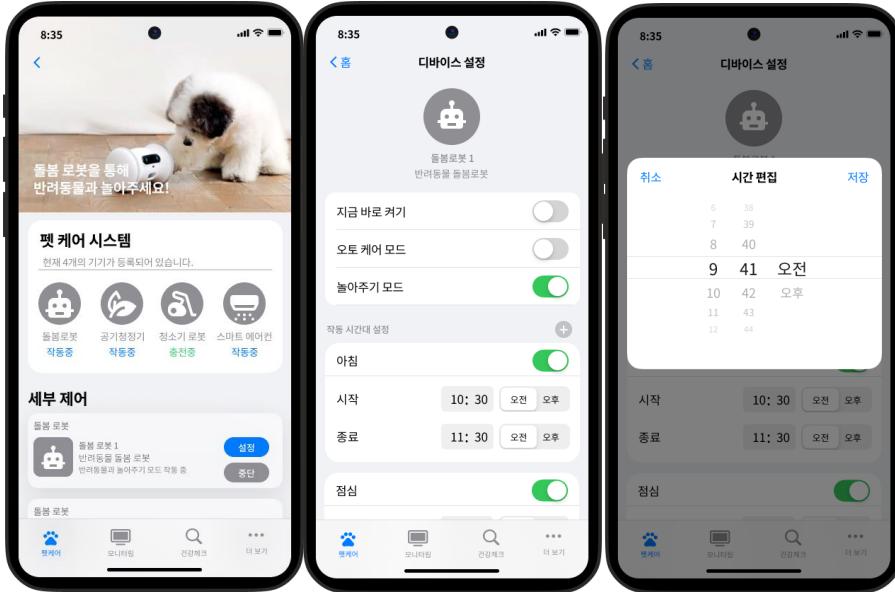
- 스마트 IoT 디바이스를 통한 반려동물 건강 체크
- 사용자에게 수치화 된 데이터와 해석 결과로 제공
- 특정 지표에서 비정상적 수치 확인 시, 필요한 조치사항 제공
- 맥박 수, 산소포화도, 체온, 스트레스 수치 등





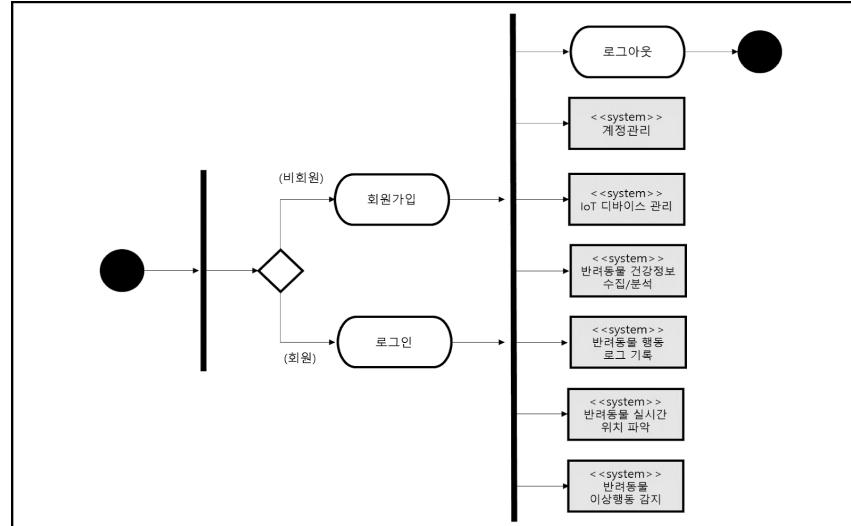
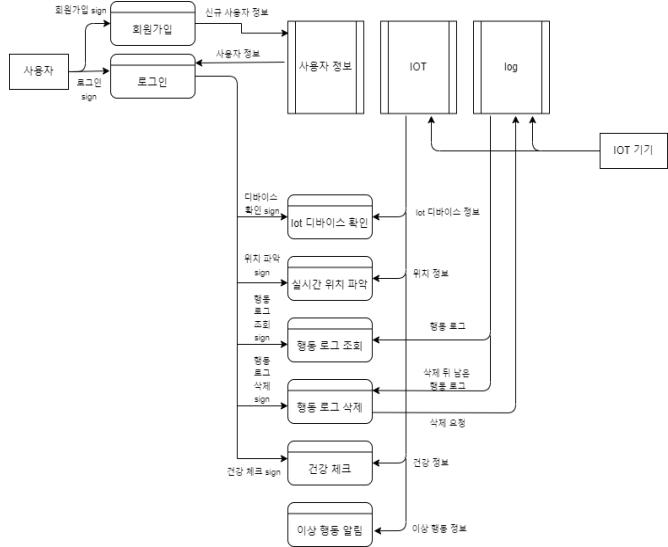
반려동물 맞춤 케어 시스템 설정

- 등록된 IoT 디바이스를 조작해 원격 케어
- 먹이 공급 시스템, 청소 로봇, 놀이 로봇, 온도 관리 시스템, 공기 관리 시스템 사용 가능
- 기기의 작동 시간 직접 설정 또는 자동 설정 가능
- 기기의 상태 여부 확인 가능



3. SRS

전반적인 구조



Maria DB Server

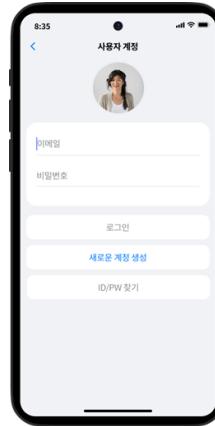
- 사용자 계정, 로그 등 정보 저장 DB
- JSON, MP4 포맷의 packet 교환
- Query Statement 사용



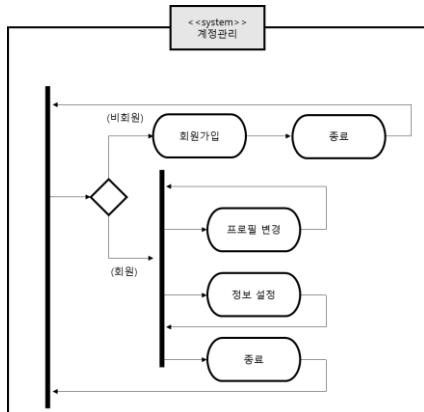
개요

- 로그인 시 String type의 user_id와 user_pw를 입력, 승인 시 메인 페이지 이동
- 위 입력값들을 패킷화하여 DB 서버에 전송
- 등록되어 있는 유저 데이터와 대조 후 알맞은 응답 메시지 팝업
- 신규 이용자일 경우 회원가입, 기존 이용자의 경우 ID/PW 찾기 가능

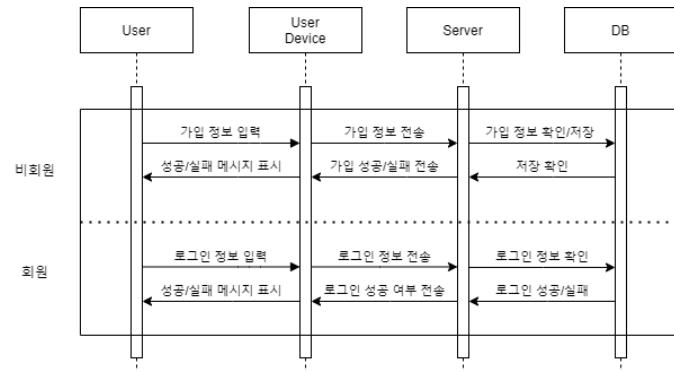
UI



- Process Model



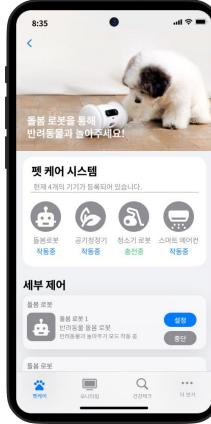
- Sequence Diagram



개요

- 등록된 IoT기기 상태 확인
- 비동기적 프로그래밍을 통해 시스템은 사용자 요청에 즉각 응답
- 설정 버튼을 통해 IoT기기 설정 페이지 이동
- 동작/중지 버튼을 이용하여 IoT기기 통제

UI



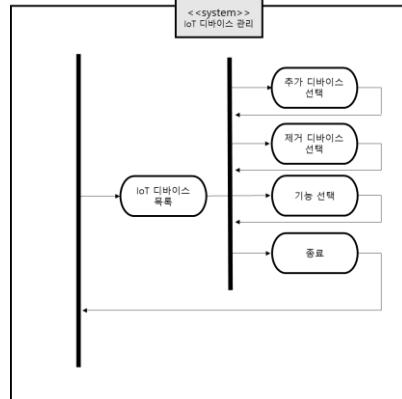
개요

- Wi-Fi AP를 이용한 디바이스 등록
- 상단에 현재 연결되어 있거나, 이전의 연결로 인해 등록되어 있는 기기 표시
- 하단에 추적 가능한 기기들 표시

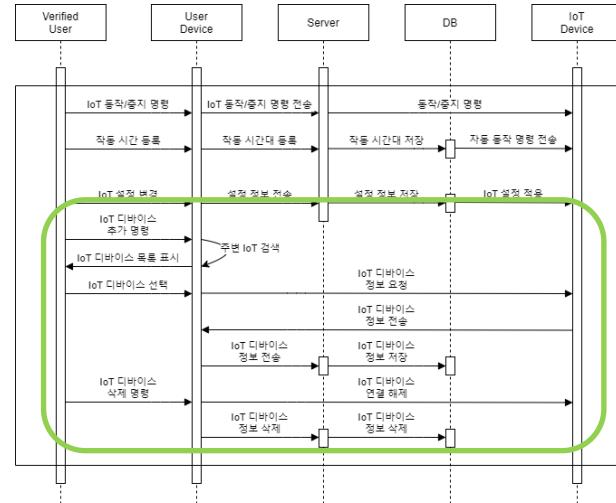
UI



- Process Model



- Sequence Diagram



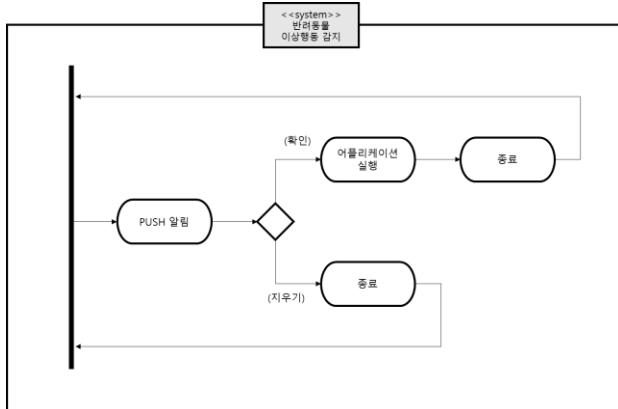
개요

- 실시간 반려동물 촬영영상 분석 후 이상행동 감지 시 알림
- 어플리케이션 push 알림의 형태로 수신
- 제목과 내용, 수신시각은 Text type, 배지 아이콘은 Image type으로 구성

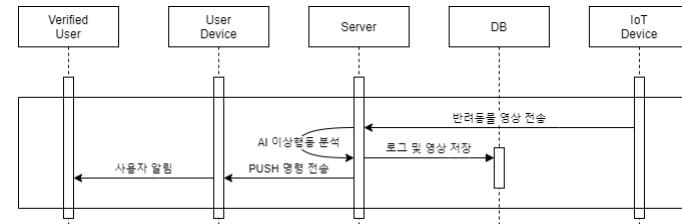
UI



- Process Model



- Sequence Diagram



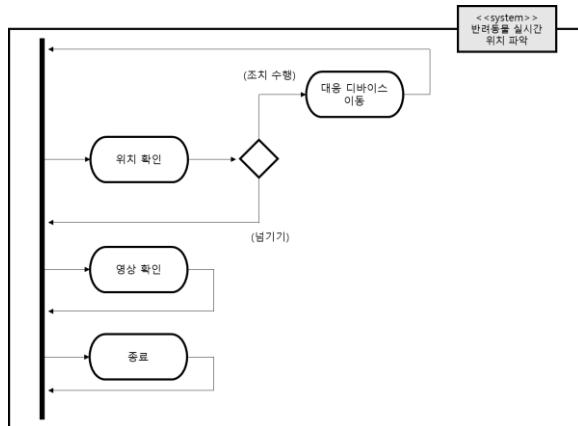
개요

- 인터페이스 프로토콜 - NMEA
- 보고싶은 위치 지도에서 선택, 해당 위치로 카메라가 탑재된 디바이스 이동 후 촬영
- 연결된 GPS에서 NMEA Sentence 수신 후 JSON String으로 변환, 서버에 전송

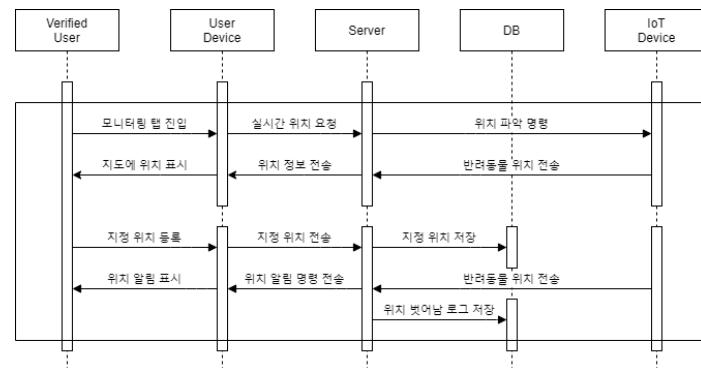
UI



- Process Model



- Sequence Diagram



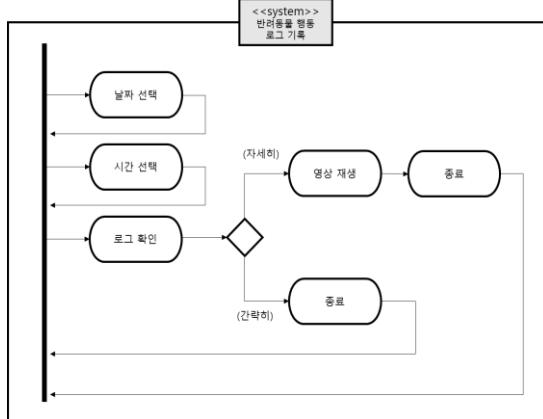
개요

- AI의 이상행동 탐지/IoT 디바이스의 반려동물 감지 이벤트를 패킷화하여 서버로 전송
- 서버에 로그 저장, 1개월간 보관
- 날짜, 시간, 내용 3부분 구성
- 영상의 경우 MP4 형식으로 재생 가능

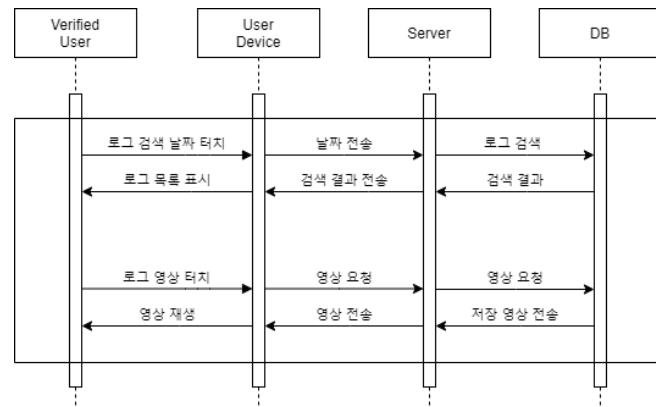
UI



- Process Model



- Sequence Diagram



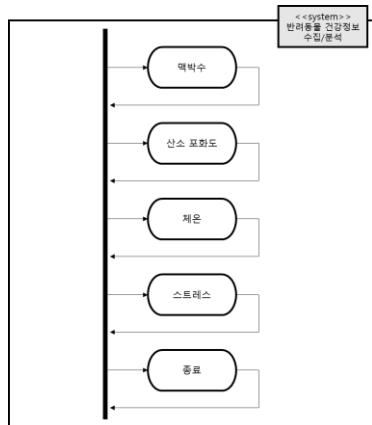
개요

- 애완동물에 부착된 웨어러블 디바이스를 통한 건강 지표, 분석 결과 DB 서버에 저장
- 새로고침 버튼을 이용한 실시간 정보 최신화 가능
- 지표별로 더 보기(….)버튼을 통해 보다 자세한 정보, 기록 파악 가능

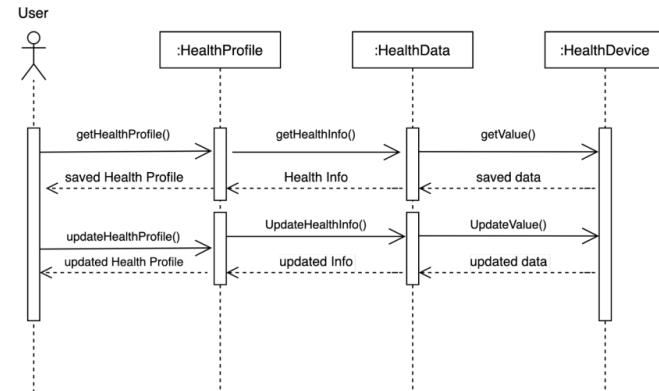
UI



- Process Model



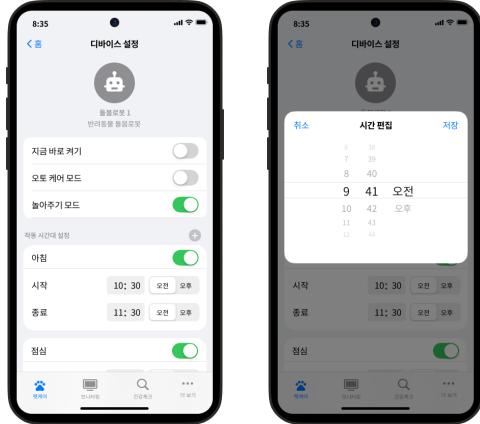
- Sequence Diagram



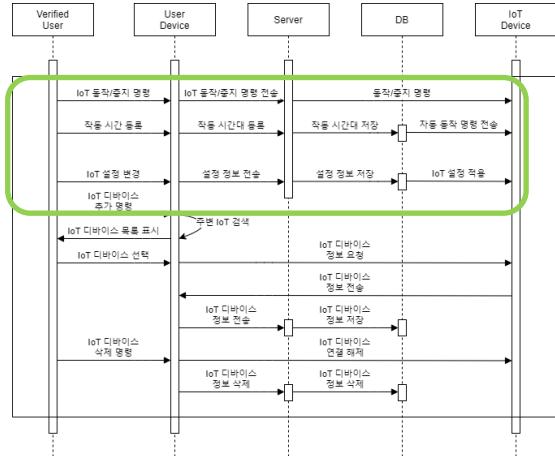
개요

- 등록된 IoT기기 설정 조절
- Auto-Care 기능을 통한 기기 작동 자동화
- 작동 시간대 구체적인 설정 가능
- 설정값들은 기본적으로 DB 서버에 저장
- 변경사항 적용시 서버에 변경내용 전송

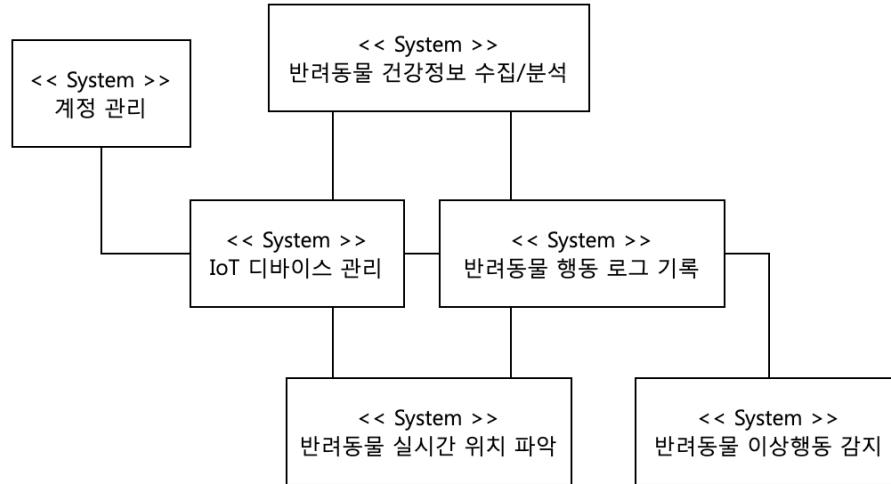
UI



- Sequence Diagram

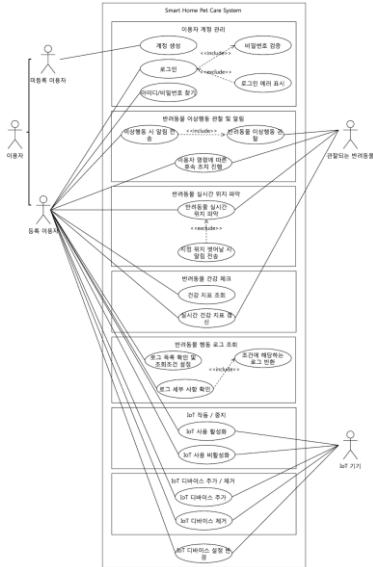


4. SDS



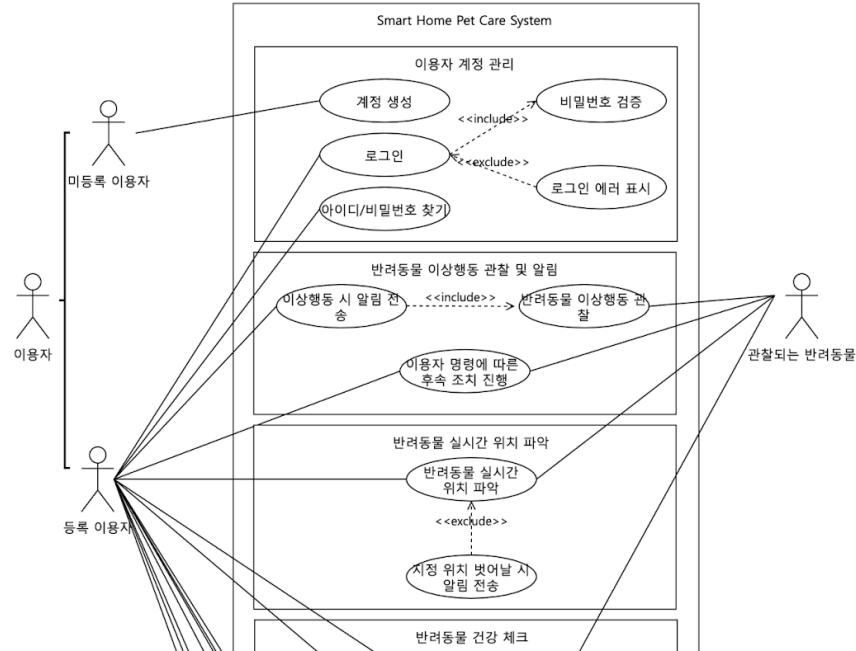
전체 구조

51



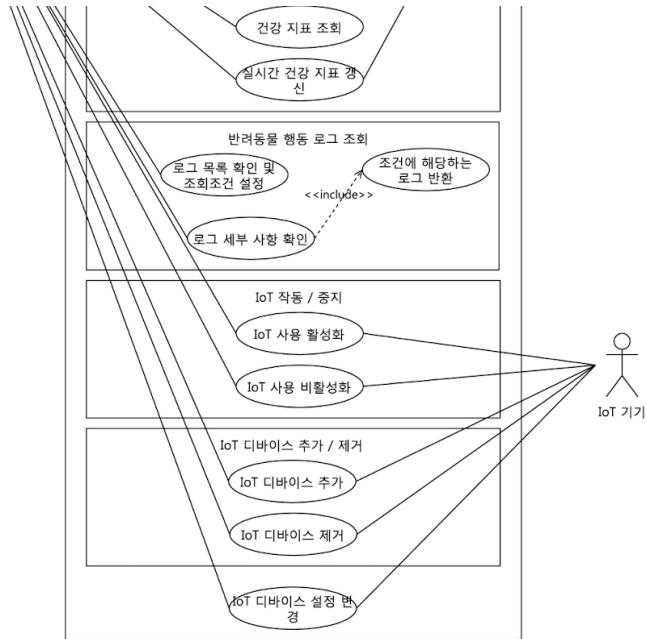
전체 구조 - 1

52



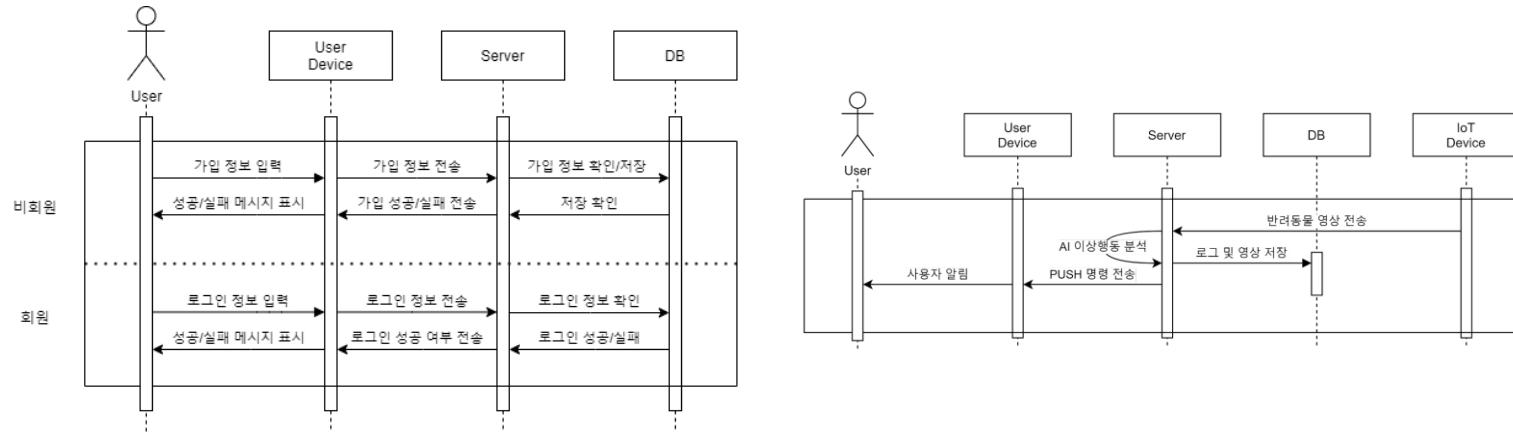
전체 구조 - 2

53



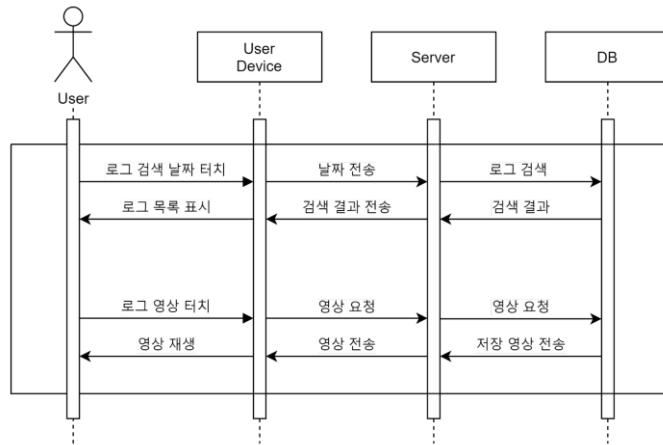
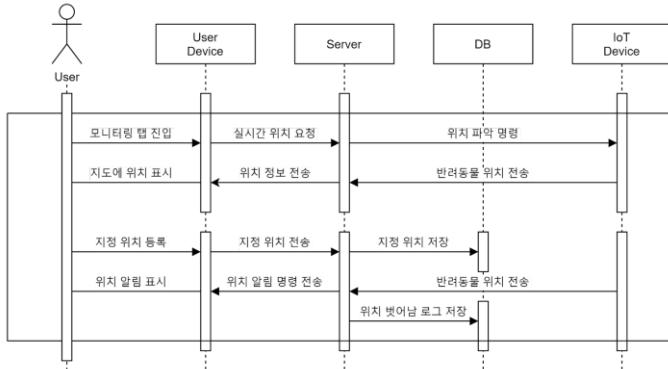
Sequence Diagram

54



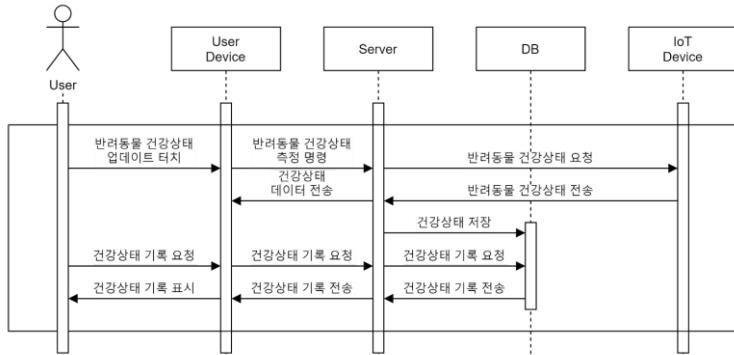
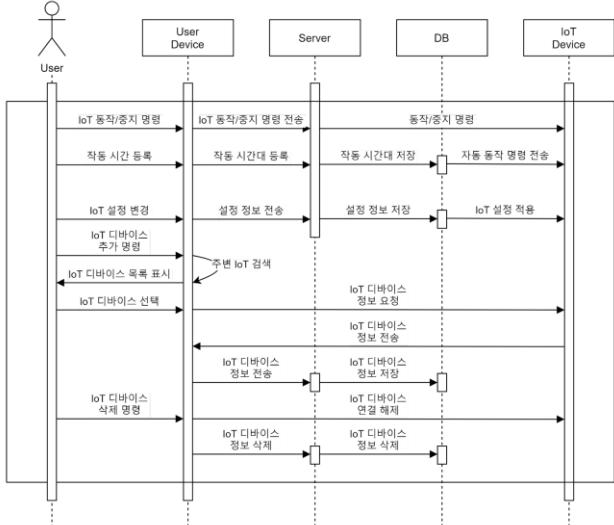
Sequence Diagram

55

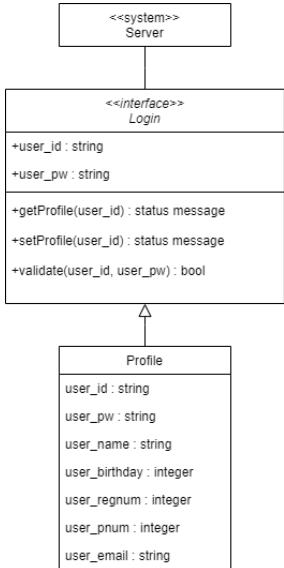


Sequence Diagram

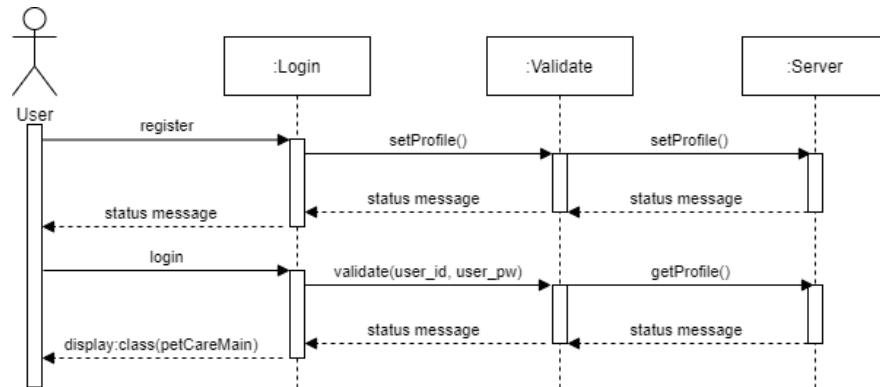
56



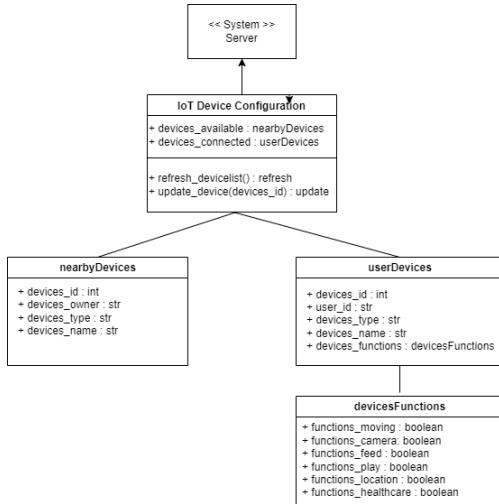
- Class Diagram



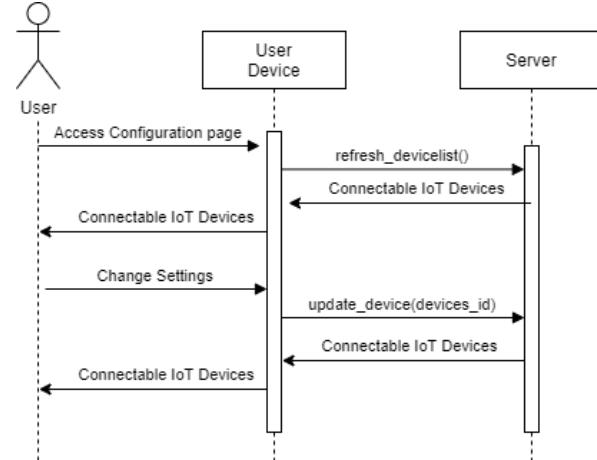
- Sequence Diagram



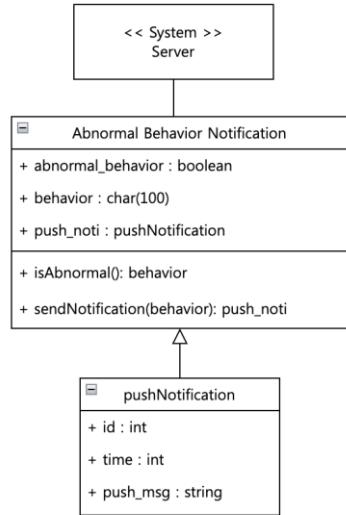
- Class Diagram



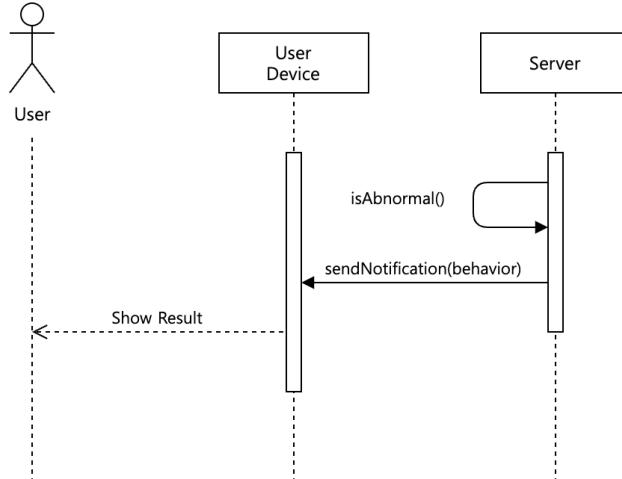
- Sequence Diagram



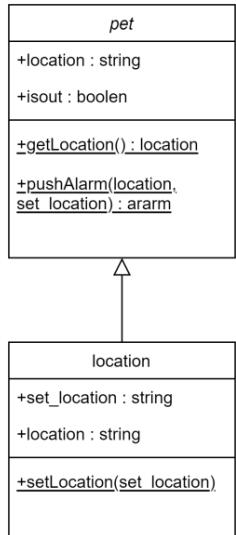
- Class Diagram



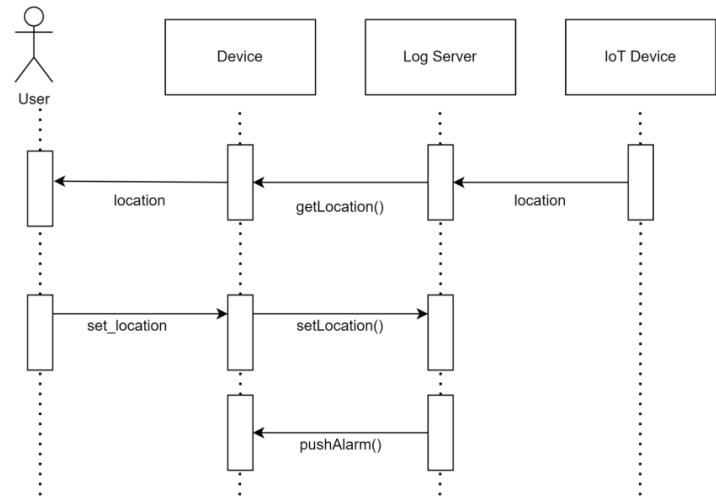
- Sequence Diagram



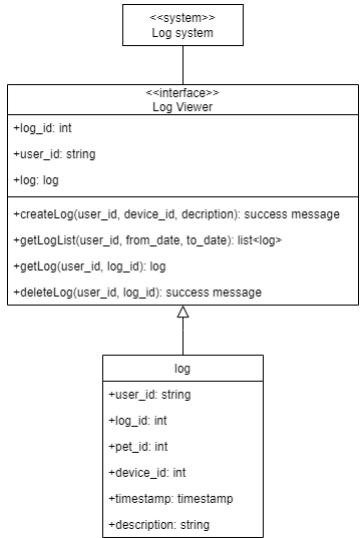
- Class Diagram



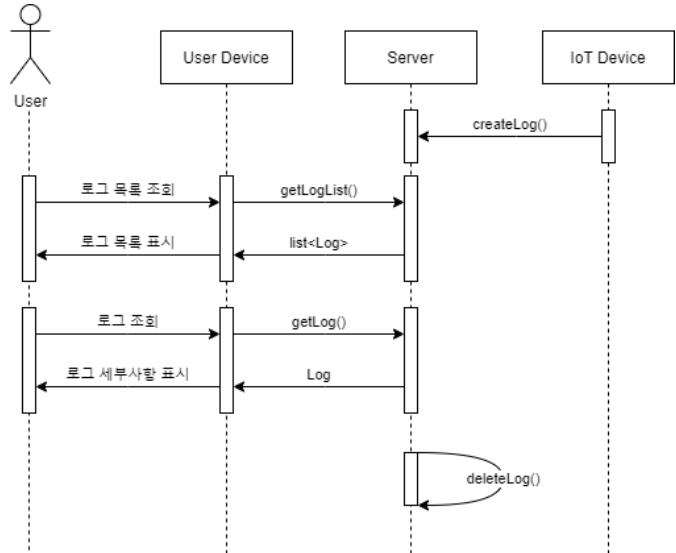
- Sequence Diagram



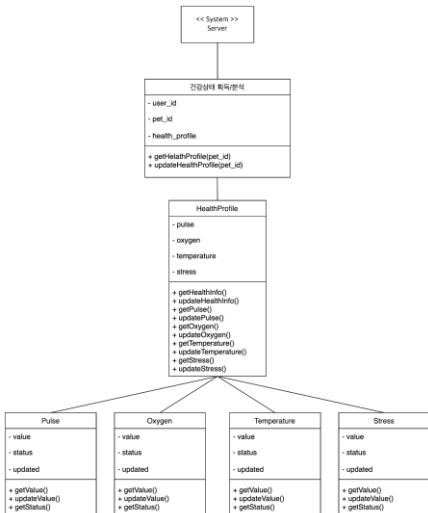
- Class Diagram



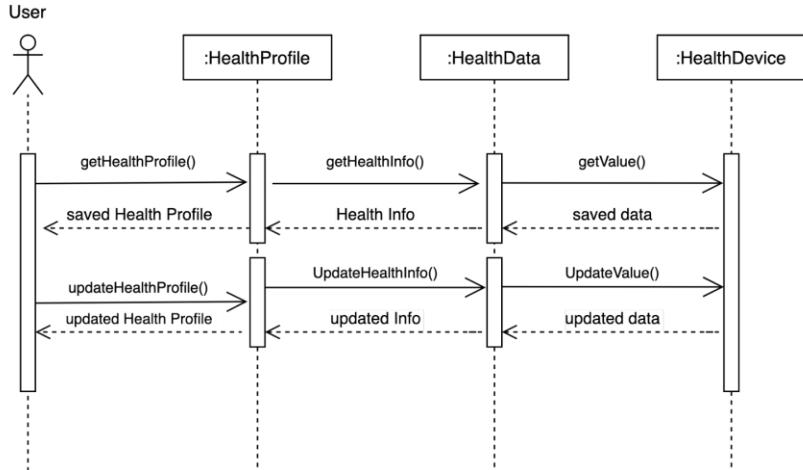
- Sequence Diagram



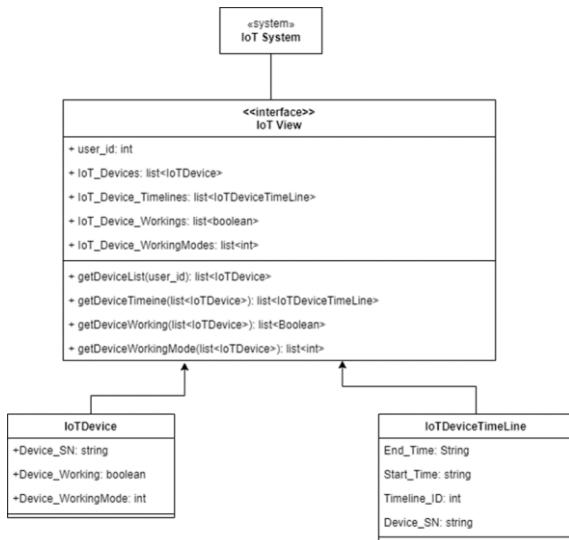
- Class Diagram



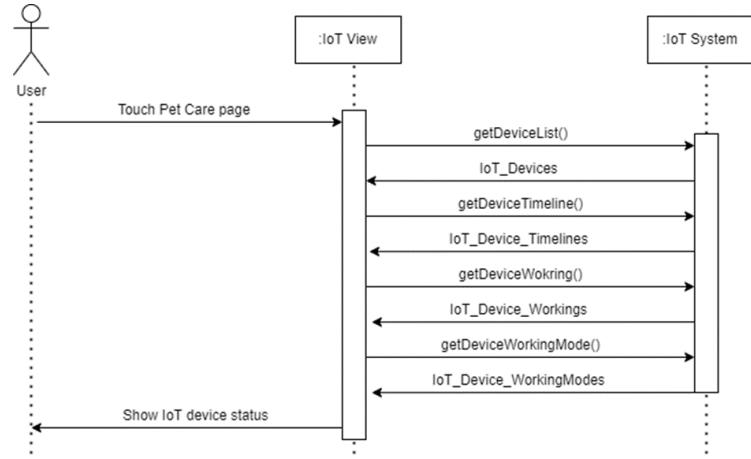
- Sequence Diagram



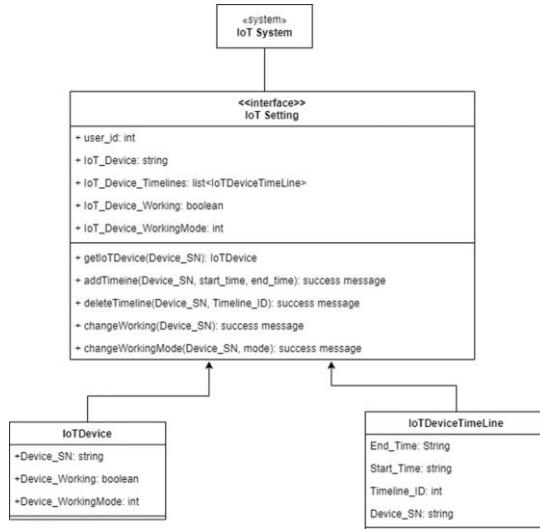
- Class Diagram



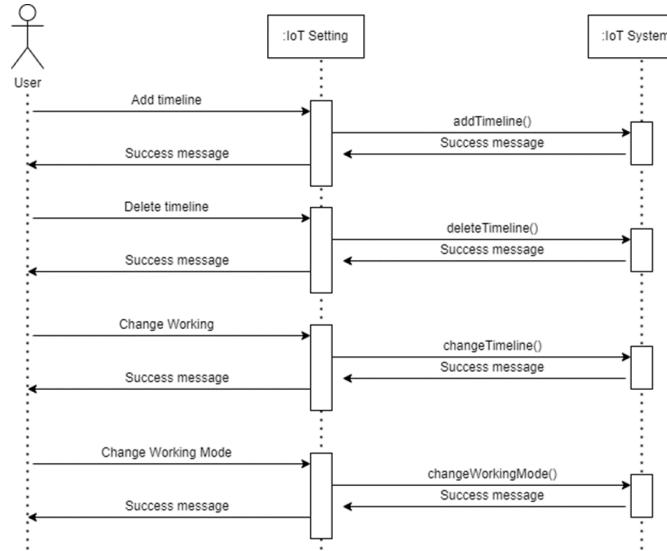
- Sequence Diagram



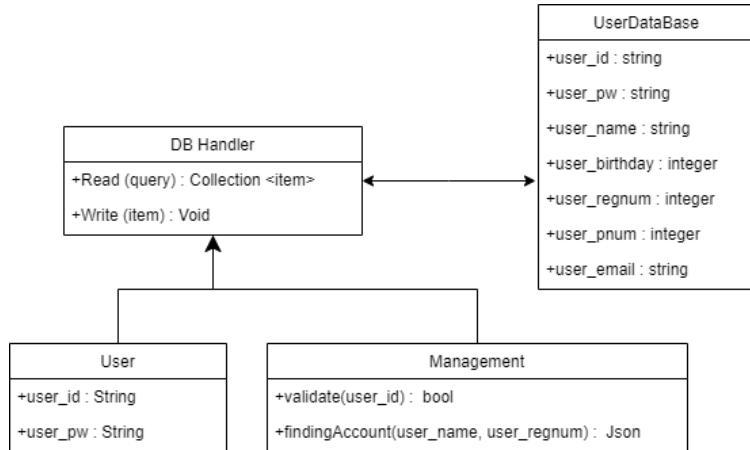
- Class Diagram



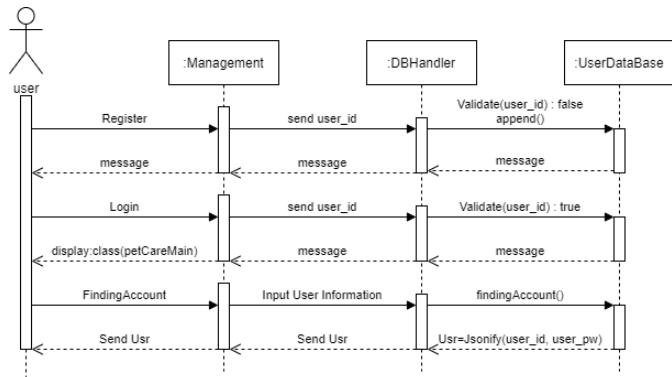
- Sequence Diagram



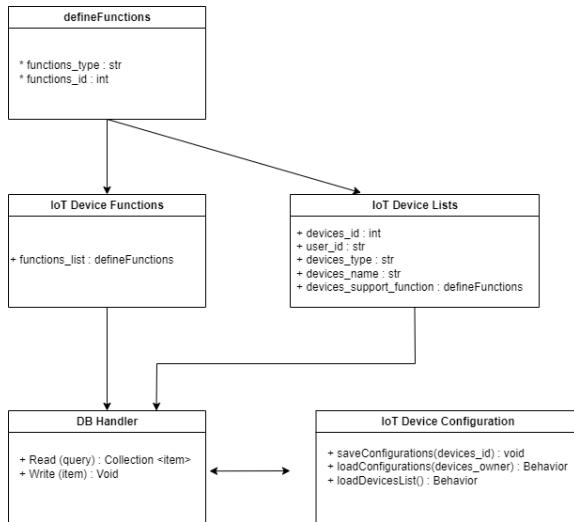
- Class Diagram



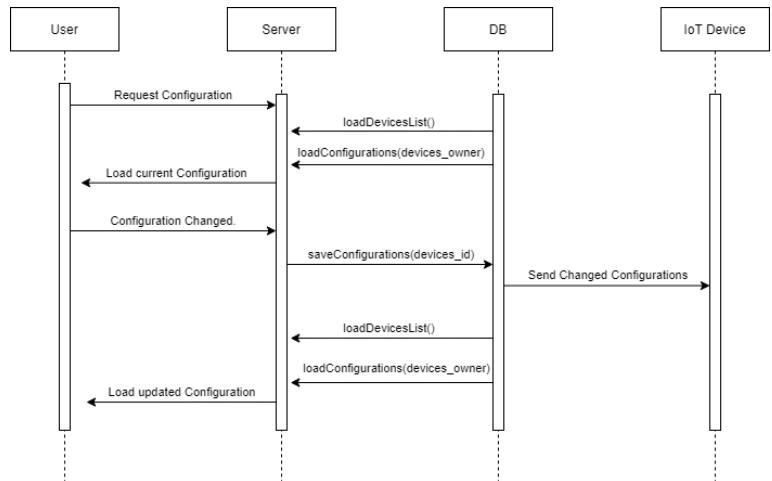
- Sequence Diagram



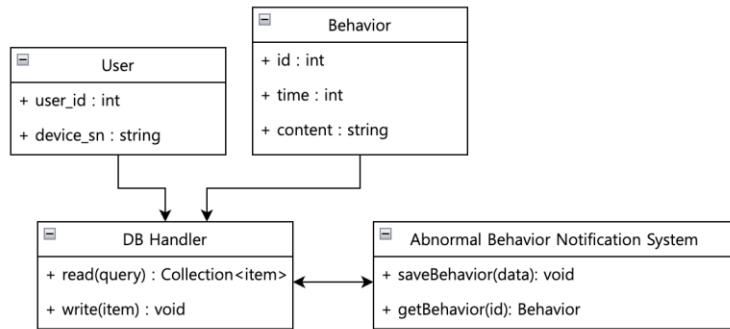
- Class Diagram



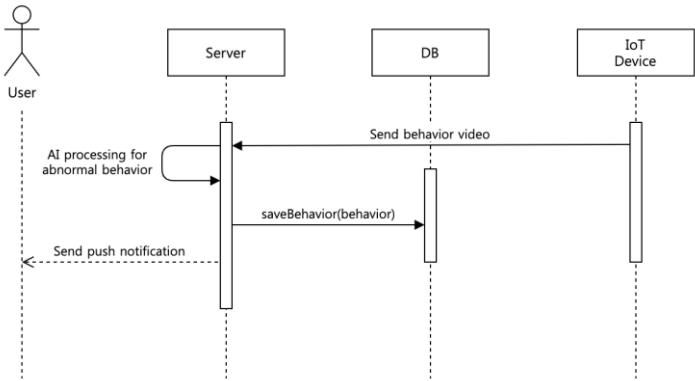
- Sequence Diagram



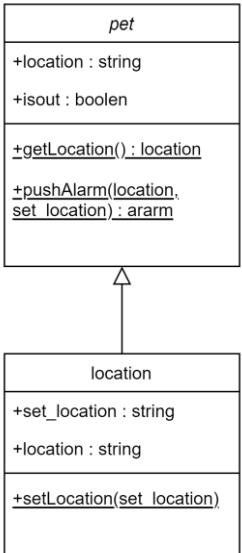
- Class Diagram



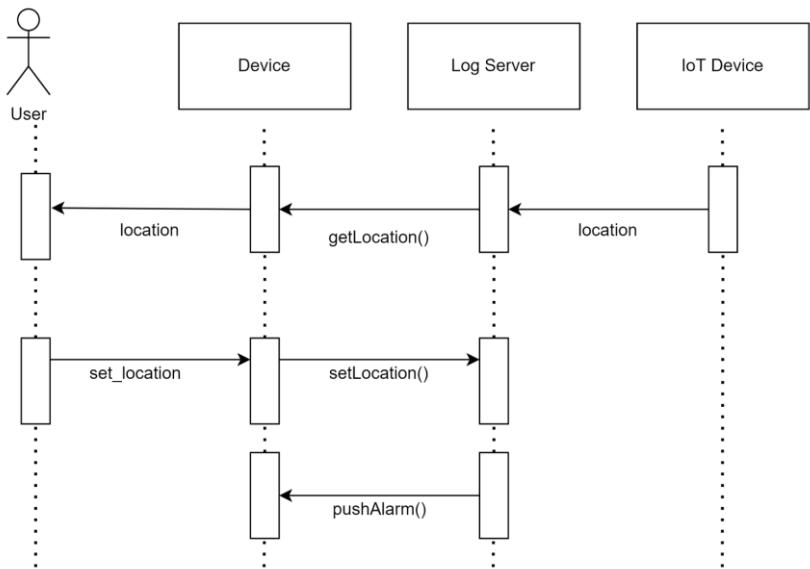
- Sequence Diagram



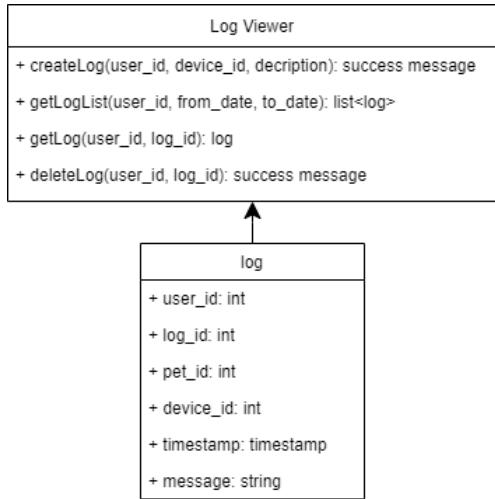
- Class Diagram



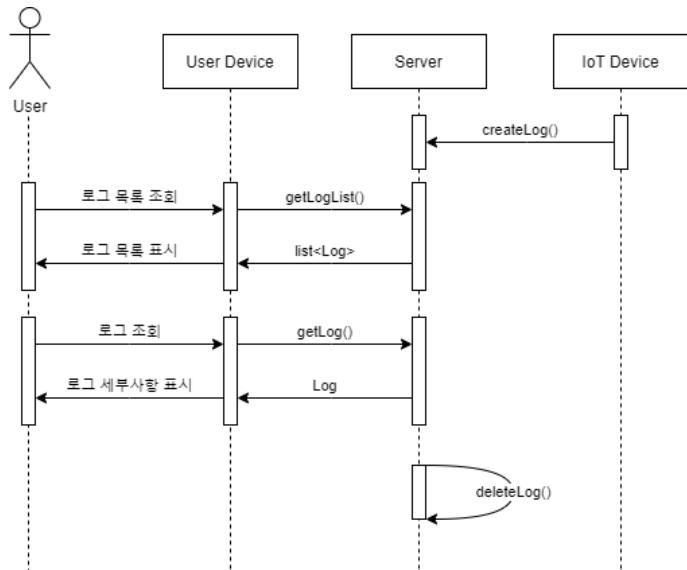
- Sequence Diagram



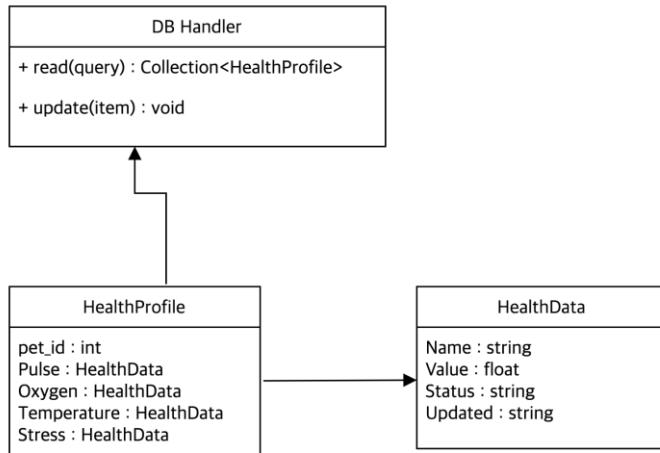
- Class Diagram



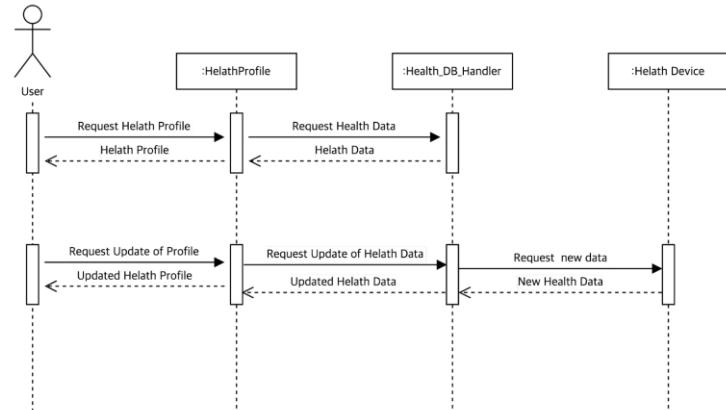
- Sequence Diagram



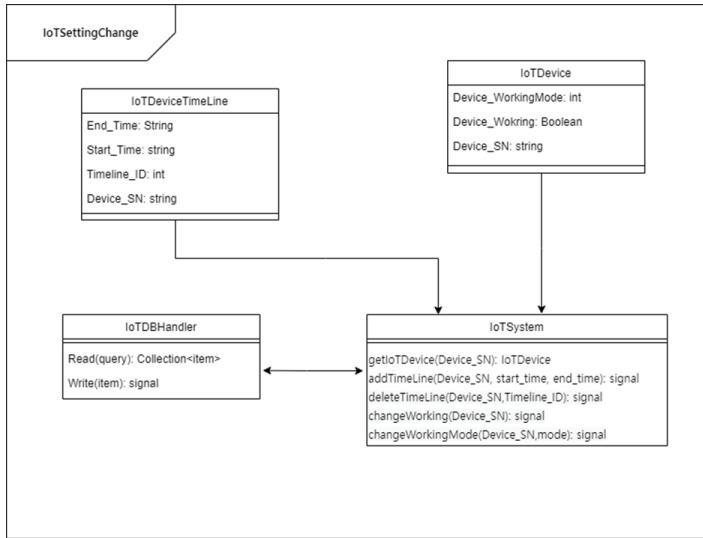
- Class Diagram



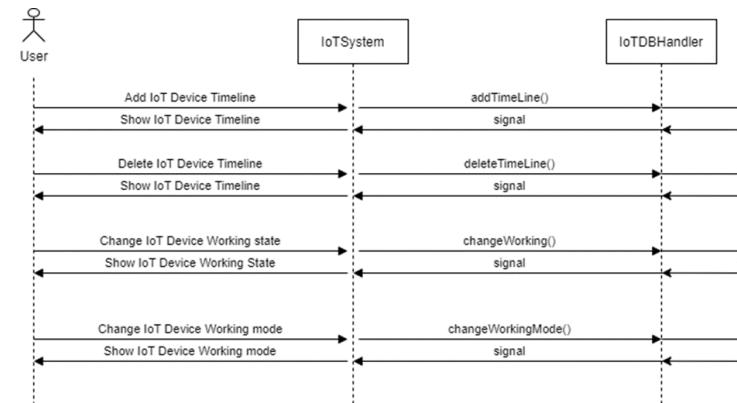
- Sequence Diagram



- Class Diagram

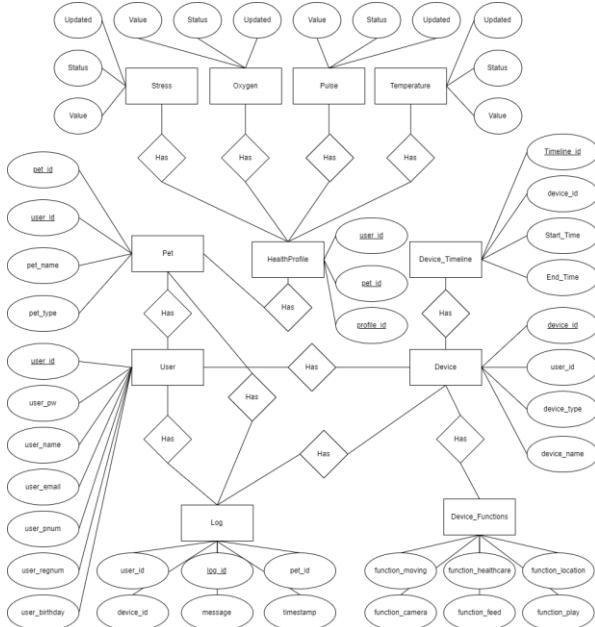


- Sequence Diagram



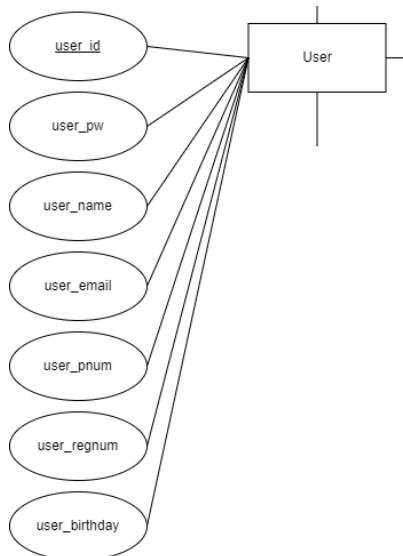
Database Design

72



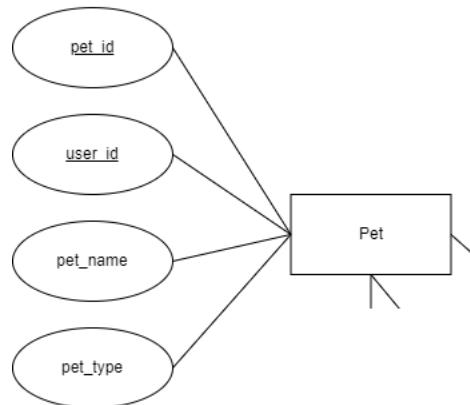
User

- 유저와 관련된 정보 표현
- 아이디, 패스워드, 이름, 휴대폰 번호, 주민등록번호, 생년월일 포함
- 기본키 : user_id
- Device, Pet, Log, HealthProfile Entity와 1:N 관계



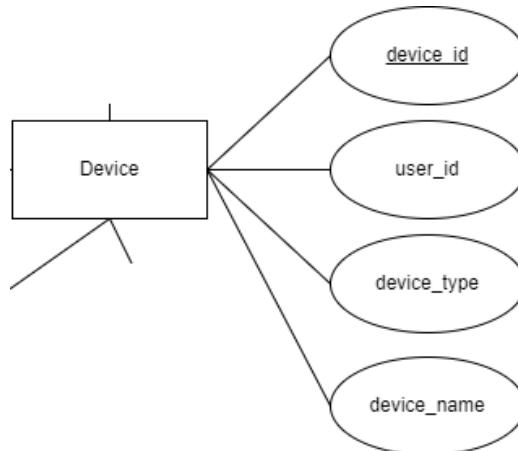
Pet

- 반려동물에 대한 정보 표현
- 반려동물의 이름, 종류 저장
- 기본키 : pet_id
- 외래키 : user_id
- HealthProfile Entity와는 1:1 관계
- Log Entity와는 1:N 관계



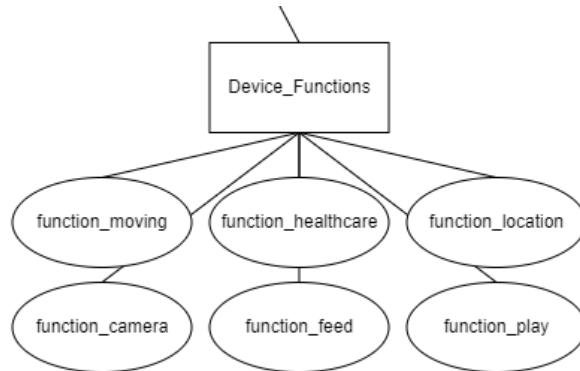
Device

- IoT 디바이스에 대한 정보 표현
- IoT 디바이스의 이름, 종류 저장
- 기본키 : device_id
- 외래키 : user_id
- DeviceFunctions, DeviceTimeline Entity와는 1:1 관계



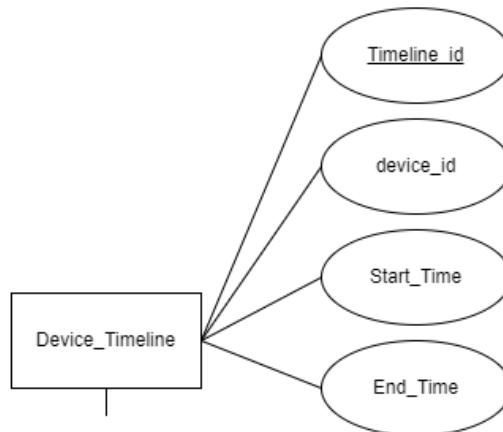
DeviceFunctions

- IoT 디바이스가 수행하는 기능에 대한 정보 표현
- IoT 디바이스의 이동, 건강상태, 위치, 카메라, 먹이 급여, 놀이 기능의 여부 (Boolean)
- Device Entity와는 1:1 관계



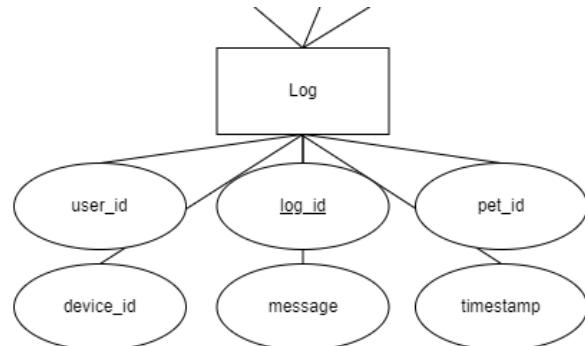
DeviceTimeline

- IoT 디바이스의 작동 스케줄에 대한 정보 표현
- IoT 디바이스의 작동 시작/정지 시간 저장
- Device Entity와는 1:1 관계



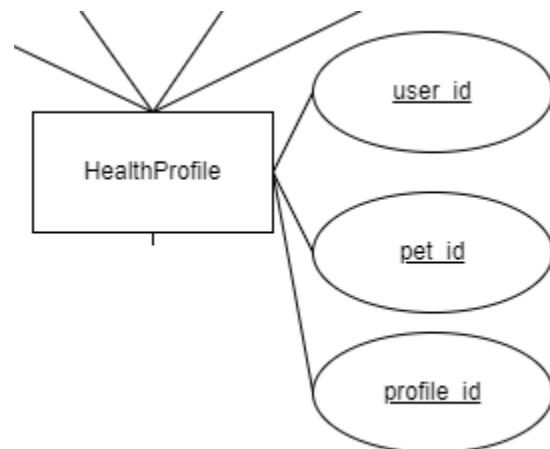
Log

- IoT 디바이스에서 감지된 반려동물의 행동 로그에 대한 정보 표현
- Message : 로그 세부 내용
- Timestamp : 로그 발생 시간
- 기본키 : log_id
- 외래키 : user_id, pet_id, device_id



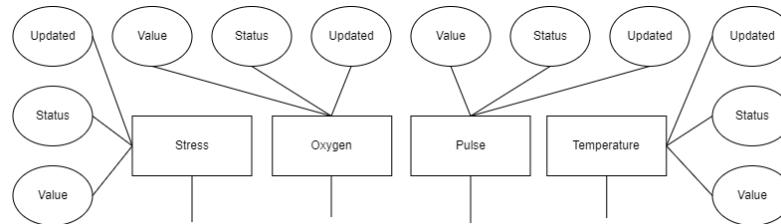
HealthProfile

- IoT 디바이스에서 감지된 반려동물의 건강 상태에 대한 정보 표현
- 기본키 : profile_id
- 외래키 : user_id, pet_id
- User Entity와 1:N 관계
- Pet Entity와 1:1 관계
- Stress/Oxygen/Pulse/Temperature Entity와 1:1 관계



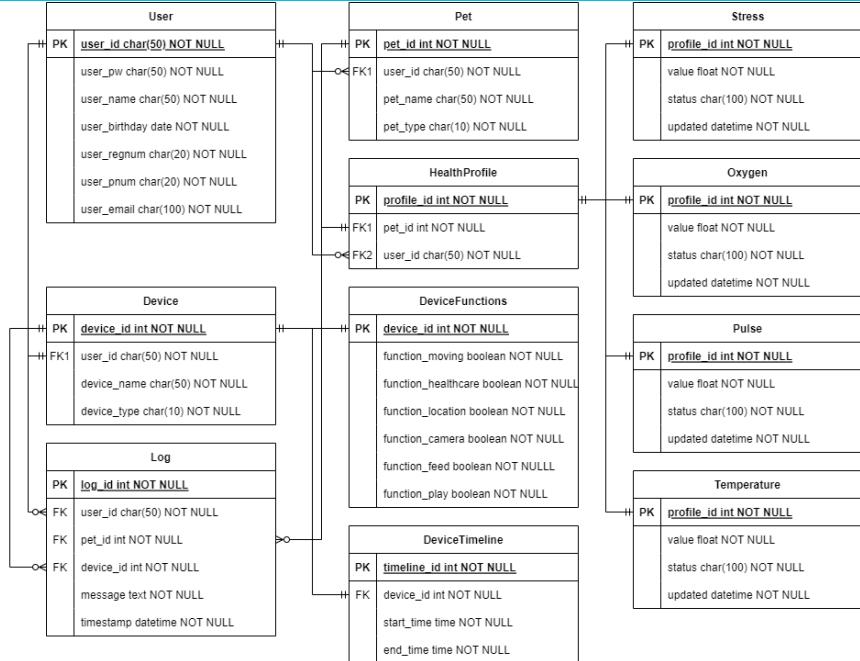
Stress/Oxygen/Pulse/ Temperature

- IoT 디바이스에서 감지된 반려동물의 건강 상태의 세부 정보 표현
- 모든 Entity가 value, status, update 가짐
- 각각 HealthProfile Entity와 1:1 관계



Relational Schema

81



```
CREATE TABLE IF NOT EXISTS `User` (
  `user_id` VARCHAR(50) NOT NULL,
  `user_pw` VARCHAR(50) NOT NULL,
  `user_name` VARCHAR(50) NOT NULL,
  `user_birthday` DATE NOT NULL,
  `user_regnum` VARCHAR(20) NOT NULL,
  `user_pnum` VARCHAR(20) NOT NULL,
  `user_email` VARCHAR(100) NOT NULL,
PRIMARY KEY (`user_id`))
```

User	
PK	<u>user_id</u> char(50) <u>NOT NULL</u>
	user_pw char(50) NOT NULL
	user_name char(50) NOT NULL
	user_birthday date NOT NULL
	user_regnum char(20) NOT NULL
	user_pnum char(20) NOT NULL
	user_email char(100) NOT NULL

```
CREATE TABLE IF NOT EXISTS `Pet` (
  `pet_id` INT NOT NULL AUTO_INCREMENT,
  `user_id` VARCHAR(50) NOT NULL,
  `pet_name` VARCHAR(50) NOT NULL,
  `pet_type` VARCHAR(10) NOT NULL,
  PRIMARY KEY(`pet_id`, `user_id`),
  CONSTRAINT `fk_Pet_User`
    FOREIGN KEY(`user_id`)
    REFERENCES `mydb`.`User`(`user_id`))
```

Pet	
PK	<u>pet_id int NOT NULL</u>
FK1	user_id char(50) NOT NULL
	pet_name char(50) NOT NULL
	pet_type char(10) NOT NULL

SQL DDL - Device

84

```
CREATE TABLE IF NOT EXISTS `Device` (
  `device_id` INT NOT NULL
  AUTO_INCREMENT,
  `user_id` VARCHAR(50) NOT NULL,
  `device_name` VARCHAR(50) NOT NULL,
  `device_type` VARCHAR(10) NOT NULL,
  PRIMARY KEY(`device_id`, `user_id`),
  CONSTRAINT `fk_Device_User1`
    FOREIGN KEY(`user_id`)
    REFERENCES `mydb`.`User` (`user_id`))
```

Device	
PK	<u>device_id int NOT NULL</u>
FK1	user_id char(50) NOT NULL
	device_name char(50) NOT NULL
	device_type char(10) NOT NULL

SQL DDL - DeviceFunctions

85

```
CREATE TABLE IF NOT EXISTS
`DeviceFunctions` (
`device_id` INT NOT NULL,
`function_moving` TINYINT NOT NULL,
`function_healthcare` TINYINT NOT NULL,
`function_location` TINYINT NOT NULL,
`function_camera` TINYINT NOT NULL,
`function_feed` TINYINT NOT NULL,
`function_play` TINYINT NOT NULL,
PRIMARY KEY(`device_id`),
CONSTRAINT
`fk_DeviceFunctions_Device1`
FOREIGN KEY(`device_id`)
REFERENCES `mydb`.`Device`(`device_id`))
```

DeviceFunctions	
PK	<u>device_id int NOT NULL</u>
	function_moving boolean NOT NULL
	function_healthcare boolean NOT NULL
	function_location boolean NOT NULL
	function_camera boolean NOT NULL
	function_feed boolean NOT NULL
	function_play boolean NOT NULL

SQL DDL – DeviceTimeline

86

```
CREATE TABLE IF NOT EXISTS
`DeviceTimeline` (
  `timeline_id` INT NOT NULL
  AUTO_INCREMENT,
  `Device_device_id` INT NOT NULL,
  `start_time` TIME NOT NULL,
  `end_time` TIME NOT NULL,
  PRIMARY KEY (`timeline_id`),
  `Device_device_id`),
CONSTRAINT `fk_DeviceTimeline_Device1`
  FOREIGN KEY(`Device_device_id`)
  REFERENCES `mydb`.`Device`
(`device_id`))
```

DeviceTimeline	
PK	<u>timeline_id int NOT NULL</u>
FK	device_id int NOT NULL start_time time NOT NULL end_time time NOT NULL

SQL DDL – Log

87

```
CREATE TABLE IF NOT EXISTS `mydb`.`Log`  
(  
    `log_id` INT NOT NULL AUTO_INCREMENT,  
    `user_id` VARCHAR(50) NOT NULL,  
    `pet_id` INT NOT NULL,  
    `device_id` INT NOT NULL,  
    `message` TEXT NOT NULL,  
    `timestamp` DATETIME NOT NULL,  
    PRIMARY KEY (`log_id`, `user_id`,  
    `pet_id`, `device_id`),  
    CONSTRAINT `fk_Log_Device1`  
        FOREIGN KEY(`device_id`)  
            REFERENCES `mydb`.`Device`  
            (`device_id`)  
    CONSTRAINT `fk_Log_User1`  
        FOREIGN KEY(`user_id`)  
            REFERENCES `mydb`.`User`(`user_id`)  
    CONSTRAINT `fk_Log_Pet1`  
        FOREIGN KEY(`pet_id`)  
            REFERENCES `mydb`.`Pet`(`pet_id`))
```

Log	
PK	<u>log_id int NOT NULL</u>
FK	user_id char(50) NOT NULL
FK	pet_id int NOT NULL
FK	device_id int NOT NULL
	message text NOT NULL
	timestamp datetime NOT NULL

SQL DDL – HealthProfile

88

```
CREATE TABLE IF NOT EXISTS
`HealthProfile` (
`profile_id` INT NOT NULL
AUTO_INCREMENT,
`pet_id` INT NOT NULL,
`user_id` VARCHAR(50) NOT NULL,
PRIMARY KEY (`profile_id`, `pet_id`,
`user_id`),
CONSTRAINT `fk_HealthProfile_Pet1`
FOREIGN KEY(`pet_id`)
REFERENCES `mydb`.`Pet`(`pet_id`)
CONSTRAINT `fk_HealthProfile_User1`
FOREIGN KEY(`user_id`)
REFERENCES `mydb`.`User`(`user_id`)
)
```

HealthProfile	
PK	<u>profile_id int NOT NULL</u>
FK1	pet_id int NOT NULL
FK2	user_id char(50) NOT NULL

SQL DDL – Stress/Oxygen/Pulse/Temperature

89

```
CREATE TABLE IF NOT EXISTS `Stress` (
    `profile_id` INT NOT NULL,
    `value` FLOAT NOT NULL,
    `status` VARCHAR(100) NOT NULL,
    `updated` DATETIME NOT NULL,
    PRIMARY KEY(`profile_id`),
    CONSTRAINT `fk_Stress_HealthProfile1`
        FOREIGN KEY(`profile_id`)
            REFERENCES `mydb`.`HealthProfile`(`profile_id`)
)
```

Stress	
PK	<u>profile_id int NOT NULL</u>
	value float NOT NULL
	status char(100) NOT NULL
	updated datetime NOT NULL

Reference

- » Team 7, 2021 Spring, Software Design Document, SKKU
- » Team 12, 2021 Spring, Software Design Document, SKKU
- » IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In IEEEExplore Digital Libraray. (https://web.cs.dal.ca/~hawkey/3130/srs_template-ieee.doc)
- » Team 1. “Project Highlight Requirement”. SKKU, Last Modified: April 25. 2021. (https://github.com/skkuse/2021spring_41class_team1/blob/main/doc/Project%20Highlight_Requirement%2020.4.pdf)
- » Flutter API. (<https://github.com/PDFTron/pdftron-flutter>)

감사합니다!

